Volume 2

Pages 224 - 429

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

BEFORE THE HONORABLE WILLIAM H. ALSUP

ORACLE AMERICA, INC., Plaintiff, VS. GOOGLE, INC., Defendant.	No. C 10-3561 WHA
	San Francisco, California
	April 17, 2012

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

APPEARANCES:

For Plaintiff:

MORRISON & FOERSTER
755 Page Mill Road
Palo Alto, California 94304
BY: MICHAEL A. JACOBS, ESQUIRE
KENNETH A. KUWAYTI, ESQUIRE
MARC DAVID PETERS, ESQUIRE
DANIEL P. MUINO, ESQUIRE

BOIES, SCHILLER & FLEXNER
333 Main Street
Armonk, New York 10504
BY: DAVID BOIES, ESQUIRE
ALANNA RUTHERFORD, ESQUIRE

(Appearances continued on next page)

Reported By: Katherine Powell Sullivan, RPR, CRR, CSR #5812
Debra L. Pas, RMR, CRR, CSR #11916
Official Reporters - U.S. District Court

APPEARANCES (CONTINUED):

For Plaintiff:

BOIES, SCHILLER & FLEXNER
1999 Harrison Street, Suite 900
Oakland, California 94612
BY: WILLIAM FRED NORTON, ESQUIRE
STEVEN C. HOLTZMAN, ESQUIRE

ORACLE AMERICA, INC.
500 Oracle Parkway
Redwood Shores, California 94065
BY: ANDREW C. TEMKIN, CORPORATE COUNSEL
DORIAN DALEY, GENERAL COUNSEL

For Defendant:

KEKER & VAN NEST
633 Battery Street
San Francisco, California 94111-1809
BY: ROBERT ADDY VAN NEST, ESQUIRE
CHRISTA MARTINE ANDERSON, ESQUIRE
DANIEL PURCELL, ESQUIRE
MICHAEL S. KWUN, ESQUIRE

KING & SPALDING LLP
1185 Avenue of the Americas
New York, New York 10036-4003
BY: BRUCE W. BABER, ESQUIRE

GOOGLE, INC.
1600 Amphitheatre Parkway
Mountain View, California 94043
BY: RENNY HWANG, LITIGATION COUNSEL

For Dr. Kearl:

FARELLA BRAUN & MARTEL LLP
235 Montgomery Street, 30th floor
San Francisco, California 94104
BY: JOHN L. COOPER, ESQUIRE

Also Present:

SAFRA CATZ, President and CFO
Oracle Corporate Representative

CATHERINE LACAVERA
Google Corporate Representative

— — —

P R O C E E D I N G S

APRIL 17, 2012 **7:27 a.m.**

(Proceedings held in open court, outside the presence and hearing of the jury.)

THE COURT: Good morning. Please be seated.

May I see the letter you all wanted me to look at no one bothered to give to me, the one involving Mr. Ellison? I can't rule on it without looking at it.

Mr. Van Nest, this is your motion. You should give me the letter.

MR. VAN NEST: I will, your Honor.

THE COURT: While we're looking for that, do we have the right number of copies of the glossary and the timeline?

MR. JACOBS: Yes, your Honor.

THE COURT: Please hand 12 copies to the clerk.

Mr. Jacobs, have these been cleared with the other side?

MR. JACOBS: Yes, your Honor.

THE COURT: Dawn, at the appropriate time that I will mention to you, please give copies to the jury.

THE CLERK: Okay.

THE COURT: I'd like to see one copy of the timeline myself.

THE CLERK: Okay.

(Whereupon, document was tendered to the Court.)

THE COURT: May I see the letter that you're trying to keep out of the of evidence?

MR. VAN NEST: I guess we pulled it out of the official exhibits.

Do you guys have it at counsel table?

(Whereupon document was tendered to counsel.)

MR. VAN NEST: Thank you.

(Whereupon document was tendered to the Court.)

(Brief pause.)

THE COURT: Let me say a few things first about this. This trial is not Java against Android. This trial is not about all of Java.

To some extent the Court has got to give the plaintiff leeway -- to some extent the Court must give leeway to counsel because the case has a lot of emails that refer to Java.

I need to ask whoever this is over here to please sit down. It's a distraction. This is a trial in the U.S. District Court. It's not the public library. So please, please, when the judge is talking, pay attention.

This is not a trial about Java versus Android. Now,

you've already used this large number. Mr. Van Nest is going to have permission when -- if, if, and when Mr. Ellison testifies or anyone else on the plaintiff's side testifies to large numbers like this, you get to bring up the fact that there are thousands of patents -- I'm talking about now to Mr. Van Nest -- that Java was covered by thousands of patents and only two patents are being asserted here covered by copyrights, but has now boiled down to four things. And if he professes not to realize that, you go to town on cross-examination. You will have a freehand on that.

This is not going to degenerate into an argument piece for large numbers when everyone in this courtroom knows that the cases boil down to a small number, perhaps very important, but small number of items of intellectual property.

It is not Java against Android. And I do not approve of the idea that you can you present that theory to the jury.

As I said, to a limited extent because those emails refer to it as Java, I'm letting you make that argument because you, in fairness as an advocate, need that flexibility, but you may not leave the impression that this case is all of Java against all of Android. It is not.

So, Mr. Van Nest, if and when Mr. Ellison tries to work these large numbers in about what he was willing to pay for Java, you have the Court's permission to bring this back to this case, which is this case is not about Java against

Android. And the Court is going to be diligent and require candid answers. So I give you that heads-up on your side over there so you will know how much you want to open that door.

This letter is not going to come into evidence at the plaintiff's behest. I'm talking about the March 12 letter. It's too self serving. It was an offer for 2 billion, but it's not for Java. It's for Java and lots of other things. It's impossible to say how much of this was allocated to Java.

And, of course, Mr. Ellison will now, I'm sure, if asked by Mr. Boies, of course, he will say it was mostly for Java. That's after-the-fact Monday morning quarterbacking. It didn't say that at the time. This letter did not say that at the time.

So you're not allowed, on the plaintiff's side, to bring this letter to the attention of the jury as some kind of support for the theory that Java is worth $2 billion. So that's the ruling.

All right. Anything you want to bring up with me before we bring the jury in and go to the next opening statement?

MS. ANDERSON: Yes, your Honor. Christa Anderson for Google.

THE COURT: Go ahead.

MS. ANDERSON: I have a brief question for your Honor and request, both related to designations of deposition

testimony for playing at trial.

My first question for your Honor is: Our understanding under the rules is you would like the completed packets with all the designated testimony two days before --

THE COURT: Don't give it to me until -- yes, I do, but I don't want -- yesterday you gave me something and my clerk said, "The lawyers are just giving you this and they are still talking. They think they are going to work it out, but just in case they want to you have all of this." I'm not going to start looking at your write-ups with that kind of message. Your duty is to work through the process and get it down to the bare minimum and then I'm going to rule on it.

MS. ANDERSON: Absolutely. Understood, your Honor.

THE COURT: I will do that very quickly. But don't give it to me in that kind of a mish-mash form that came to me yesterday.

MS. ANDERSON: Yes, your Honor. Actually, I'm not familiar with -- perhaps it was Oracle's counsel turned it in.

But I just wanted to find out from your Honor would you like it the day before we plan to play the testimony or two days before?

THE COURT: Whatever my ground rules say, I have forgotten.

MS. ANDERSON: Okay. Thank you.

THE COURT: Here is what I learned from practice.

I know that I am sometimes tough on the lawyers for good reason, but I learned this from practice.

I'm not tough in one respect that other judges will make you designate weeks in advance of the trial. I tried a lot of cases as a lawyer and every time you designated something, it turned out by the time you really wanted to use it, it went down to about 10 percent of what you originally had thought you had wanted to use because the points got covered in some other way and it was not worth putting all that other fluff in there.

So the best way to do it is to postpone the depo designations until the last possible minute so it will be down to the real kernel of what you want to use. So that's when I want to see it and that's what those ground rules are designed to do, is to save you all that work until it really matters, but when -- at that point it does matter and you should give me just what you propose to use teed up with the objections and so forth. So I'm trying to make it easier for you, not harder.

Okay?

MS. ANDERSON: We appreciate that your Honor. And that brings me to my last request for your Honor.

Relatedly, we are having an issue with Oracle's counsel vastly overdesignating testimony in these five-day Notice of Deposition designations.

THE COURT: They will then read every word of it.

They will not be allowed to withdraw it. They will waste their time putting in large amounts of fluff. If that is what is going on, they are going to use their time up accordingly, that's fine. Why are you complaining? But that is an old trick and I'm not going to allow it. They are going to read every word of it, if that's what they do.

MS. ANDERSON: All right. Thank you, your Honor.

THE COURT: All right. So I'm going to let them fix it by the end of today, but otherwise you're going to read every word of it. It will burn up your good time, and don't complain to me. It's an old trick to overdesignate. I understand that. Do not do that.

MS. ANDERSON: We appreciate that. Thank you, your Honor.

THE COURT: All right. What's your next point?

MR. VAN NEST: We have one other issue, your Honor, which Mr. Paige is going to address on exhibits that may come up with Mr. Larry Page today. Mr. Paige has been doing the meet-and-confer with Oracle on it.

THE COURT: How many Pages do we have here?

MR. PAIGE: I spell it right your Honor, P-A-I-G-E.

THE COURT: All right.

MR. PAIGE: Good morning. We have an issue, potential issue with Trial Exhibit 431. It was originally in Mr. Page's --

THE COURT: Hand it up to me. I don't remember it.

MR. PAIGE: Of course, your Honor.

(Whereupon, document was tendered to the Court and counsel.)

THE COURT: Thank you. 431.

MR. PAIGE: It was originally in Mr. Page's, P-A-G-E, designations for today. It's since been removed, so it's taken out that issue of having it used with him at -- in his deposition, but Oracle still says they are not certain whether they are going to use it with him when he appears live today. And so we wanted to raise with your Honor the fact that pages three through six of this document --

THE COURT: Who does Mr. Page work for?

MR. PAIGE: He works for Google, your Honor.

THE COURT: And why are you complaining about this then? This is a party admission, isn't it?

MR. PAIGE: Your Honor, it's a question of what's in here contains estimates going forward into 2013. This is highly sensitive information that Google has not released to the public --

THE COURT: Oh, come on. Oh, come on. This is a public trial. Denied. Denied.

MR. PAIGE: Your Honor, your Honor --

THE COURT: Denied. Denied.

MR. PAIGE: It's not relevant to the phase, any issue

in Phase 1.

THE COURT: Too bad. It's a public trial. If they want to bring it out, they are going to be allowed to do it.

MR. PAIGE: It's 403 evidence, your Honor, in Phase 1.

THE COURT: It's not. It's not. They are going to be allowed to do that.

MR. PAIGE: Thank you, your Honor.

THE COURT: Okay. What else do you want to raise?

MR. JACOBS: We have one question, your Honor.

THE COURT: Sure.

MR. JACOBS: It has to do with exhibits and whether exhibits must come in through a witness or not.

We have party admission exhibits. We would like at an appropriate time, maybe early in the case move, to into evidence, draw whatever objection we're going to get at that time. We don't think it will be very compelling and we don't think -- we think on reflection Google may decide not to object at all, and then with the admission publish the document to the jury.

I think this is particularly important given your admonition that they have to look at evidence, not at attorney argument. So it seemed to us to be advisable to move some of these exhibits -- some of these documents into evidence early.

THE COURT: Okay. Here is the way that part works.

It's okay if the lawyers agree that something can come into evidence out objection. For example, if you want to move right now what that three or four exhibits be placed into evidence, I will ask the other side and they probably would -- I don't know what they will say. Let's say they say, "Fine, no objection." So then we make a note that those are in evidence, and then at some appropriate point I would allow you to publish them to the jury without any comment. You wouldn't have any witness, I guess, to explain what it is, but they would be in evidence. Maybe could use -- have some witness explain what they are. So as far as I'm concerned, you have that flexibility as long as the other side agrees.

If the other side does not agree and says, "No, we object to it coming into evidence," I listen -- you know, they have to -- let's say they say it's hearsay on they say it's not a business record or whatever the objection is, then you have to lay the foundation. Do it the hard way. So it would not just come in by Court order. I would have to have foundation for it.

So maybe the best way to do it would be take an example right now of one you would like to use in that way and then we can see how it works.

MR. JACOBS: Okay, your Honor. Give us just a minute.

MR. VAN NEST: Your Honor, could I make a suggestion

on this --

THE COURT: Okay.

MR. VAN NEST: (Continuing) -- that I think will save time?

THE COURT: Sure.

MR. VAN NEST: We didn't understand that that procedure could work. In light of that, if they have some exhibits they would like to discuss with us during the break or whenever, we can do that. They simply sent us a long list of exhibits not attached to a witness and said, "We're going to use these." And I said, "Wait a minute. I thought they have to come in through a witness." We're certainly willing to do --

THE COURT: Usually they do. If you do it the old-fashioned way, they have to come in through a witness. But in almost all trials there are a few documents -- I wouldn't say hundreds of documents. I would say maybe in a trial like this two dozen in the entire trial, where one side will get two dozen in by stipulation. The other side would get two dozen in by stipulation and at the right time you show them to the jury, but I would not think 100 documents per side.

MR. JACOBS: We sent over 14, your Honor.

THE COURT: Fourteen is in the ballpark of reasonableness. I think you would get the same courtesy from them. I encourage it.

But at the same time if you have a legitimate objection -- and there will be, of course, legitimate objections -- you're entitled to assert your objection.

MR. VAN NEST: With that guidance, your Honor, what I suggest is that the parties meet-and-confer and we perhaps can come up with an agreed list and get them in.

THE COURT: But they have got to put on their witnesses soon. So you should do that at the very first opportunity so that they are not prejudiced.

MR. VAN NEST: We'll do it. Thank you, your Honor.

MR. JACOBS: Thank you, your Honor.

THE COURT: All right. What else is next? Can we see if all members of the jury are present?

MR. JACOBS: It's a tiny thing, your Honor, but it's -- well, it's worth raising. You had asked for a binder of key documents. We each gave you 10.

THE COURT: That's great. That's all I need.

MR. JACOBS: Okay.

THE COURT: If I need more, I will ask for it, but 10 is -- you know, I've learned the hard way. Joe Alioto used to try a case. He only needed two documents. One was the worst document in the file of the other side and the other was the Magna Carta. And he never lost a case.

You know, I need more than that. I need 20 documents in a case like this. But if I need more than that, I'm going

to see what they are and you will give them to me and I will ask for them. I have so little space up here, I can't deal with about more than 20 documents. So that's fine.

MR. JACOBS: Thank you, your Honor.

THE COURT: Yes? They are here?

All right. Before we get started, are you prepared to go with your opening statement, Mr. Van Nest?

MR. VAN NEST: I am, your Honor. I would just a moment to set it up.

THE COURT: How long do you need?

MR. VAN NEST: About 30 seconds.

THE COURT: All right. You set up and then we'll bring the jury in.

(Brief pause.)

THE COURT: I'm going to give these documents to the jury after they are seated. How is that? Is that okay, Mr. Van Nest?

MR. VAN NEST: That's fine. I will wait for your queue.

THE COURT: Are we now ready?

MR. VAN NEST: Yes.

THE COURT: All set?

Wait, wait, Dawn. I do -- I want to say to all members of the two teams here, and for that matter members of the public. I have a strong view that when someone is speaking

in the courtroom, like a lawyer giving an opening statement to the jury or a closing or examining a witness, that lawyer has the absolute right to the full attention of the jury and any hacking and coughing or paper shuffling, no. You cannot do anything. I don't even like it when people get up and go back and forth out that door because a member of the jury may look and see and lose track of what the thought is. So really, I ask you to be considerate to the counsel who have the floor and to be absolutely quiet.

The other thing I wanted to say, Mr. Cooper, I am going to introduce you and the 706 expert at the appropriate time, but I don't think it's the right time yet. It may be much later in the case.

MR. COOPER: Thank you.

THE COURT: I did not forget. I just think it's too much to lay on the jury this early in the case.

MR. COOPER: We understand. Thank you.

THE COURT: Okay. Thank you.

Are we now ready, Mr. Van Nest?

MR. VAN NEST: Yes, your Honor.

THE COURT: Let bring in the jury.

(Jury enters courtroom at 7:48 a.m.)

THE COURT: Okay. Welcome back and good morning. Have a seat. Everybody over there doing good?

(Jury nodding affirmatively.)

THE COURT: Enjoying the coffee and the Federal donuts?

JUROR THOMPSON: Thank you.

THE COURT: We want to hand something out to you, and Dawn will hand them out now.

(Whereupon, documents were tendered to the jury.)

THE COURT: As I've said to you, we have excellent lawyers in this case and they have done something that I totally approve of and it will help you.

First of all, there is a timeline that you already saw and we're going to give each of you your own individual copy. What I recommend you do is fold it over like this (indicating) and stick it in the back of your little steno pad and whenever a witness is on the stand and refers to a date of some meeting, maybe you can look at this and get a rough idea of where it fits in the overall timeline.

And this is stipulated to; correct, counsel?

MR. JACOBS: Yes, your Honor.

MR. VAN NEST: Yes, your Honor.

THE COURT: So the facts are agreed to by both sides and you can take this as evidence in the case.

Now, after that is handed out, we have got a second one-page document. One-page documents are good. We get things down to one page.

(Whereupon, document was tendered to the jury.)

THE COURT: This is another one that you can fold over. Both sides have agreed to it, right?

MR. JACOBS: Yes, your Honor.

MS. ANDERSON: Yes.

THE COURT: This one looks more complicated. It has a glossary of maybe 30 terms like API, AOSP and so forth. And these are terms that you're going to be seeing in the case, the trial. You don't need to memorize any of this right now. It's just a handy reference sheet for you to look at from time to time in case something is confusing and they are using an abbreviation. These will be terms that come up frequently, okay? So I would recommend you fold that one over and put it in the back of your pad, too.

Now, you can make notes on these. It's up to you. We will not look at your notes. When the case is over, we shred all your notes and never look at them, but this is for your use. These two handouts. We will not look at them and you're free to add, for example, to the timeline. You might put in some important meeting that you think is important that's not on the timeline.

Obviously on a timeline like this, this is just the bare bones. There will be several important events that occurred in this case that are not on the timeline. But if we

put everything on there, it wouldn't be very useful, so we scaled it back for your benefit.

All right. So there we go. In due course all of this will become clearer, but we'll get there one step at a time.

Now, you will remember yesterday we had an opening statement by counsel for Oracle, Mr. Mike Jacobs. And now we're going to have the opening statement by the other side, Mr. Van Nest. And that's spelled V-A-N N-E-S-T.

And I say this about both of the opening statements, what I've already said to you. None of it is evidence yet. The evidence comes when the trial starts with witnesses and so forth. This is an opportunity for you to get a preview of what the lawyers expect will be proven or not proven. And so it is a very important opportunity, but, again, it is not evidence.

At this time on behalf of Google, Mr. Van Nest will give the opening statement. The floor is yours.

MR. VAN NEST: Thank you, your Honor.

OPENING STATEMENT

MR. VAN NEST: Good morning, ladies and gentlemen. Welcome back.

As you know, my name is Bob Van Nest, and it's my responsibility and my privilege during our trial to represent Google, one of the leading consumer technology companies in the world today. This is my opportunity to give you a recap, a

review of the evidence, what we expect it will prove.

So the phenomenal success of Android is due to the hard work of Google engineers, their ingenuity and to Google's decision to build Android using open source, free and open technology, so that Android could be made available to everyone. The whole idea of Android was to get it in the hands of as many consumers as possible.

Now, the Java language, the Java language which was used to build parts of Android, is free and open. It's been in the public domain for years. The Java language was created way back in the 90's and Sun made it public. They made it public without any charge, without any cost because Sun wanted a system, a language system, that could be used by developers all over the world to build products using Java that Sun could sell services and support to. So that language has been public. It's in the public domain and as you heard last night, there is no claim, no claim in this lawsuit that there is anything wrong with using the Java language.

Now, those Application Program Interfaces, those APIs you heard about, they are necessary just to use the language. You'll hear testimony that without the APIs, the language is virtually useless and programmers learning Java, which is taught in college courses all over the country, they learn how to use the APIs when they learn how to use the language. And Sun has always promoted the APIs as a fundamental part of the

language. So the language is free. The APIs are necessary to use the language, so they are free, too.

Google didn't need a license to use Java in Android because the Java language is free and the APIs are free as part of the language. So Google didn't need a license. The actual source code, the computer code within Android that runs all the applications, that was written from scratch by Google engineers or taken from other open source products, other open source platforms, that were open and available for use. So the code in Android, the stuff that makes it run written by Google, original Google work or work of third parties donated to Android.

Now, yes, it's true that at the very beginning of Android, Google and Sun negotiated for a partnership to build Android together, and you'll hear about those negotiations. But when those negotiations failed, Google engineers built Android on their own without any Sun technology whatsoever. It took three years, thousands of engineering hours, and hundreds of millions of dollars to build Android.

Now, when Android was released to the public -- and this happened way back in 2007 -- it was released to the public, posted on a website so everybody could see, including Sun, what was in Android: The Java language, these 37 Java APIs and a whole lot of other stuff.

Sun didn't object. Sun didn't complain. Sun didn't

say, "You need a license." They knew all this was open. Sun's chief executive officer, Jonathan Schwartz, who will be a key witness in this case, he will be presented by Google during our presentation of evidence, Jonathan Schwartz who was running Sun posted on a public Sun website the following:

> "Google, congratulations. We welcome Android to the Java community. We support Android's use of Java. We want to help because Android has strapped a set of rockets onto Java."

Those were the words of Sun's CEO back in 2007. We'll see them a little bit later this morning.

And not only that, but in the years following, 2007, 2008, 2009, both publicly and in private communications directly to the top brass at Google, Sun said, "We support Android's use of Java. What can we do to help? What can we do? We would like to be part of it."

So why are we here? Well, the claims you heard last night, the claim that Google has copied these APIs, that wasn't raised until much later, until after Oracle came on the scene, had purchased Sun, and had tried and failed themselves to get in the smart phone market. So in 2009 Oracle reached an agreement to buy Sun and buy Java. And they were very interested in getting into the smart phone market themselves to compete with the iPhone and compete with Android. So that was a big deal for them.

So first they tried to build their own Java phone. That failed. Then they tried to buy smart phone technology out on the market. That failed. Then they actually tried to partner with Google. They tried to sell Sun technology, Oracle technology to Google, so they could become part of Android, too. That failed. It was too late. Then and only then did Oracle do a complete about-face, a complete about-face and make the claim that Android was wrong. And now what they want is they want to share in Google's success in Android even though they had absolutely nothing to do with the Android platform, the Android product or any of it. And that's why we're here.

It's not about them protecting their I.P. It's not about them protecting the Java community. They want to share in Android's profits without having done a thing to bring that about.

So if you take a look at your monitor, I'm going to review what the evidence will show on four key issues.

First issue: Sun gave the Java language to the public. I don't expect this will even be disputed. There is no claim in the case that using the Java language is wrong.

Second point: Google built Android using free and open technologies. There are 15 million lines of code. 15 million lines of code in Android, all written by Google engineers. They didn't need a license because they didn't use Sun technology to build Android.

Sun publicly approved Android's use of Java. That proves both the first and second points. Sun was fully aware in 2007 of everything that was in Android. It was published on a website. They knew everything that Google was doing. They stood up in public and approved it.

And the final point is that Google made fair use of the Java language APIs in Android. No one before Google had been able to use Java to build a successful smart phone. Nobody. Sun tried and failed and, as I just said a minute ago, Oracle tried and failed.

Google transformed Java. They transformed those APIs into something that would work on a smart phone and that means they made fair use of those APIs in Android.

Let's start on our first point. Sun gave Java language to the public.

Secondly.

Next slide, please.

The Java programming language is open and free for anyone to use. Not a dispute. Back in the 90's Sun made the language open. It's taught in colleges. It's taught to programmers. It's out there. There are thousands of Java developers. It's been around a long time.

Mr. Jacobs said last night, I think his words were, "You can use the Java language to your heart's content," and there is no claim in this case that using the language is wrong

or inappropriate in any way whatsoever.

Now, Mr. Ellison knows this. He'll be one of our witnesses today. I had a chance to take his deposition. That's part of our process here. I asked him:

"QUESTION: And you understand that nobody owns the Java programming language right?

"ANSWER: That's correct.

"QUESTION: Anybody can use that without any royalty at all?

"ANSWER: Correct."

Mr. Ellison is well aware.

And the names of these API packages -- there are 37 packages. And as Mr. Jacobs mentioned, there are classes within the packages and methods. I'll get into that structure of it in a minute. All those names are free to use. You can't copyright a name. So there is no dispute about the names. All these package names, all these -- these are the package names here, 37 of them. All free to use. Anybody can use those. They have been used by developers and companies for years.

All the class names. The classes are organized into packages. They are all free to use.

The method names, the methods are organized into the classes. It's part of the system of organization. Free to use. No dispute about it. Everybody can use these.

Now, what about these APIs? These Application

Programming Interfaces? According to Sun, they are fundamental to the Java language. This is from one of the Sun websites. Java.lang is one of the 37 API packages at issue in this case and this is what Sun says about it:

"Fundamental to the design of the programming language."

You can't use the language without the APIs and the language is free.

This was a book they published back in the 90's. They wanted people to use the APIs. That was the whole point. Let's get people hooked on Java. And part of that was let's show them all these APIs.

Here is Volume 1, *Core Packages*. This describes the libraries that are the foundation of the Java language. The foundation of the language. These are the general purpose libraries fundamental to every Java program.

Not only that, their expert in this case testified -- he will be testifying as a witness here -- and we asked him what about these APIs? What are they?

He said:

"These elements here are like parts of speech. So nouns, verbs, adjectives and so on."

And what he's referring to are the APIs names and packages.

He says:

"Using those parts of speech, package names, classes, interface names, relationships" -- that's what you're going to hear that they are complaining about -- "you use those just like you use the parts of speech."

So imagine being told, "You can use English. You're free to speak English, but don't use the nouns or the verbs or the parts of speech." That's the claim that Oracle is making in this case.

We asked their head architect -- he will be here to testify, too, Mark Reinhold:

"QUESTION: What can you do with the language without the APIs?

"ANSWER: Nothing."

Nothing. This is what he said:

"ANSWER: You would be able to write basic computations that never did any IO."

That means Input/Output. So I could write something, but I couldn't print it. I couldn't put it on my screen. I couldn't communicate with the outside world.

"You couldn't communicate with the outside world or the underlying platform. You could write -- you know, you could do computations on numbers and strings, but you wouldn't be

able to do anything with them."

That's the point that the APIs are necessary to use the language, and the language is out there for free. Okay?

Now, this is to the point that other companies, other groups wrote platforms using the same APIs that they are accusing Google of using years ago with no complaint from Sun.

This is one example. GNU Classpath was a group that created an open source platform for programming in 1998. It used all the APIs that they are complaining about here.

Now, just like Google they wrote their own computer code in the libraries themselves. They used the API structure and the names, but they wrote their own computer codes. Sun said fine. No problem with GNU.

Same thing happened in 2005. This is all before Android. Apache Harmony. They also created a group of Java libraries. They used the APIs, which were open and free and part of the language, and they wrote their own computer code in the libraries. Sun said fine.

Next slide proves it.

Jonathan Schwartz, who will testify live, this is what he said in public back in '07 about Harmony.

"There is no reason that Apache cannot ship Harmony today."

Harmony was the Apache product that used these APIs in the same way that Android does. And companies used Harmony,

IBM. I've listed some of the products that IBM used, they built out of Harmony. Again, with no license from Sun, just like Android.

Okay. So let's talk about Android. I want to take you inside a smart phone and talk about how Android came to be and how Google engineers built it.

Next slide.

So we know what Google is. One of the -- it's the leading search platform in the world and we've heard the word "Google" a number of times in court already. We know what that is.

Google's business model is pretty simple. Give the product free to a consumer and then charge the advertisers. So when we do a search, we have to look at some ads. That's how Google makes money.

And these next two products, same way. Google Maps, same thing. Free to use to consumer. Gmail, free to the consumer. Some ads, we look at the ads. Maybe the ads are a little bit annoying, but that's how -- that's how we get the product for free.

Now, important point. All these products were going great on laptops and desktops, but Google wanted to make them great on a mobile device. And back then in the mid-thousands, all of us were using these cell phones (indicating), right? You're going to hear the word "feature phone." "Feature

phone." I never heard that either, but I use "cell phone." But "feature phone" is what the business guys call it.

And a feature phone is fine. You can make calls and do some texting and play basic games, but you're not going to surf the web on this or check out restaurants or put Google Maps on here. So the whole genesis of Android was: Can we build a platform like this on which our products can be used, you know, more efficiently? Let's put -- let's let people get out on the go.

How do you do that? Research and development. Here is Google campus down in Mountain View. 13,000 Google employees down there. Four out of ten do research. Four out of ten Google employees do R&D for all the products that they make available. And Android was no different. That took three years, dozens of engineers, and thousands of engineering hours, hundreds of millions of dollars.

You're going to hear from two of the top brass at Google. Eric Schmidt is now the chairman at Google. He was the CEO. He is the one that talked directly with Schwartz and will tell you that Schwartz at Sun was always aware of what Google was doing and was always supportive. So this idea that there was a difference between public and private, not going to be borne out by Mr. Schwartz.

Andy Rubin, he's the -- we call him the father of Android. He started a start-up. Android, he had the idea for

this smart phone platform and he was purchased, business was purchased by Google and he will be an important witness.

Let me reintroduce our Google representative while I'm talking about Google. Catherine Lacavera, who is the director of litigation at Google, and she's going to be with us throughout the trial.

These two gentlemen will be witnesses to the hard work that Google did to build Android, which they did by themselves. Bob Lee and Dan Bornstein will both be witnesses at our trial.

Next slide.

What is the Android Platform? Well, the Android platform is the operating system that makes your smart phone run. And Google makes it available for free to two groups that you see on the slide here. They make it available free to Samsung and Motorola and HTC and all the companies that want to build these. They get Android for free and they build the phones that we all -- some of us purchase.

They also give it away to developers. Now, we had one of a prospective jurors yesterday talk about developing an app. Well, if you have a weather app on here or a gardening app or a recipe app or you have a sports app or whatever, game, those are all applications. Those are computer programs that somebody has to write, and developers do that. There are 100,000 Android developers developing applications for Android

and as of now, about 450,000 applications.

So that's how the Android system runs. Open source made available for free.

Next one.

Now, I'm going to put up my timeline. It has a few more entries on it than the agreed-upon timeline, but it covers the same period. And I want to talk now about the story and the timing of Android.

So you'll notice we start here in July of 2005, but all of the Java programming language and GNU Classpath and Apache, they already had occurred. That's in the 90's and earlier than Android.

So in 2005 Android acquired -- Google acquired Android from Mr. Rubin, and Mr. Rubin became a Google employee, and they had a key decision to make. The key decision -- this is really important -- are we going to buy or build? Because anybody, anybody that's launching a new product has that choice. "Buy" means can we partner with another company that already has technology and use that in Android. That's the buy option.

And so Google talked to a lot of companies about that, including Sun. And what they said to Sun was, We'd like to have a partnership and build Android together.

Could I have the next slide?

This is an email. You will see a number of these.

This is an email between Eric Schmidt, who was the boss at Google at the time, and Scott McNealy, who was the chairman of the board at Sun. And what Mr. Schmidt is saying is:

"I'm in a product review. We're looking at a very interesting partnership proposal. We have engaged with Sun's team in an effort to form an alliance. Sun and Google should do this together."

They were talking about a partnership to build Android together. And in that scenario, yes, Google would need a license. If you're going to buy Sun technology, not the APIs, not the language, but the actual code and the libraries, that would have accelerated this process a lot because it took Google three years to do it on their own.

They could have bought it and all those emails you saw -- you'll see a bunch more today -- We need a license. We need a license. That's all in this period early on, '05 and '06. But that never happened. That agreement was never reached. Sun and Google couldn't come to terms.

Google wanted to make Android available as an open source platform and Sun wanted to charge for it. They wanted to charge the handset makers and charge developers and so on for using the platform. So they couldn't reach agreement. And what happened next was Google built Android by itself. They used their own technology, their own people. They didn't use

Sun technology.

They would have used Sun technology if this agreement had come to pass, but it didn't. And so this long period is the time in which it took Google to generate the platform to write the libraries themselves, to write the entire platform.

And let's see what that platform looks like. I want to take you inside the handset now so you can see what the technology is.

What I'm showing here is something that Google publishes on its website. This is the Android Platform. It wasn't made for this litigation. It exists and it's existed since the day Android launched. It shows the layers, and I want to walk through those layer.

The top layer, the application layer, that's what we see as a user. So we have a home page. That's actually an application. We have a phone. That's an application. We have a browser, if you want to surf of the web. That's an application. These applications were all designed by Google. There is no claim by Oracle of any use of their technology there, none.

Application framework. To make these apps work, you need a framework below that. And you can see some of the names here: An activity manager, a window manager. These are all the things you need to make your applications run right. All designed by Google. No claim by Oracle of anything in that

framework.

Then the cool part of the phone, the cool part of the phone is the libraries. Not the Java libraries. Not what they are complaining about. The libraries in Android are what give you all the hot features that people like, and they are all open source products that Google developed with other people.

So take, for example, our graphics. High-end graphics that people like. Play a game, *Angry Birds*. That's an open source project from Open GL. Let's say I want to go on the web. WebKit. That's an open source project originally built by Apple. FaceBook. You can look at Amazon. You can look at Zappo's. When you're going on the web, you're using WebKit. Again, has nothing to do with Oracle.

Let's say you want to watch a video, your grandson's video. That's from Media Frameworks. High-end video. All of that. Open source, created by other people.

Now, below this layer there is another layer, the Linux kernel. Let's call that the basement. Linux is an open source product. Linux was modified by Google to use on Android. It's sort of a plumbing.

Power management. That's your battery, display, camera, your memory in there. That's all the basement. That's not even written in the Java language. That's in a completely different language.

15 million lines of code. That's what we have

reviewed, 15 million lines of code. 57,000 separate files created by Google and its partners.

All right. What's the dispute about in this case? What are we talking about here? Let's go to the runtime and the core libraries and open those up. There are 51 core libraries and there are 37 library packages that Oracle is complaining about.

Now, I'm showing these packages in blue because they are filled in. I'm showing the content inside. There is no complaint about that content. That content was all written by Google. The complaint is about something a lot more, let's call it abstract.

The structure selection and organization. That's what their claim is. That's the heart of their claim, is that you use the same organization of these API names and objects and classes. Those -- that structure and organization is nothing more than a system for organizing Java. It's a system of organization, and that's why it's necessary to use it if you're going to program in Java. That's what we're talking about.

So let's go to the next slide.

What is an API? They didn't explain to you what an API was or how it worked. I'm going to try my darndest to do that because I think it's important to understand it.

The APIs are the words that programmers use when they

are writing in Java. And let's give an example. These are the words that a programmer would use when they are going to call up some function.

This is a developer. Let's say she works for Amazon. She is going to develop a new app for Amazon that will allow us to download all their stuff and sort products by price. We might want to know what the pricing is.

Next slide.

So let's say she wants to create a program. One of the things you would need in that program is something to tell you which is the highest price and which is the lowest. Java has a function called max(). Max() is simply choosing the greater of two numbers. It's a math function. Choosing the greater of two numbers. And, obviously, if I'm going to create this application, I need to sort my prices by number. Is $64 higher than $59? Okay, it is. Let's put it on top. Is $51 less than $59? Yeah, it's less. Let's put it on the bottom. All that is is the idea of choosing the greater of two numbers.

Next slide.

See, what my developer is going to do here on the left, she's got max(), which simply means choose the greater of the two numbers. On the right is the code. That's the original source code that actually does the work. When you say, "I want to choose the greater of two numbers," that source code on the right -- whoops, I can't do that. The source code

on the right below the title there, that tells the computer what to do.

This is the Android platform so all that source code was written by Google. All that source code was written by Google. No dispute about that.

But there are other platforms, like GNU that I mentioned a minute ago. They use the same Java, API, max(), but they wrote their own source code. Different from what Sun did.

Third example. Sun, they use max(). They have different source code to do the same thing.

So these APIs are the names and the organization. The names are free to use, remember. The name max(), that's not protected. The class that it's in, the method, the package all unprotected. So what is it that we're talking about when we say the structure of these APIs?

Mr. Jacobs kept talking about API design. He talked about API design. He said blueprint. They are not blueprints. It's not a book. It's not a novel. All they are is an organization system so my developer can find the right source code to use in her program.

Next slide.

So how do we illustrate this? There's my source code. I want to be able to find it, so I have to organize it. How would I do that? I might put it in a folder called max().

Obviously, this isn't inside the computer. This is an analogy. This is an illustration of how this works.

I might put the max() folder in a drawer called Math, because it's a math function. And then I might put the math function in a package here called Java.lang and it's in that API package.

Now, look at the top there. It says Java.lang.Math.max(). Those are the words the developer writes when she wants the source code in the folder to work. The source code written by Google or, if you're using Sun's code written by Sun or GNU, they are all using the same word, which is free to use, to bring up and call that source code into action.

So now let's look at the next slide.

So now I've got my developer. The system is set. My source code is written. It's in my folder. My developer writes at the top Java.lang.Math.max() because she wants to find a source code.

And by the way, programmers are taught this day one. This is programming one in Java. They all know these APIs. They are taught in college. They are in books. They are published. They are all out there for people to use.

She writes java.lang.Math.max() and, boom, the Java.lang cabinet has the Math drawer. She opens the Math drawer and the max() folder is in it. And then what happens?

The source code comes out of the folder and into her program.

So just by writing the API, Java.lang.Math.Max(), that source code appears and comes into the program.

And now I actually created -- excuse me, your Honor. I'm going to approach the cabinet. I actually created a cabinet to illustrate this because, again, I think it's important for everybody to understand what we're talking about when we say structure and organization of an API.

This is a cabinet. This is the Java language package. It happens to be a file cabinet. There are 37 of these that they are complaining about. They are not complaining about using the language, because that's free. The names were all free. The complaint is about the system of organization. But you need that in order to program in Java.

So if I want to find this max() function. I write java.lang.Math.max() and the system knows I go to the java.lang package. I open the Math drawer.

Now, in the Math drawer are all the methods that are in the math class in Java. And by the way, they are typically organized alphabetically. Nothing too magic about that. They are organized alphabetically. But one of them would be my max() folder. And I take my max() folder out and inside it is the source code. That's the original source code that Google wrote. This is what Google was trying to purchase from Sun. This is what would have made Android happen faster, but that

didn't happen. And all this source code was written original from scratch by Google.

And what we're talking about here is nothing more than this system of organization that has been around for years and programmers had been using whenever they program in Java. That's what is at issue in this case.

Our point is the language is free and you need that system to find the source code that Google wrote.

Now, I want to drive that home with two other slides.

Let's go forward if we can, Ben. Let's go forward to the source code.

All right. Let's pause it... Pause it there.

(Document displayed)

All right. That's just the illustration of what we did. Java.lang is the cabinet. That's the package. Math is the class. Max is the method.

The source code inside Android is very different from the source code inside a Sun or Oracle product.

Here is just two examples. This is on the left, Oracle's version of the methodstring.compareTo. And on the right is Google's version. The first line of each of them is the same because that's the API that calls the source code into play. That's the organizational system.

But the source code is different. The implementation. How you tell the computer what to do. That's

all different. Here's one example.

I've got a second example. And we're talking now about 15 million lines of code.

And I'll pause to comment on one thing that Mr. Jacobs said last night. He did say there was copying. He said, "Not a lot." That tends to be a big overstatement. The copying, the only copying of source code they are complaining about is nine lines of code, nine out of 15 million. Nine -- it's trivial. Inconsequential. They scoured the whole Android system to find evidence of copying and out of 15 million lines they found nine.

We'll explain what happened to those nine. Those nine were written by an engineer named Josh Bloch. They were originally written when he worked at Sun. He came to Google. He came to the Android team late after Android had already been out there. He wrote a new file that he thought was really neat. He originally wrote it for Java, and he gave it to Sun and Sun said this is neat.

The mistake he made was he put the file in Android and those nine lines of code that he wrote back at Sun shouldn't have been in there, because the Android developers knew and the engineers knew you shouldn't -- we're not using Sun's source code. The language is free. The APIs are free. The source code is not. This is trivial.

Can we show this, Ben? Let go back one.

Okay. Nine lines of code. The code file it's in is 924 lines total. So it's trivial right there. But if you compare it to all of Android, it disappears; nine lines in 15 million. And even their expert couldn't assign any value whatsoever to it. None. No value.

Okay. Let's go back to the chronology and we'll walk through the rest. In 2007 -- I'm going to be talking now about what happened after Google developed Android. They published it. It was never a secret. It's open source. So everything we have been talking about, all that source code, the APIs, the Java language, anybody in the world could download it and see what it was, including Sun. And Sun was well aware from earlier discussions that Google was developing the Android platform based on the Java language and they were well aware that Google was using the Java APIs.

So when Google released the Android kit in 2007, Sun did not object. Sun did not say, "You need a license." Sun didn't say, "Oh, you've copied the APIs."

This is what Sun said. Right on the screen. Jonathan Schwartz on a Sun-sponsored website said:

"Congratulations Google." Congratulations.

"I want to add my voice to the chorus of others from Sun." Everybody at Sun.

"Heartfelt congratulations on the announcement of their new Java/Linux phone

platform, Android."

Don't tell me they didn't know exactly what was going on. It's a Java platform. "Congratulations."

He says at the bottom:

"We have obviously done a lot of work -- a ton of work to support developers on all Java based platforms and we're pleased to add Google's Android to the list."

We're going to support Android, too. Just like other Java platforms and other development platforms.

He went on to say:

"Google and the Open Handset Alliance just strapped another set of rockets, another set of rockets to the communities' momentum.

With friends like Google and Red Hat, it sure seems like the momentum behind Java is on the rise."

In other words this wasn't a secret. This was public. This was Jonathan Schwartz telling the world, We welcome Android. We welcome its use of Java. It's a good thing for us.

And this wasn't just what they said publicly. He said it privately, too. Here is an internal email that we got during the course of the case between Mr. Schwartz and Eric Schmidt at Google right around the same time, November 2007:

"Let us know how we can help support your announcement next week."

See the subject line there? "Subject: Android." This is right around the time in '07 when Android is being released for the first time.

So Schwartz from Sun says to Schmidt from Google: "Let us know how we can support your announcement. We're happy to do so." Happy to do so.

Now, you'll hear other testimony from Schwartz and Schmidt. They continued to talk. They continued to discuss business. But Schwartz never said, "You're wrong. You've copied. You've done anything wrong." None of that. None of that.

Now, what happened next? Well, there was more praise for Android in May. Mr. Schwartz gave an interview and then he made some comments at Java One, which is a big development conference.

And in June of '08, not even on the timeline, Sun showed up at a big developer conference and they had built a Java -- a Sun product on top of Android. They were trying to figure out how to build their own products using Android and so they did that. And then in the fall, Google launched the first Android phone. It was called the G.

And then in '08 HTC released a phone. HTC is one of

the handset makers. And as you can see there, HTC released one in '08, in October of '08.

Now what happened next? Enter Oracle. Oracle then acquires Sun. And you'll hear testimony from Mr. Ellison himself about that today. They acquired Sun and they wanted to get in the smart phone business. That was one of the big motivations.

One of the first things Mr. Ellison did after he bought Sun was he appeared at a Java conference in the summer of '09. All the Java developers were there. All the whole Java community, and he said: We are flattered by Android's use of Java. We are flattered by Android's use of Java. I've got his remarks here, and I think we can play them.

Can we play these on the screen, Ben? This is Mr. Ellison.

(Videotape played in open court.)

What he said there was, Sun has opened up Java. Sun has given Java away, just like he admitted in his testimony -- Java language is free -- and we, Oracle, expect to be doing more of the same.

Then he went on and made the following comments about Android in particular.

(Videotape played in open court.)

Can we put that up as a slide, Ben?

(Document displayed)

"We're flattered by Android. We're flattered. Android is very exciting. Everyone should be flattered. I think we can see lots of Java devices. Some coming from our friends at Google."

So this is Oracle now, just like Mr. Schwartz, publicly endorsing Android, publicly endorsing its use of Java.

So, again, there wasn't any complaint at this point. We're back now in April of '09. And Google went forward with Android. And Samsung then released the Galaxy in June of '09. That's another manufacturer of Android products. Motorola -- I'm sorry. This is November '08. Should be '09. November '08 Motorola releases the Droid.

So we have now got a Google version of this. You have a Samsung version of this. You have a Motorola version of this. And these phones are getting out there and becoming popular and sell like wildfire.

So my next point. Beyond approval is that Google made a fair use of Java in Android. What does that mean? Google transformed these APIs into something that no one else is able to do. No one else was able to build a smart phone platform using Java, but a lot of people tried.

Now, this next slide was one that Mr. Jacobs put up last night. It's important. This he represented as the Java community. Look at all those companies. The Java community is

a big community. There is one really important member of that community missing on his slide. Who do you think that would be? Google.

Probably the biggest contributor to the Java community is Google. Google engineers use Java language all over the place. Not just in Android, but a lot of other places. And they contribute what they do to the Java community process. So a lot of the code in Java that's out there and improving all the time is being contributed by people at Google.

So Google knows Java, too, and Google is a part of this community and I was disappointed to see that we were left off that slide.

Next point.

You know that Google transformed Java because the real experts, Sun, they weren't able to build a smart phone like this (indicating). All Sun or Oracle were ever able to do was use Java on a feature phone like this (indicating). Java is very, very prevalent. When you see numbers about how popular Java is on handhelds they are talking about feature phones, not these (indicating). This is what Android did (indicating).

So these three examples I have on the board are three of the examples of efforts at Sun to build a smart phone using Java. They had a project called Acadia. They didn't work.

They had a project called One Java. That didn't work. They had a project called Deneel, actually using some of the Android features. That didn't work.

So Sun was never able to build a smart phone platform using Java. They were the Java experts.

Next slide.

Enter Mr. Ellison at Oracle. And they wanted to get into the smart phone market right away, too. They saw how popular this was.

Slide next.

One of the very first things that Mr. Ellison did on behalf of Oracle was to write to Scott McNealy. You saw him in the video. He was the chairman of Sun. He was Mr. Schwartz's boss at the time. McNealy was the chairman; Schwartz was the CEO.

Mr. Ellison says, Scott, I think the best way, the best way to increase Java revenue is to build a mobile phone application on top of JavaFx which will make Apple's iPhone a direct competitor.

So he sees iPhones out there. They are making money. We can do it too. JavaFx is a Sun Java product. One of the products that Sun makes with the Java language. And so they assembled a group of engineers.

Next slide, please.

This was Project Java Phone. Project Java Phone at

Oracle. They assembled the engineers. They looked at everything. They actually looked at building their phone on top of Android, just like Sun had. No complaint about Android. Nothing. They tried to build it on top of Android.

Next slide.

They weren't able to do it. There were a lot of problems, but this was one. Problem three:

"Very limited internal expertise to make smart decisions at Oracle."

So limited expertise. We can't do it.

Now, you're going to hear from Mr. Ellison that they are experts in Java, too. They use Java all over Oracle products. It's in a lot of Oracle products. But they weren't able to build a smart phone.

So what do they do next?

Let's go back then to the previous slide.

After they failed to build a smart phone on their own, they look to buy one. They talk about buying RIM, which makes Blackberry. They talked about buying Palm, which makes Palm Treo. Those didn't work out. They couldn't do it that way.

Then Mr. Ellison went to Eric Schmidt, the boss at Google, and said, "We would like to partner with you guys on Android. We think you can do better. You guys should buy our Sun virtual machine." It was a Sun product based on the Java

language.

Next slide.

"And put it in Android." And this is what I asked Mr. Ellison during his testimony:

"QUESTION: What was the business proposition?"

This is about his discussion with Mr. Schmidt.

"QUESTION: Did you want Google to replace its virtual machine with a Java machine; is that what you said?

"ANSWER: Yeah. Our JVM would boot faster, run faster" --

"Boot" means start up.

"...run faster, use less power, do all those things, and we would then have a joint project with Google where we could save them money."

That's what he wanted. A joint project with Google where he could save them money. That didn't happen either. That's too late. Google said no, and that didn't happen either.

Now, I want to address one other email that you saw three times, three times during yesterday's opening. An email by Tim Lindholm that says the alternatives to Java suck and we need to negotiate a license. Remember that email? That email

was never part of the dialogue back here. That's got nothing to do with the negotiations back in '05 and '06. That email wasn't written until July of 2010 after Oracle failed to build their own Java Phone, failed to buy a smart phone, failed to partner with Google. They then claimed for the first time: Google you're wrong. We want money based on Android. You've copied our technology.

They didn't even mention the APIs, there was no mention of APIs, none. They said, We have all these patents, and so on and so forth.

So Mr. Lindholm's email was an effort to look and see in 2010 what can we do? Can we change in 2010? By 2010 Android had been launched. The phone was out. All the manufacturers were using Android; Motorola, HTC.

So, obviously, if you've built your house and painted the walls and put the roof on and you've moved in and you're living there, it's awful hard to change the floor plan at that point. But that's what Oracle was demanding. Change your floor plan after years of our telling you this was fine and we support Android. Change your floor plan because we want a piece of the smart phone market. That's what was going on.

So let me have my next slide.

So this is what we expect to prove on behalf of Google. Sun gave the Java language to the public and the APIs are necessary to use the language.

Google then built Android using free and open technologies. They didn't need a license from Sun. Didn't need a license. They weren't using Sun technology.

Sun publicly approved what they did. Sun knew everything about what was in Android and said, You're a rocket. Congratulations. We're going to support you.

And Google made a fair use of the APIs by being the only company to be able to take those APIs and put them into a smart phone platform that works.

And now as Judge Alsup has told you, we will begin the evidence in just a few minutes and the evidence will be presented over the next couple of weeks. And I have a really important request to make; that is, we go second. Oracle goes first. So I have to ask you to keep an open mind because we'll have our opportunity to present evidence when they are done, but we won't have an opportunity to do much until that time.

When I come back after all the evidence is in, I'm going to ask you to make three findings. I'm going to ask for three findings.

Finding one: There is no copyright infringement because the language is free and because the APIs are necessary just to use the language.

Finding two: Sun was aware from day one of what Google was doing and approved it, endorsed it, and it's too late now to make a complete about-face and ask that the whole

floor plan be changed.

And the third finding that we'll be asking for based on the evidence is that Android is a fair use of the Java APIs because it took those APIs and transformed them into the only, only smart phone platform working on Java today.

So I thank you very much for your time and attention and I really look forward to presenting evidence on behalf of Google.

Let me reintroduce one more time our team because you will be hearing from other attorneys on behalf of Google. Christa Anderson. And Dan Purcell. Bruce Baber. And Michael Kwun. We all look forward to representing Google as the case proceeds.

Thank you.

THE COURT: All right. Thank you Mr. Van Nest.

May I ask your team to sort of push back the placard and the file cabinet?

Can you all over there in the jury box go about another half hour? Is that okay so we can get started with the evidence?

(All jurors respond affirmatively.)

THE COURT: Let me just -- you're all brand new to this, and I need to repeat things every now and then. We will have the trial in three parts. The first part concerns copyrights. You're not going to hear about patents in the

first part of the case, or at least not very much about patents.

The second part of the case does involve patents. And you might say: What is the difference between a copyright and a patent? And I will explain that in due course, but not right now. Right now we're dealing with the copyright part.

And then in part three, assuming that you find liability in part one or part two, then we have a part on damages and some other related issues. Damages meaning compensation that the jury might award based upon infringement, assuming infringement is found in the first place.

Now, after each of these parts -- the part we're in right now is going to go two to possibly three weeks, something like that. After each one of these parts you will have a -- I will instruct you on the law. I will give you the guidelines. I will give you a verdict form and then you go back into the jury room and decide that part of the case. And then when that's decided, we come back out here and go to the next part.

Think how much farther up to speed you will be when we get to part two because you will now know all of that background and everything that you've already heard. You will know the name of the lawyers, the companies, the product, the internal workings and so forth.

So, yes, it is a huge burden on you 12 to sit through so many weeks of testimony and evidence, and even a greater

burden to decide those issues, but you can see that it is -- it makes sense to do it, for one jury to do all of that. So that's why we have you over there. You are the ones selected to do this.

All right. At this time we will begin the evidence in the case. The way that works is one side calls a witness. The plaintiff gets to start, and the other side gets to cross examine. There are time limits that I don't need to get into, but I keep track myself and each side will allocate their time accordingly.

So at this time on behalf of Oracle, the plaintiff, you may call your first witness, Mr. Boies.

MR. MR. BOIES: Thank you, your Honor. We call as our first witness my party admission videotaped deposition of Mr. Page. That would be relatively brief and then our next witness will be live.

THE COURT: All right. So let me -- before you play that, I will explain to the jury what a deposition is. You're going to hear this all over the place in this case.

Before a trial, each side has the opportunity to do investigation. And, in fact, it has evolved to the point that the rules allow for what are called depositions. And one side can request -- in fact, require -- someone to sit for a deposition. It's done under oath with a court reporter, just like we have here. And these days they often videotape these

depositions so that the -- you can not only hear, but see as well as get the printed word. It's also allowable to just read to you the deposition, if that's what counsel wishes to do.

But in this case you want to play it, right?

MR. MR. BOIES: Yes, your Honor.

THE COURT: All right. That's fine.

So this is under oath. It's testimony under oath. And it is it is given -- you should give it just as much weight as you want to give it, of course. It's up to you how much weight to give it, but it counts just as much as if the testimony were here live in court.

Now, this deposition is likely to have been just a few parts, right? How long do you think this will be?

MR. MR. BOIES: This part is about 18 or 19 minutes, your Honor.

THE COURT: All right. So probably the deposition was a day or maybe longer, but they have reduced down to selected parts what they would like to present.

And the witness again is who?

MR. MR. BOIES: Mr. Larry Page, the chief executive officer of Google.

THE COURT: All right. And who was asking the questions?

MR. MR. BOIES: I was asking the questions, your Honor.

THE COURT: And the date of the deposition was what?

MR. MR. BOIES: Date of the deposition was August 24th, 2011.

THE COURT: All right. So you now have -- the stage is set and what we will do is play this. We'll just all sit here in silence and watch it. This is the first item of evidence that we will hear in the case.

Please roll the tape.

MR. MR. BOIES: Thank you, your Honor.

WHEREUPON:

LARRY PAGE,

called as a witness for the Plaintiff herein, testified via videotaped deposition played in open court in the presence and hearing of the jury.

(Time noted: 8:50 a.m.)

MR. MR. BOIES: Your Honor, for the record, this is Trial Exhibit 1. When they refer to Deposition Exhibit 517, it's Trial Exhibit 1.

THE COURT: All right. Thank you.

(Playing of the videotaped deposition was resumed in open court.)

MR. MR. BOIES: And this is Trial Exhibit 2, your Honor. The trial exhibit numbers are different from the deposition numbers.

THE COURT: Well, before you -- are there going to be

more examples of that?

MR. MR. BOIES: There are going to be some additional examples.

THE COURT: What you might want to do, so you don't have to stop each time, is just give us the translation right now. The jury will write it down. We all realize now that your numbering is different.

If you don't have that handy... But if it's convenient now, we'll do it.

MR. MR. BOIES: I don't have that handy, but we'll do it for the next deposition.

THE COURT: To explain this to the jury. At the depositions they mark exhibits just like we mark them here in court and it's not always possible to have them use the same numbers as we will here in court.

So they have remarked the exhibits so it will be easier for you, but one way that makes it harder is that we have got a mismatch with the depo numbers and you'll just have to bear with us on that. We will get you the translation as we go along. So that's what's happening here with the exhibit numbers.

Okay. Please continue on.

(Playing of the videotaped deposition was resumed in open court.)

MR. BOIES: And this is Trial Exhibit 7, Your Honor.

Exhibit 490 at deposition. Trial Exhibit 7.

(Video resumes.)

MR. BOIES: Your Honor, plaintiff's deposition Exhibit 496 is Trial Exhibit 401.

(Video resumes.)

THE COURT: Thank you.

(Video concluded.)

MR. BOIES: Your Honor, that completes the portion of the deposition.

THE COURT: Okay. And can we go a few more minutes over there? Everyone seems to indicate yes.

Next witness.

MR. BOIES: Your Honor, the next witness we call is Mr. Larry Ellison.

THE COURT: All right. We will do this. We will go about 15 minutes.

Can you go 15 minutes over there?

(Jurors respond affirmatively.)

THE COURT: We'll get started on the next witness, and then we'll take a break at approximately 9:30.

All right. The next witness will come in.

Welcome. Please come up to the witness stand, Mr. Ellison, and raise your right hand.

LARRY ELLISON,

called as a witness for the Plaintiff herein, having been first duly sworn, was examined and testified as follows:

THE WITNESS: I do.

THE COURT: All right. Thank you. Welcome. Please have a seat.

And you see how close you've got to get? It will move all around at your convenience. It will also move back so you don't have to lean forward.

THE WITNESS: Thank you, Your Honor.

THE COURT: Why don't you say your name to make sure it's coming through.

THE WITNESS: My name is Larry Ellison.

THE COURT: Perfect.

Go ahead, counsel.

DIRECT EXAMINATION

BY MR. BOIES:

Q. Good morning, Mr. Ellison.

A. Good morning.

Q. Let me begin, just very briefly, by asking you to tell the jury a little bit about yourself.

Where were you born?

A. I was born in New York City.

Q. And where did you go to school?

A. Let's see. In grammar school -- I went to school in

Chicago. I went to the Eugene Field Grammar School and South Shore High School. They are public schools in Chicago.

Q. Where did you go to college?

A. Went to the University of Illinois and the University of Chicago.

Q. And did you do any graduate work?

A. I did not.

Q. You founded Oracle, correct?

A. In 1977.

Q. And when you founded Oracle, how many people were part of the company?

A. There were four of us.

Q. And what was your role?

A. I was the CEO and the head of engineering.

Q. And what is your role at the company today?

A. I'm the CEO and head of engineering.

Q. No promotions?

A. Not in 30 years.

Q. And how -- how many employees does Oracle have today?

A. Over 100,000.

Q. Could you briefly describe what Oracle's business is?

A. Oracle is in the business of designing, building, and selling computer hardware and software.

Q. Now, in connection, I want to focus particularly on the software part of the business. With respect to the software

part of the business, is intellectual property important?
A. What we do is create intellectual property. We create hardware designs and we create, in this case, software designs.
And we, again, design computer software, they are computer programs. So we design computer programs. And then we build those computer programs. Both the design of the program and the program itself are both intellectual property.
Q. Now, we're going to talk in this trial both about patent and copyrights, but I just want to focus on copyrights right now.
Does Oracle use copyrights to protect its intellectual property?
A. We use copyrights to protect both our software designs and the programs -- and the computer programs themselves.
Q. Is it expensive to design software programs and develop software programs?
A. Oracle spends about $5 billion a year on research and development. 90 percent of that is spent on creating -- designing and programming and creating computer software.
Q. And would it be possible to make that kind of investment if you did not have copyright protection for the intellectual property, the software that you created?
A. Well, no. If people could copy our software, in other words create cheap knock offs of our products, we wouldn't get paid for our engineering and we wouldn't be able to continue to

invest at the rate we invest.

Q. Let me focus on Java very broadly. And could you explain what Java consists of.

A. Okay. There are two parts to Java. You create -- you write computer programs. And you write them, and you run computer programs.

So there's the Java development environment that you use to write a computer program, and then there's the Java runtime environment that you use to run the Java program.

Q. Let me focus on writing the computer program.

There has been already some mention in the trial of something called the Java programming language, something called API, something called the Java virtual machine.

A. Right.

Q. Could you explain what those terms refer to?

A. The Java -- when you program in Java, when you write a program in Java, you do two very different things. One, you use the Java programming language; and, two, you get to reuse a library of prewritten programs.

And, if you like, I can give you examples of both of those things.

Q. Please.

A. All right. So the Java programming language, an example would be if commission is greater than zero, salary plus commission equals pay. That's an example of a Java program.

The Java compiler, if you will, understands "if," it understands "greater than" and it understands "plus." If commission, greater than zero, salary plus commission equals pay.

The Java -- the prewritten Java program -- so that's one of the things you do, you write in this Java programming language.

Q. Let me stop you just for a second there because you used the word "compiler." Could you explain what a compiler is?

A. The compiler takes the Java programming language and it converts it into a form that the computer can understand very quickly. So it converts it into something called Java bytecode.

So when you run the Java program -- that's the other part, you -- we're talking about writing the program. When you run the Java program, before you run it you put it through the compiler. Compiles the Java program.

And then the Java program runs -- when you're actually running it, it runs on the Java virtual machine and the Java runtime environment.

Q. Now, you had given us just an example of the Java programming language, and you said you were going to also give us an example of a prewritten Java program.

A. Okay. Again, Java programmers do two things. One, they write in the Java language. And then they do something else

that's quite different. They get to reuse these prewritten Java programs. Let me give you an example of a prewritten Java program.

We could have a prewritten program to manage lists of names and addresses. And that program really has two parts. It's APIs and the rest of the program. And I'll explain both of those.

The APIs are a command structure you give to the program. For example, in a program that manages -- when you write a program that manages lists of names and addresses, the command structure or the APIs would be: create new list, add name and address to list, delete name and address from list, sort list.

So these prewritten program -- so what the program does, the program has two parts. The program has the API portion of the program that understands these commands: create list, sort list.

The API portion of the program decodes these commands to figure out what to do, like sort list. And then it actually obeys the command. It actually goes out and sorts the list of names and addresses.

So we have a very large library of these prewritten programs that Java programmers can use as building blocks when they are creating their own Java programs.

So, back to what does a Java programmer do? The Java

programmer writes in the Java language, and then the Java programmer gets to reuse these building blocks, these programs, and include them and build a still larger program.

Q. When you refer to "these building blocks," these prewritten programs, what are you referring to?

A. That's, again, this library of programs that you access through APIs.

Again, the Java -- the Java program environment includes these two parts: The Java language and this library of prewritten programs.

And those prewritten programs, again, are -- the command structure of those prewritten programs are these APIs.

So when you program in Java, you write language statements, and you use the APIs to these prewritten programs.

Q. Now, is it necessary to use the Java APIs that Sun has created in order to use the Java programming language?

A. Absolutely not.

MR. VAN NEST: Objection, Your Honor. Calls for expert testimony.

THE COURT: Do you know the answer to the question?

THE WITNESS: Yes, Your Honor.

THE COURT: From personal knowledge?

THE WITNESS: Yes, Your Honor.

THE COURT: Overruled. Please answer.

THE WITNESS: There's a company in the UK that built

its own Java environment. And they used the Java programming language, but they created their own set of APIs, prewritten programs. And that other environment is called Spring.

So Spring uses the Java programming language, but it doesn't use the Sun-created APIs. They have their own set of APIs and their own set of prewritten programs.

Furthermore, there are lots of programming languages that are just programming languages and don't have any prewritten library or programs for reuse.

BY MR. BOIES:

Q. Now, is it difficult/expensive to create APIs?

A. Uhm, arguably, it's one of the most difficult things we do at Oracle. When you design a program, the very first thing you do is create or define the APIs of the program. That's a task that's done by our most senior experienced and talented software engineers.

Q. Does Sun, and now Oracle, offer licenses to people who want to use the APIs for Java that Sun has created or Oracle has now created?

A. We do have a variety of licenses for Java.

Q. Can you explain what those types of licenses are?

A. Yeah. There are three kinds of licenses. There's the GPL open source license. There's a specification license. And then there's a commercial license.

Q. I want to go through each of those licenses and talk about

what their nature and character is.

First, let me begin with what you referred to as the GPL license. Is that correct?

A. Yes.

Q. You said that was an open source license?

A. Yes.

Q. Can you explain what that means.

A. Well, all of the Java code is published. So it's open -- it's openly published so people can look at it. And that's the notion of open source. Anyone can read the source code.

But just because something is open source doesn't mean you can do whatever you want to do with it. The open source code is governed by a license agreement called GPL. And what GPL says is you are free to take this code, these programs, and do whatever you want, so long as any changes or additions you make to that code you also publish freely and openly under that same GPL license.

Q. Let me see if I understand what you're saying. If somebody accepts a Java GPL license, they get the source code for free. But, in return, they have to make free to anyone who wants to use what they create whatever it is they've created?

A. They -- the answer is yes, but they get more than just the source code. They not only get the source code, they also get a license to the Java patents and a license to the Java copyrights, so long as they adhere to the rules of that GPL

open source license.

Q. Now, was a Java GPL open source license available to Google?

A. Absolutely.

Q. Now, you said a second type of license was a Java specification license. Do you recall that?

A. Yes.

Q. And would you explain what the nature of a Java specification license is.

A. Of course.

The Java specification license lets you look at all of the source code -- excuse me. Excuse me. Let me be clear. The Java specification license -- that is incorrect, what I said. It doesn't let you look at the source code. Let me back up.

The Java specification license lets you look at the Java documentation. Not the source code. All of the Java documentation, something that's in English. It's printed out on a sheet of paper. Let's look at all of those specifications, those design specifications.

And, then, using those design specifications you can build your own version of Java. So you can use our -- you can use the designs and all the specs. You cannot look at the code. Very specifically, you are not allowed to look at the code. And then using those specifications, you can then build

your own version of Java.

Once you have built that version of Java, you must run a -- what's called a compatibility test, to make sure that it is Java, to make sure -- because we want everyone's version of Java -- IBM, by the way, did this. IBM has its own version of Java. Oracle has a version of Java. SAP has a version of Java.

There are lots of companies that have built their own versions of the Java environment; both the environment for writing programs and the environment for running programs.

Lots of people have done this, and they got a specification license. But they must -- part of the specification license requires them to run this compatibility test called the TCK. I think it's Test Compatibility Kit. They have to run this TCK. And Oracle -- before, Sun -- charges for this compatibility test.

You can build your own version of Java using the specifications, but you must pass the compatibility test. When you do pass the compatibility test, you then are granted a license for Java copyrights and Java patents. But not until you pass the compatibility test.

Q. Is the Java specification license itself free?

A. The Java specification license itself is free. What we charge for is the TCK, the compatibility test kit.

Q. In order to use a Java specification license, you must buy

a TCK; is that correct?

A. Yes.

Q. And you said that the TCK was designed to ensure compatibility?

A. To ensure all Javas are Java. You know, that -- that there aren't multiple dialects of Java.

Q. Can you explain why that's important to Oracle.

A. The history of Java is kind of interesting.

It used to be when you wrote a program for a Microsoft Windows computer, that program would not run on a MacIntosh, an Apple MacIntosh computer. So if you wanted something that ran on Windows and something that ran on Apple Mac, you would have to write that program twice. First for Windows, and then for the Apple Mac.

The big idea behind Java was write once, run anywhere. So we wanted to be able to write a program once, and that program would run on Windows, and run on MacIntosh, and run on other computers. The whole notion of write once, run anywhere.

Fundamental to that notion of write once, run anywhere is, Java is Java. That perhaps Apple would have created the runtime environment for MacIntosh, and Microsoft or, say, Sun could have created the runtime for Windows, but the programmer would only have to write the program once. And because of compatibility, the program would run on Windows and

run on Mac.

That was the big value proposition behind Java, write once, run anywhere, which only can occur if there's compatibility.

THE COURT: Would this be a good breaking point, Mr. Boies?

MR. BOIES: It would, Your Honor.

THE COURT: Over there in the jury box, we'll take a 15-minute break. Remember all the admonitions. No talking about the case.

THE CLERK: All rise.

(Jury out at 9:31 a.m.)

THE COURT: All right. Please, be seated. Very briefly, and we'll then take our break.

I have a couple of items. One is the exhibits in the -- that were called out in the deposition that you showed --

MR. BOIES: Yes, Your Honor.

THE COURT: -- I haven't allowed anything in evidence yet. I'm sure they're fine. Are they stipulated in? What is the story there? If they all were party admissions, they'll sail right in. But I need to keep track, one by one, what's in evidence.

MR. BOIES: We have not offered them yet, Your Honor. I would like to offer them.

THE COURT: All right. All of those are received because they were all party admissions. Any objection?

(Trial Exhibits 7 and 401 received in evidence.)

MR. VAN NEST: No, Your Honor.

THE COURT: All right. But what I need is the real trial exhibit number. So when we come back, sometime before the end of today, please give me those numbers so I can keep track of what's in. I do it myself, as well as Dawn. So I need to run an orderly thing.

Here's the other suggestion I have. Are we going to have more exhibits where you lawyers did not follow my guidelines?

You're supposed to use the same deposition number so you won't have this problem. You understand that. But you didn't do it. Okay. We'll live with it. But what we now need is a translation kit. And I have a sheet of paper that does the translation so the jury can -- it will help them follow the evidence.

MR. BOIES: Your Honor, we started that now.

THE COURT: So for each witness it ought to say page deposition, the ones that were, you know, A means K. L means Z. So you need to do that in due course.

And we'll backdate -- not backdate, but retroactively back it up to the start of trial so the jury can have that in case they took notes. All right. But you're busy right now.

You don't have to do that at this moment. We'll take 15 minutes. Thank you.

MR. BOIES: Thank you, Your Honor.

(Recess taken from 9:33 to 9:52 a.m.)

THE COURT: Be seated. Let's go back to work.

Just so everyone will know, the rule against talking to the witness only applies on cross-examination. So we aren't there yet.

Please, bring in the jury.

(Jury enters at 9:52 a.m.)

THE COURT: All right. Be seated, please.

Mr. Boies, please continue.

MR. BOIES: Thank you, Your Honor.

BY MR. BOIES:

Q. Mr. Ellison, before the break you were talking about the importance of compatibility for Java. Do you recall that generally?

A. I do.

Q. Now, the jury has heard and will hear terms about fragmenting Java and forking Java. And could you explain what those terms mean.

A. It means creating incompatibility versions of Java and -- that creates two problems.

One, it fragments the developer community because they have to, if you will, learn how to program in industry

standard Java and then this incompatible version of Java. So you would have to learn how to program two slightly -- you know, somewhat different ways. So it fragments the developer community.

Second, it ends the notion of write once, run anywhere. No longer can a programmer write a program once and expect it to run on all the different computers that run the Java runtime environment.

So it fragments the developer community and it breaks the write once, run anywhere promise.

Q. Now, you mentioned earlier that IBM had its own version of the Java; Oracle has got its own version of Java; SAP, another big software company, has its own version of Java. Are those different versions of Java all compatible?

A. They are all compatible. The ones that you mentioned, IBM, SAP, Oracle, Red Hat, Sun -- we can go on and on -- they are all compatible.

Q. And I think you said the Java specification license required that a person taking that license, in order to get the copyrights and the other intellectual property that they got, had to agree to create a compatible version of Java. Is that correct?

A. Right. They got a specification license. They created their compatible version of Java, and they proved it was compatible by running the compatibility test.

Q. Was a specification license available to Google?
A. Yes.
Q. You said there was a third kind of license, in addition to the GPL license and the specification license. Did you?
A. Yes. There's a commercial license.
Q. And can you explain what a commercial license is.
A. A commercial license lets you use the actual -- Oracle's code. It lets you use our Java environment, the actual code itself.
Q. And you said that a GPL license was free. You said a specification license was free, but you had to then buy a TCK. Is a commercial license free?
A. No. We charge for a commercial license.
Q. And do companies take commercial licenses?
A. Yes, they do.
Q. Can you give me some examples of companies that have taken commercial licenses?
A. RIM, that makes the Blackberry. Amazon that makes the Kindle.

There are -- there are lots and lots of examples of companies that take commercial licenses. Nokia, the phone company, takes a commercial license. LG and Samsung take commercial licenses. There are lots and lots of examples of companies that take commercial licenses.

Q. Was a commercial license available to Google?

A. Yes.

Q. Are you aware of any company today that is using Java APIs created by Oracle or Sun that has not taken one of these three licenses?

A. The only company I know that hasn't taken any of these licenses is Google.

Q. Let me go to another subject the jury will hear about, and that is something called a clean room.

Are you familiar with what that means in the context of the software industry? Not in the case of cleaning services.

A. Yes.

Q. What is -- what is a clean room?

A. Well, if you take a Java specification license and you write your own version of Java, you have to do it in a clean room.

What that means is, you're not allowed to look at the Oracle version of Java, the source code. You're not allowed to look at our computer programs when you build your system. That's called a clean room.

In other words, it's cleaned of our intellectual property. The source code itself is not available for you to study, look at, copy, learn from, anything. You just can't look at that. Or you can't have people who have looked at that. You have to have -- they have to -- it has to be, if you

will, independent. They have to do the work independent of what's included in our source code.

Q. And is the concept of a clean room something that is well understood in the industry?

A. Yes. Everyone who takes a specification license understands that they have to build their version of Java in a clean room.

Q. There is a term called the "JCP," I think it is, Java Community Process?

A. Yes.

Q. And can you explain what that is.

A. Okay. Oracle doesn't develop Java by itself. Java is developed by a community that makes contributions, a variety of contributions, to Java.

So it's not just Sun and Oracle. IBM makes contributions to Java. A company called Red Hat makes contributions to Java. SAP makes contributions to Java. There are lots of individual programmers who make contributions to Java.

It's the notion of this open source community, this community of programmers and companies and individuals who, as a group, make contributions and develop Java. It's a big group. It's a big group. So what they do is, they elect an executive committee that kind of governs and decides the future of Java.

Q. Can you give me some examples of companies that are on the Java executive committee.

A. IBM -- Oracle, IBM, SAP, HP, Red Hat. All of whom are our competitors, by the way. We compete, but we still cooperate around Java. And Google is also on the executive committee for Java.

Q. And from time to time are new versions of Java created through this community process?

A. Yeah. The people ask for improvements for Java, so they basically -- they'll come up and they'll say, we'd like to add this feature to Java, we'd like to add that feature to Java.

And these recommendations then go before the executive committee. The executive committee votes on them. Eventually, we have a collection of improvements, and we come up with a new version of Java.

And Java 7, recently the executive committee voted on and approved Java 7.

Q. And is Java 7 a new version of Java?

A. Java 7 is the new version of Java.

Q. And when was that approved by the executive committee?

A. Several months ago.

Q. And when this Java 7 was up for approval, did all of the members of the executive committee have an opportunity to vote?

A. Yes. Everyone -- everyone on the executive committee had an opportunity to vote.

Q. And did anybody vote against this new version?
A. Everyone voted for it except for one company. The only company that voted against Java 7 was Google.
Q. With respect to the APIs that you've mentioned earlier, is using the APIs that Sun and Oracle created an advantage to a company that wants to program in the Java programming language?
A. We -- we think these -- this library of prewritten programs -- and they use those prewritten programs through their APIs.

We think this library of prewritten programs, it's a good library of programs. We think it makes programmers much more productive if they use the library, they use our library.
Q. You mentioned there was this company, I think you said Spring, who had written their own APIs?
A. Yes.
Q. Does it take a period of time and expense and resources if you're going to go that route?
A. Yeah. Spring had to design their own APIs, and then they had to teach the developer community about these new APIs. And they had to persuade them that their collection of APIs, their library of programs, was in some ways better than the library of programs that Oracle and Sun had produced.
Q. And there came a time when you became aware that Google was using Java APIs that Sun had copyrighted and Oracle now owned the copyrights to; is that correct?

A. That's correct.

Q. And did you take any action to try to address that?

A. I met with Eric Schmidt when he was CEO of Google, and I met with Larry Page, the current CEO of Google. And I tried to persuade them to build -- to be compatible with the industry standard version of Java.

Q. And did they agree to do that?

A. No.

MR. BOIES: Your Honor, we have no more questions at this time.

THE COURT: Before we -- thank you.

Before we go to the cross, someone out there has a very loud and noisy keypad. Who is that? I'm going to ask you to stop typing because it is distracting. And if you don't, the marshals will remove you.

I said before, when the lawyer has the floor there will be no distractions. And I mean that.

This is an important case to these parties, and the jury is going to hear every word without a bunch of tick, tick, tick, tick.

I don't know who it was, but I know the direction it was coming from. I don't think it was your table, Mr. Van Nest. I'm not accusing you. It's somebody out there in the public seating.

I'm sorry to be so strong about this, but this is

important to the parties in this case. I want every word of the witness's testimony to be heard and every word of the cross-examination and what Mr. Boies was saying to be heard.

Where is my marshal, my court security officer? Okay. If you hear any loud -- loud typing by one of these reporters or anyone else, tell them to stop. If they don't stop, remove them from the courtroom. All right.

CSO: Yes, Your Honor.

THE COURT: All right. Thank you.

Mr. Van Nest, the floor is yours.

MR. VAN NEST: Thank you, Your Honor.

CROSS EXAMINATION

BY MR. VAN NEST:

Q. Good morning, Mr. Ellison.

A. Good morning.

Q. The Java programming language, you understand that nobody owns that, right?

A. Uhm, we're making no claims to the Java programming language itself.

THE COURT: Mr. Ellison, you can say yes or no. That's a yes or no question. Come on.

THE WITNESS: Could you repeat the question?

BY MR. VAN NEST:

Q. Yes, sir.

You understand nobody owns the Java programming

language, right?
A. I'm not -- I'm not sure.
Q. Anybody can use the Java programming language without any royalty at all, right?
A. Again, I'm not sure.
MR. VAN NEST: Your Honor, at this time I would like to play as a party admission from Mr. Ellison's deposition, at page 47, lines 5 through 10.
THE COURT: Proceed. Proceed.
(Videotaped deposition was played in open court, and was not reported by the court reporter.)
MR. VAN NEST: We need to replay that. The sound was off.
THE COURT: We'll replay it as soon as we get hooked up online.
(Videotaped deposition was played in open court, and was not reported by the court reporter.)
BY MR. VAN NEST:
Q. Now, in this case the Java programming language was originally built by Sun, not by Oracle, right?
A. That's right.
Q. It's been around a long time?
A. Yes.
Q. And it was Sun that created the copyrights that are at issue in this case, not Oracle, right?

A. I believe so.
Q. The APIs that are at issue, those were created by Sun, not Oracle?
A. Yes.
Q. Now, you distinguished between the APIs and the API libraries during your testimony. Did I get that right?
A. I distinguished between the API designs and then the library of programs that implement the APIs.
Q. So the library of programs, that has in it the source code that actually tells the computer what to do when the program comes to that point, right?
A. That's correct.
Q. And that source code is source code that Sun created in the case of these libraries and put in its Java libraries, right?
A. That's correct.
Q. The APIs, those are a command structure, I think you said, that instructs the computer what code to run, right?
A. No. The APIs -- there's an API design, and then there is the actual code that implements the APIs that are a part of the program.
Q. Didn't you testify on direct examination, Mr. Ellison, that the API is a command structure that allows the program to be accessed?
A. I was using the expression "command" in layman's terms,

trying to explain in general what -- what a program does.
Q. So command structure in layman's terms that's the way in which you get access to the source code libraries, right?
A. It's the way you ask the program to do something.
Q. It's different from the source code in the library itself, right?
A. No. The program under -- repeat the question. I'm not sure I understood the question.
Q. The command structure is different than the source code that's in the libraries?
A. No. The command structure is implemented in the source code. So the command structure is a part of the source code. The program has to understand the command first, if you will, understands the API, the direction like sort list. And then it has to know how to sort the list.
Q. Now, I take it that you did not participate in the discussions between Google and Sun that occurred in the 2005-2006 time frame; is that right?
A. That's right.
Q. That was done by folks at Sun, including Mr. Schwartz, who was the CEO?
A. I believe so.
Q. Okay. And to the extent licensing was discussed in that 2005-2006 time frame, you weren't around for any of that?
A. Correct.

Q. And do you actually know when the GPL license you talked about came into existence?
A. What year GPL came into existence?
Q. Yes.
A. I don't know what year GPL came into existence.
Q. You don't know whether it was even in existence in 2005 and 2006, when Google and Sun were having discussions. Do you, Mr. Ellison?
A. I just testified, I don't know what year the GPL came into existence.
Q. Now, as far as you know, Google has never branded its smart phone platform as Java; it's always been called Android?
A. Correct.
Q. So Google is not promoting its smart phone platform under the name Java. It uses a different name, Android?
A. Correct.
Q. And do you know what, if any, position the senior management at Sun took about that back in 2006, 2007, and 2008, before you acquired Sun?
A. I don't.
Q. Was Mr. Schwartz running Sun during that period of time?
A. He was.
Q. He was the chief executive officer?
A. Correct.
Q. Responsible for all decision-making?

A. Yes.
Q. Setting corporate policy?
A. Yes.
Q. Ultimately, responsible for any negotiations with companies like Google and anyone else, right?
A. Yes.
Q. You would say that if the CEO says it, his word goes?
A. Well, no.
Q. Now, you mentioned discussions that you'd had with Mr. Schmidt and Mr. Page. Those occurred in early 2010, right?
A. Yes.
Q. Do you recall approximately when that took place?
A. My most recent discussion with Larry Page was a week ago.
Q. Let's -- let's go back to Mr. Schmidt. That occurred --

MR. VAN NEST: Actually, may I put my timeline up, Your Honor?

THE COURT: Is this your timeline or the one we gave to the jury?

MR. VAN NEST: Well, it's my timeline, but I can put either one up.

THE COURT: I think it's better to use the one we gave to the jury.

MR. VAN NEST: Okay.

THE COURT: Yours is an argument piece, and that's fine, but it would be better for consistency to use the one we

gave the jury.

MR. VAN NEST: That's fine. We'll do that.

BY MR. VAN NEST:

Q. So do you know, in fact, Mr. Ellison, that Android was released originally back in 2007? Is that something you know?

A. Approximately. I didn't know if it was 2007 or early 2008.

Q. And in 2008, the first Android phone came out --

A. That's what -- that's what was in my head, was 2008 was the first Android phone.

Q. And then you entered into an agreement -- it's not on this timeline, but you entered into an agreement to acquire Sun in about April of 2009, correct?

A. Correct.

Q. And you then met with Mr. -- the deal closed sometime in early 2010?

A. Yes.

Q. And your meeting with Mr. Schmidt was in approximately March of 2010?

A. Yes.

Q. Now, by that point in time had you already appeared at JavaOne and said you were flattered by Android's use of Java?

A. Yeah, I said I was very excited that there was a Java phone on the marketplace, yes.

Q. And you said that you expected more Java-based products

from your friends at Google, right?

A. I don't recall saying that, but ...

Q. Do you recall making the statement that you were flattered by Android?

A. That I was flattered that there was a Java phone on the market, yes.

Q. And that Java phone you're talking about was Android --

A. Yes.

Q. -- right? You said it was exciting?

A. Yes.

Q. It was a Java phone, right?

A. Yes, yes.

Q. And you expected to see more Java devices coming from your friends at Google, correct?

A. I mean, I'll take -- I don't recall that part, but okay.

Q. Do you remember actually going up on stage with Mr. McNealy, prior to this meeting with Mr. Schmidt, and making these comments in public?

A. I do.

Q. Do you remember your remarks that day?

A. Yes.

MR. VAN NEST: What I'd like to do, Your Honor, is play an excerpt of Mr. Ellison's remarks, and ask him to identify them and authenticate them.

THE COURT: All right. I will let you do this.

Let's do this the technically right way. What is the Exhibit number?

MR. VAN NEST: It's TX 2939.

THE COURT: Any objection to that being received in evidence?

MR. BOIES: No, Your Honor.

THE COURT: Give me the exhibit number again.

MR. VAN NEST: TX 2939.

THE COURT: 2939.

Over there in the jury box, please, I know your heart is sinking when you hear there are 3,000 exhibits. We won't go through that many.

The lawyers will -- are excellent. They will winnow this down to probably a couple hundred exhibits. But they have numbered a lot more than that.

2939 received in evidence. You may play it for the jury and the witness.

(Trial Exhibit 2939 received in evidence.)

(Video recording played in open court.)

BY MR. VAN NEST:

Q. Mr. Ellison, is that you on the video?

A. Yes, it is.

Q. Now, were your remarks that day recorded and filed with the government?

A. I have no idea.

MR. VAN NEST: Your Honor, may I approach the witness?

THE COURT: Yes. What are you showing him? What exhibit number is that?

MR. VAN NEST: 2041, Your Honor.

THE COURT: 2041.

BY MR. VAN NEST:

Q. Mr. Ellison, would you turn to the -- it's about the sixth page in to 2041. At the top there's a quote from Mr. McNealy:

"Great. With transition there's always nervousness."

A. What page did you say this was on?

Q. Unfortunately, the pages aren't numbered. The Bates number at the bottom is 1732.

A. Okay. Give me a second. I got it. Okay.

Q. All right. Do you see the remarks there that are attributed to you in the middle of the page?

A. I do.

Q. And do those appear to be the remarks that we just heard about on the video?

A. Can I have a second to look?

Q. Sure.

A. They are.

Q. All right.

A. They are the same remarks.

Q. Is it often the case that sometimes when you make public statements, because of the securities rules those statements have to be filed with the government?

A. Yes.

Q. That happens from time to time?

A. It does.

Q. Does this appear to be one of those occasions?

A. It is.

MR. VAN NEST: I would offer 2041 in evidence, Your Honor.

MR. BOIES: No objection, Your Honor.

THE COURT: Received in evidence.

(Trial Exhibit 2041 received in evidence.)

MR. VAN NEST: Could we put up on the screen, I believe it's about page 7, please.

(Document displayed.)

BY MR. VAN NEST:

Q. And there in the middle of the page, are those the remarks that you made at JavaOne in 2008 -- excuse me, 2009?

A. They are.

Q. And you said:

"James, Sun, has done a fantastic job opening Java, opening up Java, giving Java to the world. And we're going to do more of the same."

Do you see those remarks?

A. Yes.

Q. Those are remarks you made that day?

A. Yes.

MR. VAN NEST: Could we go to the top -- to the bottom of the page, please. Highlight that.

BY MR. VAN NEST:

Q. Now, this is what I asked you about earlier.

Was this a reference to being flattered by the fact that Android was using Java?

A. Yes.

Q. You're talking about an Android phone being shaken. You meant out in the audience?

A. Correct.

Q. And you indicated you were very excited by Android?

A. Yes.

Q. And you liked the fact that it used Java?

A. Yes.

Q. And that you expected Google to do more?

A. Yes.

Q. And to expand its use of Java in Android into netbooks and other products, right?

A. Yes.

Q. And that was made at a developer conference with thousands of people in the audience?

A. Yes.
Q. The press were there --
A. Yes.
Q. -- right? Other companies were there?
A. Yes, definitely.
Q. Java developers were there?
A. Lots of them.
Q. Now, when you met with Mr. Schmidt, that was several months after JavaOne in 2009. It was in 2010, correct?
A. Yes.
Q. And, in the meantime, hadn't you made an effort at Oracle to build your own Java phone?
A. No.
Q. Hadn't you told Mr. McNealy, as soon as Oracle had agreed to acquire Sun, that the best way to increase Java revenue would be to build a Java phone?
A. It was an idea that perhaps a good way or the best way would be to build a Java phone. I had the idea for building a Java phone. And we explored that idea, decided it would be a bad idea.
Q. And then that idea was one that you expressed very soon after you agreed to acquire Sun, right to the top, to Mr. McNealy himself, right?
A. Yes.
Q. Did you send him an e-mail?

A. Yes, I did.

Q. Let me back up a minute. When you acquired Sun, wasn't one of the key motivating factors to get into the smart phone market?

A. No.

Q. Was one of the factors driving your interest in Sun the chance to expand Java into smart phones?

A. No.

MR. VAN NEST: Your Honor, I would like to play as a party admission from Mr. Ellison's depo at page 15, line 20 through page 16, line 01.

THE COURT: All right. Please do so.

(Videotaped deposition was played in open court, and was not reported by the court reporter.)

THE WITNESS: I believe you said one of the primary factors. I believe that was your wording.

BY MR. VAN NEST:

Q. So it was a factor?

A. Yes.

Q. And it was a factor that you thought was a good idea?

A. I thought it was an idea worth exploring, and we explored it. But it was not one of the primary motives for buying Sun, which is what you said.

Q. I take it you were hoping to compete not only with Apple but with Google, too, right?

A. I had an idea that we could enter the smart phone business and compete with everybody in the smart phone business, yes.
Q. So that included Apple?
A. Yes.
Q. And it included Google?
A. Yes.
Q. And that's because, at that time, the primary use of Java in phones was in these feature phones that I'm holding in my hand, right?
A. Primary use? Java, at that time, was certainly used in a lot of feature phones, yes. The biggest use of Java, I'm sorry, was in server computers.
Q. Sure. Big computers and --
A. And that's why we bought Sun --
Q. But in the mobile phone the main use, at the time you acquired the company, was in phones like this feature phone --
A. Correct.
Q. -- right?

And your idea was, could we expand the use of Java in smart phones and compete with Apple and Google, right?
A. I wanted to explore it. We explored it, and it turned out to be a bad idea.
Q. Now, you -- one of the very first suggestions you made to Mr. McNealy at Sun was to explore this idea, correct?
A. Yes.

Q. I'd like to hand you Exhibit 2042. Could you identify that, Mr. Ellison?

A. Give me a second to read it. It's an e-mail -- okay, yeah, I've read it.

Q. It's an e-mail that you wrote to Mr. McNealy?

A. Correct.

Q. Mr. McNealy was the chairman of the board of Sun?

A. Yes.

Q. Was he a friend of yours?

A. Yes.

Q. You guys go back several years?

A. Yes.

MR. VAN NEST: And I'd offer 2042 in evidence, Your Honor.

MR. BOIES: No objection.

THE COURT: Thank you. Received.

(Trial Exhibit [redacted] received in evidence.)

MR. VAN NEST: Could we put it on the screen, please.

THE COURT: Publish to the jury.

(Document displayed.)

BY MR. VAN NEST:

Q. Let's focus on the very first sentence.

"Scott, I think the best way to increase Java revenue is to build a mobile phone application suite on top of JavaFX, which

will make Apple's iPhone a direct competitor of ours."

Is that the sentence you wrote?

A. Yes, it is.

Q. This is in May of 2009, so about a month after you agreed to acquire Sun?

A. That's right.

Q. So early on in your period you didn't even formally own Sun yet but you agreed to acquire the company --

A. Correct.

Q. -- right?

And your idea was you would then license the Java phone software to people like Verizon and charge money for it, right?

A. Yeah. The idea was, you know, we were exploring building the smart phone using JavaFX, and then we would charge carriers like Verizon for it, yes.

Q. Right. So this was not going to be open source. You were going to charge the carriers for this product?

A. No. Open source and charging have nothing to do with one another. It was going to be open sourced.

Q. But your idea was charge the carriers; that's how you would make money?

A. Yeah. As opposed to charge for advertising, yes.

Q. Now, didn't you learn early on in this period that Sun had

tried and failed to build its own smart phone?

A. I'm not sure how -- how much Sun tried. I know they had the idea.

Yes, I think James Gosling wrote me an e-mail saying that Sun had tried, and was unable to build a smart phone. I don't know if he actually even started the project.

I know they -- again, it was an idea they explored, and they didn't pursue it. My understanding is they didn't really pursue it.

Q. And James Gosling, he's the father of Java, right?

A. Yes, he is.

Q. He was one of the senior engineers at Sun?

A. Correct.

Q. And he told you right away, right after you acquired the company, hey, we tried it and it never happened, right?

A. Well, no. Let me be clear. I think what he said was, We had the same idea. We explored the same idea and we didn't go through with it.

Q. All right. Let me --

MR. VAN NEST: May I approach the witness, Your Honor?

THE COURT: Yes.

BY MR. VAN NEST:

Q. I have Exhibit 2043, Mr. Ellison. Would you take a look at that, please.

Is that an e-mail that you received from Mr. Gosling in May of 2009?

A. Yes.

MR. VAN NEST: I'd offer 2043 in evidence, Your Honor.

MR. BOIES: No objection, Your Honor.

THE COURT: Thank you. Received.

(Trial Exhibit 2043 received in evidence.)

(Document displayed.)

BY MR. VAN NEST:

Q. This is an e-mail that was part of the same e-mail string with Mr. McNealy, that we just looked at, right?

A. Yes.

Q. And James Gosling wrote you and Mr. McNealy on May 26.

MR. VAN NEST: Could we pan down to the bottom of the page, please, Ben. So right there.

BY MR. VAN NEST:

Q. (As read:)

"Scott, the best way to increase Java revenue is to build a mobile phone."

That's the e-mail we just looked at a minute ago?

A. Yes.

Q. That's your e-mail to Mr. McNealy?

A. Yes.

Q. And he probably, for the record, forwarded that to

Mr. Gosling?

A. Right.

Q. And Mr. Gosling sent an e-mail back to both of you?

A. Yes.

Q. So let's go back to the top of the page, now we have the context. Mr. Gosling said:

"Oddly enough, this has been our wishful thinking strategy of record for years. We could never fund it internally so got nowhere."

Right?

A. I think that's what I said. They never started the project. They had the same idea we had, but they never started the project.

Q. Now, he mentions here that they actually purchased the assets of an outside company called Savage, to try to use those to build a stack, right?

A. It says, We could never fund it internally. We got nowhere. So, I mean, that's the part -- that's what I read right at the top of this e-mail message, that we never funded it. It was a project that never got funded. That's my recollection.

Q. Let me take you a little further down the page, Mr. Ellison. It says:

"There was an outside company, Savage, that

did essentially the perfect thing almost five years ago, but back then they were far too expensive to acquire."

He then says, a sentence later:

"We bought their assets a couple of years ago in a 'fire sale' and doing a more complete handset stack became our real strategy of record a couple of years ago."

Do you see that?

A. And it said he couldn't fund it.

Q. Do you remember him telling you that they actually went out and bought assets at Sun --

A. Yes.

Q. -- to try to build a smart phone?

A. Yes. They bought the assets, but they never put people on the project. They never funded the project, is what this says to me.

MR. VAN NEST: Could we highlight the last sentence of the e-mail. That's it.

BY MR. VAN NEST:

Q. He said:

"We had a pretty big handset applications team, but they appeared in the last RIF."

THE COURT: No, "disappeared."

MR. VAN NEST: I'm sorry. "... they disappeared in

the last RIF."

Thank you, Your Honor.

BY MR. VAN NEST:

Q. Now, handset applications team, you understood that to mean a team to build handset applications in the smart phone market, right?

A. Yes.

Q. None of these efforts came to fruition, in that Sun was never able to build its own Java smart phone, as far as you know, Mr. Ellison, right?

A. James Gosling told me Sun never funded the project. That's what he told me. They never provided adequate funding for the project, yes, correct.

Q. Now, you assembled a team at Oracle to look at building your own Java phone, right?

A. Yeah, to examine -- when we have an idea, a proposal to build a new product, we put a team of people together that, you know, kind of analyzes what it would cost, how long it would take, what are the market dynamics.

So we did a detailed analysis of the idea, what it would take to build a smart phone. That's correct.

Q. And that involved getting a bunch of engineers to do some work?

A. Yes.

Q. And run some simulations?

A. Well, I wouldn't say simulations. Maybe build prototypes, yes.

Q. One of the things they looked at as a starting point was Android, right?

A. Yes.

Q. They actually looked at the Android system to see what parts of that you at Oracle could use to build your own smart phone, right?

A. That's correct.

Q. And they ran some tests on Android as part of that?

A. Yes.

Q. And they examined this platform that Android entailed, right?

A. Yeah. Android included a lot of open source code like the Linux operating system. And we looked at all of the pieces of Android. We did a detailed study of Android. That's one of the things we did when we looked at our own smart phone design.

Q. This is all in 2009, before you met with Mr. Schmidt?

A. That's correct.

Q. And one of the things you looked at was using the Android developers, the Android handset makers, to build a phone for you, right?

A. Well, yeah. The Android handset makers are people like Samsung. So they build handsets -- they build lots of handsets. Of course, there's a limited number of handset

manufacturers. So we looked at the same handset manufacturers that build for Android. LG, Samsung, Motorola, Nokia, all those people.

Q. But you expressly asked your engineers to find out who was building the handsets for Android so you could talk with them in particular, right?

A. We did a detailed study of Android, yes.

Q. Let's -- I'd like to show you trial Exhibit 2044, and ask you if you recognize it?

A. Could I just have a second to look at it?

Q. Certainly.

A. Okay.

Q. Is this a report on Project Java Phone at Oracle?

A. This is the analysis of what -- what the engineers thought it would take for -- you know, if we wanted to enter the handset business. Wanted to enter the software business portion of the smart phone business.

MR. VAN NEST: And I would offer in evidence, Your Honor, Exhibit TX 2044.

MR. BOIES: No objection, Your Honor.

THE COURT: Thank you. Received in evidence.

(Trial Exhibit 2044 received in evidence.)

(Document displayed.)

MR. VAN NEST: Could we display the cover page, two pages in, please. There we go.

BY MR. VAN NEST:

Q. Does this appear to be the opening --

MR. VAN NEST: You had it.

Q. -- the opening page there, the cover, Project Java Phone?

A. Yes.

Q. Strategy and Product Definition?

A. Yes.

Q. This is a 27/30-page report, something like that.

A. Yes, 27.

Q. Appears to include quite a bit of work involved in getting this done?

A. Yes.

Q. This took months, weeks? Do you know?

A. I think weeks, not months.

Q. Okay. Let's look at the first page, the agenda.

Is this the agenda that you were pursuing with Project Java Phone, Mr. Ellison?

A. No. This is kind of the -- what's called -- these are kind of the issues associated with, yeah, smart phone. Not quite an agenda.

Q. Fair enough.

A. Yes. Okay. Fair enough.

Q. And there's a Java phone update with some technical options and recommendation, right?

A. Yes.

Q. And then there's a proposed M&A associated with the project. What does that mean, M&A?

A. We might not have the expertise in-house or assets in-house to build a -- the smart phone software, so we might have to buy some companies. M&A stands for mergers and acquisitions.

Q. The idea would be buy some companies and get related technology from them?

A. If we didn't have the capability in-house to build the Java smart phone software, we would probably have to either hire a bunch of people from outside or, potentially, buy some companies that had that expertise.

Q. And have you heard the expression "buy versus build"?

A. Sure.

Q. And that's a common expression in terms of launching a new product or developing a new product?

A. It's a common expression in terms of developing a new product.

Q. So one option, you can build it yourself --

A. Right.

Q. -- right?

A. Right.

Q. Or you can buy it from someone else?

A. Or a combination of the two. Buy part, build part.

Q. Now, in connection with this smart phone idea of yours,

you also considered buying technology from companies like RIM and Palm, right?

A. I think we might have looked at -- again, we look at everything. I mean, and considered -- I'm not sure if we seriously considered buying RIM. At the time they were very, very expensive. Palm, in contrast, was much less expensive.

But, yes, we look at everything. We look at everything, sure.

Q. And you looked at -- one option was buying Palm, right? You looked at that option?

A. We looked -- yeah. I say we look at everything, yes.

Q. Including looking at Palm?

A. Yes.

Q. But you concluded that Palm was not competitive, right?

A. Correct.

Q. You also looked at RIM, and concluded RIM was too expensive?

A. Yes.

THE COURT: Please, remind the jury what RIM is.

MR. VAN NEST: I'm sorry. RIM makes Blackberry. Research in Motion, RIM, makes Blackberry.

THE COURT: Is that right, Mr. Ellison?

THE WITNESS: Yes, it is, Your Honor.

THE COURT: Thank you. Go ahead.

BY MR. VAN NEST:

Q. Now, could we go down a couple of pages, to page 6, please.

And was there a discussion about handsets and hardware as part of the Java phone evaluation?

A. Give me a second to look at page 6.

Okay. Go ahead.

Q. And under the recommendations for version 1.0 --

MR. VAN NEST: Could we highlight that, note 1. Maybe make it a little bigger.

BY MR. VAN NEST:

Q. That says:

"Target 3 Android handset manufacturer for parallel negotiations but finalize contract with only one handset maker."

Do you see that?

A. I do.

Q. That's a reference to identifying who's making the Android handsets and going and talking to them, right?

A. That's correct.

Q. The idea there was to save time?

A. The idea there was to -- let's say Samsung was already making this hardware that ran the Android software. It would be very convenient for -- if our software ran on the same exact hardware.

Q. And that would save you time in terms of getting to market?

A. It would save us time in terms of getting to market. It would make it more attractive to Samsung because they were already making the hardware.

Q. And it would probably make it cheaper, too, because they already had this things developed, right?

A. Correct. Saves time and money.

Q. Now, you also targeted using some of the Android software as part of this project, too, right?

A. I think we looked at using -- well, again, Android software. Part of the Android software is Java. Part of the Android software is Linux. So I'm not sure what you mean.

Linux is open source, available the GPL license. So -- but we -- but we looked at all the pieces -- the answer is, yes, we looked at every piece of Android. I just want to be specific what Android was made up of.

Q. But you looked at every single piece of Android as part of this project?

A. We looked at everything.

Q. Let's go down to page 11. And can we highlight -- take a moment there to take a look, Mr. Ellison. I'm going to be asking you about the recommendation for version 1.0. It's at the middle of the page, note 1.

A. Okay.

Q. And that says, "Target single operating system only - Android Linux," right?

A. Correct.

Q. There are lots of operating systems?

A. There are lots of operating systems. Yes. Yes, the answer is yes, there are lots of operating systems.

Q. But the one the engineers at Oracle said you should target was Android's Linux, right?

A. Well, not clear there's such a thing as the Android Linux. I mean, there's a Linux, but that's --

Q. Excuse me, Mr. Ellison. My question was, your engineers said that the single operating system to target was Android Linux. Isn't that what this says?

A. That's what that says, yes, but there is no such thing as Android Linux.

Q. Now, you also knew that you could take Android Linux for free and distribute it under the Apache license, right?

A. We knew -- Linux is open source, generally available for license. So Linux is something we use. I mean, Linux is something that Android used in their phone and Linux was something we were free to use in our phone.

That's why I'm objecting to the notion of "Android Linux." It's the Linux that's in the Android phone, is a more precise description of what we were going to use.

Q. All right. But the engineering description in your

engineering report says "Android Linux," right?

A. Yes. I'm just trying to be very precise. I don't think this is very precise.

THE COURT: I just think there is possibly something that the jury has lost track of and that is this: The time frame.

I'm going to ask you to ask questions, not answer them in front of the jury. Just ask questions. But clear up: Are we talking here about Android in the possession of Google or Android in the possession of some other company not yet acquired by Google?

MR. VAN NEST: This is Android Google.

BY MR. VAN NEST:

Q. Mr. Ellison, this is in 2009 that these discussions are occurring?

A. That's correct.

Q. And by 2009 Android was already launched by Google?

A. Yes.

Q. It was out there on handsets being manufactured by Motorola and others?

A. That's correct.

Q. And Samsung?

A. Correct.

Q. And it was out there as a public open source project?

A. Yes.

Q. Okay.

THE COURT: Okay. Thank you. That clears that up.

MR. VAN NEST: Thank you.

THE COURT: Continue on.

BY MR. VAN NEST:

Q. Then if we could open up the bottom bullet in this recommendation, please, the very bottom bullet there. "Licensed under Apache Public License."

Your engineers noted for you that because Android was available under a public license from Apache, you, Oracle, could take advantage and use it, too? That's what that means?

A. Yes.

Q. Now, in the course of this report, you identified some next steps for development, right?

MR. VAN NEST: Can we have Page 12?

(Document displayed)

MR. VAN NEST: Could we highlight the next steps for development?

A. I see it. Yes, of course.

BY MR. VAN NEST:

Q. So one of the next steps was to, quote:

"Run Java ME..."

That's Java Micro Edition, correct?

A. Right.

Q. (Continuing)

"...on Android and evaluate performance and resource power usage"?

A. Correct.

Q. So one of the steps was to actually run a Sun Java product on top of Android and evaluate how that worked?

A. Correct.

Q. And the report notes.

"Work is currently underway. Will report status by August 15."

Right?

A. Correct.

Q. Now, this report ultimately generated a series of problems and issues in developing your own Java smart phone, right?

A. I'm not -- I'm not sure what you mean "problems and issues." But, yes, there are always problems and issues, sure.

Q. And one problem that identified clearly was you didn't have internal expertise at Oracle for making smart decisions?

A. Making smart decisions?

Q. In this field.

A. We didn't have -- we had never built a smart phone before. That wasn't Oracle's business. Oracle's Java business was on big server computers, you know. We have never built a smart phone before. We had no experience in-house on this at all, or very little.

MR. VAN NEST: Could I have Page 23?

(Document displayed)

BY MR. VAN NEST:

Q. Take a look at Page 23, Mr. Ellison. It's right in front of you there.

(Witness complied.)

Q. And with respect -- and I have it on the monitor as well, if it's easier to see.

With respect to this area of evaluation, the organizational model, what you identified as problem three was:

"Very limited internal expertise to make smart decisions."

Right?

A. Well, again, smart decisions is about smart phones.

Q. Fair enough.

A. We had no experience about smart phones.

Q. And that was one of the problems with developing a smart phone at Oracle?

A. Oh, absolutely. We had never developed a smart phone before.

Q. Now, I take it you never did develop a smart phone, at least not based on this project?

A. No. As a result of this evaluation, we decided not to build a smart phone.

Q. And, therefore, you didn't, right?

A. Correct. We did not.

Q. And as part of this evaluation, was it part of this evaluation or separate evaluation that you considered buying Palm and getting technology from them?

A. It was part of this evaluation. As you said earlier, we didn't have a lot of -- we had no people in-house that had ever built a smart phone before. One way to get those people in-house would have been to buy Palm.

Q. Now, it was after all of these sessions and this Project Java Phone that you met with Mr. Schmidt, correct?

A. Yes.

Q. And you met with him in your home on the Peninsula sometime in March of 2010?

A. I believe so.

Q. Just of two of you were there?

A. Yes.

Q. And you discussed the probability with him of a joint project between Oracle and Google around smart phones, right?

A. Not -- well, we -- what we discussed was him using our Java in the smart phone, in Android. If you're calling that a joint project, okay, but it really was, to be precise, about them taking our Java and putting it in Android rather than their version of Java. That's what it was.

Q. And you had an argument for him that you had a new virtual machine that was better than what was in Android, right?

A. Yes.

Q. By then you had done this analysis of Android and concluded that you, at Oracle, could do better?

A. Yes.

Q. You told him your virtual machine would boot faster, right?

A. Yes.

Q. That it would run faster?

A. Yes.

Q. That it would save power?

A. Yes.

Q. That it would save Google money?

A. If they got it from -- if they got it from us rather than doing it themself, yes. Yes.

Q. And didn't you tell them that Sun, Oracle and Google could have a joint project to build this?

A. No. I said that we would build -- they could take our version of Java and put it into Android. That is not a joint project. That is, we would develop Java. They would take our Java and put it in Android. That is not a joint -- that is not a joint project, no.

MR. VAN NEST: I'd like to play as an admission, a party admission, your Honor, from Mr. Ellison's deposition at Page 83, Lines 16 through 25.

THE COURT: Go ahead.

(Videotape played in open court.)

BY MR. VAN NEST:

Q. So this was a proposal for joint technology development between Oracle and Google, right?

A. If you want to call it a joint project or -- obviously, I called it a joint project, but it meant we would build Java and they would incorporate into the Android phone. I called it a joint project, okay, but...

Q. You never reached agreement with Mr. Schmidt that day, right?

A. Never reached agreement with Mr. Schmidt, correct.

Q. I think you also mentioned meeting with Mr. Page; that was several months later?

A. Yes.

Q. You met with him again at your home on the Peninsula?

A. Yes.

Q. And just the two of you were present?

A. Yes.

Q. You made the same proposal to him for a joint technical development between Oracle and Google, right?

A. I gave him that, among other options. I gave Eric other options, Eric Schmidt other options as well.

Q. And your idea, as expressed to Mr. Page, was that Google engineers and Oracle engineers would work together to improve Android, right?

A. The idea was they could use our version of Java in the

Android phone. And if you can -- if you want to call that our engineers working together with their engineers on Android, you can call it that.

I'm just saying that our engineers would work on Java, give it to the Google engineers to include in Android. If you want to call that working together, that's fine.

Q. Well, didn't you call it working together?

A. I'm happy to call it working together.

Q. You did call it working together, didn't you?

A. I certainly -- okay.

MR. VAN NEST: I would like to play as a party admission from Mr. Ellison's deposition Page 90, Lines 23 through 91, Line 1.

THE COURT: All right. Go ahead.

(Videotape played in open court.)

MR. VAN NEST: Did we get his answer? Could we play that again, please.

(Videotape played in open court.).

MR. MR. BOIES: Your Honor, for context can we have the previous question and answer read?

THE COURT: All right.

MR. VAN NEST: We can. Certainly, your Honor.

THE COURT: Do you have it handy?

MR. VAN NEST: I do.

THE COURT: All right. Here is the way it works.

You know I told you that nothing a lawyer ever says is evidence, but counsel is going to read from the deposition. That will be evidence because he's reading from part of the evidentiary record in a deposition.

Go ahead, counsel.

MR. VAN NEST: I was going to actually play the two questions and answers together. Is that -- or do you want me just to read it?

THE COURT: I think you ought to just read it because when you played it, it got clipped off. So why don't you read it from the transcript?

MR. VAN NEST: The prior question is at Page 90, Line 18.

"QUESTION: So this was a proposal for the joint technology development between Oracle and Google, right?

"ANSWER: It was an offer, yeah. It was an offer. Yeah, you know, to work on the Java inside of Android together.

"QUESTION: And the idea would be Google engineers work with Oracle engineers, improve the product, put out a better product?

"ANSWER: Correct."

THE COURT: Thank you.

BY MR. VAN NEST:

Q. Mr. Ellison, that never happened either, right?

A. No.

Q. The next thing that happened after that was this lawsuit, right?

A. Yes.

MR. VAN NEST: Nothing further your Honor.

THE COURT: Thank you.

Mr. Boies?

MR. MR. BOIES: Thank you, your Honor.

REDIRECT EXAMINATION

BY MR. MR. BOIES:

Q. During cross examination you were asked whether you knew when the GPL license was first available; do you recall that?

A. Yes, I do.

Q. Was the GPL license available in 2009 and 2010?

A. Yes, it was.

Q. You were also asked and shown questions from your deposition saying nobody owns Java programming language, anybody can use without royalty; do you recall that?

A. Yes.

Q. And you said you were not sure; do you recall that?

A. Yeah. I'm not a lawyer. I'm not exactly sure what those words mean.

Q. And why are you not sure?

A. I don't know if you can copyright a language. I just don't know the answer to that question.
Q. Okay. Now, you were also asked questions that said, Oracle didn't create Java, didn't create these APIs. Sun created these APIs. Do you recall that?
A. Yes.
Q. How did Oracle get the APIs?
A. We bought Sun and we bought all the assets of Sun and as a part of buying Sun, we got Java.
Q. And the assets that you bought when you bought Sun included a lot of things other than Java, correct?
A. It did.
Q. But was Java an important part of why you bought Sun?
A. I publicly said of all the things we have ever purchased, by far the most important thing we have ever purchased was Java.
Q. And when did you say that publicly?
A. Shortly after we announced we were buying Sun.
Q. Now, you were also shown some statements where you said you were flattered that there was a Java phone in the marketplace from Google; do you recall that?
A. That's correct.
Q. Did you ever say that you were flattered or favored an incompatible version of Java?
A. No. I always thought we could persuade Google to make

their version of Java compatible. We want Java to be as wisely used as possible, but we want -- you know, everyone else makes their versions compatible. We thought Google would make them compatible, too.

Q. Other than Google, is there any company that you're aware of that is using Sun or Oracle APIs that has not made their version compatible?

A. Nobody.

Q. You were also asked a number of questions about whether Oracle wanted to build a smart phone and what the process was; do you recall that?

A. Yes.

Q. Was there anything in any of the documents that you reviewed in connection with that or any of the discussions that you recall that dealt at all with whether or not Oracle's intellectual property was being used by Google?

A. Could you repeat? I'm sorry. Repeat the question?

Q. Sure, sure. You saw a lot of documents.

A. Yeah.

Q. About the smart phone project.

A. Right.

Q. Did that have anything to do with the issue of whether or not Google was infringing Oracle's copyrights?

A. I'm not -- I don't think I saw any of that in relation to our smart phone project. The -- the --

MR. VAN NEST: Objection, your Honor. He's answered the question.

THE COURT: Do you think you've answered the question?

THE WITNESS: No, I don't, your Honor.

THE COURT: Please continue.

A. So, we certainly -- I think Jonathan Schwartz sent me a note after we bought Sun -- or agreed to buy Sun where he outlined the battle Sun was having with Google. And at that point I acquainted myself with all of the details of Google's use of Sun's intellectual property in the Java phone.

So, but that's when I first learned about the use of Sun's intellectual property for the phone and I knew that --

THE COURT: We're getting into --

THE WITNESS: Okay.

THE COURT: (Continuing) -- hearsay at this point. So we need to ask a fresh question.

THE WITNESS: Sorry.

MR. MR. BOIES: Thank you, your Honor.

BY MR. MR. BOIES:

Q. You were also asked when you first met with Mr. Schmidt. Mr. Schmidt is with Google, correct?

A. Yes.

Q. And you said it was in 2010; do you recall that?

A. Yes.

Q. When did the acquisition of Sun actually close?

A. January, 2010.

Q. And prior to the time that it closed, were you empowered in any way to discuss with Google or anyone else Sun's intellectual property?

A. No.

MR. MR. BOIES: No more questions, your Honor.

MR. VAN NEST: May I follow up, your Honor?

THE COURT: Briefly.

RECROSS EXAMINATION

BY MR. VAN NEST:

Q. Mr. Ellison, you made reference to an email from Mr. Schwartz that you received?

A. Yes, yes.

Q. That email --

MR. VAN NEST: May I approach the witness, your Honor?

THE COURT: Yes, you may.

BY MR. VAN NEST:

Q. Let me show you Exhibit 2362, please.

(Whereupon, document was tendered to the witness.)

Q. Do you recognize it?

A. May I just have a couple seconds?

Q. Certainly.

(Brief pause.)

A. Yes.

Q. Is that the email?

A. Yes.

MR. VAN NEST: I offer 2362 into evidence, your Honor.

MR. MR. BOIES: No objection, your Honor.

THE COURT: 2362 is received.

(Trial Exhibit 2362 received in evidence)

BY MR. VAN NEST:

Q. Let's focus on what you saw from Mr. Schwartz, the bottom of the page.

(Document displayed)

A. Okay.

Q. It's below "Larry, first congrats." Below that.

This is an email from Mr. Schwartz to you, correct?

A. Correct.

Q. Right after you agreed to acquire Sun, right?

A. Yes.

Q. He says:

"There's obviously a lot we didn't focus on during the past week... from the instrumentation we're building into our open source assets to the battles with Adobe

Flash/Google Android, Microsoft's distribution dependencies, et cetera."

Right?

A. Correct.

Q. You received this email in April of 2009, right?

A. Correct.

Q. And then it was after you received this email that you went on stage at JavaOne in June of 2009 and made those remarks about being flattered by Android, right?

A. Yes.

Q. And it was after you received this email that you collected up all those Oracle engineers to do Project Java, right, Project Java Phone? That was after this email, correct?

A. Correct.

Q. And all that work we talked about and all that review, that was after this email?

A. Yes.

MR. VAN NEST: I have nothing further, your Honor.

MR. MR. BOIES: Nothing more, your Honor.

THE COURT: May Mr. Ellison step down and be excused from any subpoena?

MR. VAN NEST: Yes, your Honor.

THE COURT: All right. Mr. Ellison, thank you for coming. Please leave behind our documents, but take with you any that you brought.

THE WITNESS: Thank you.

THE COURT: Have a good day.

(Witness excused.)

THE COURT: All right. It's now 11:05. Who is the next witness going to be?

MR. JACOBS: That would be Mr. Kurian, your Honor.

THE COURT: I think we will take a 15-minute break a little early.

Remember the admonition. We'll take a 15 minute break.

(Jury exits the courtroom at 11:03 a.m.)

THE COURT: Please be seated.

Here are some things for the record. When you play a videotaped deposition -- I think I mentioned this at the pretrial conference -- the court reporters do not take it down. It's because they only record the spoken word here in court. They don't record videotaped things.

So how will your record look to the Court of Appeals? Well, the way you have do it, the only way to do it, is for you to put into the record a marked exhibit, a CD, that has the exact excerpt that you played; not the entire deposition, because that won't do them any good.

You need to give to Dawn by tomorrow morning what you played today and give the other side a copy.

Usually in these trials the lawyers agree upon a

procedure so that if you both agreed, maybe you did it at the end of the week or some interval that's better for you, that's fine with me; but if you don't -- if you don't pay attention to your record, it will get out of control and then you'll be bombarding -- I don't know. You'll be bombarding the conformance with excuses. So I urge you to pay attention to this detail.

Now, when you do read one to the jury, for example, it's not on the videotape, I think what Mr. Van Nest did was pretty close to the way it should be done, which is you say "Question." You pause. Then you read the question exactly. Then you say "Answer." Pause. Read the answer exactly. And that's so the court reporter, as well as the jury, will know where the question started and stopped and the answer started and stopped.

So far you have only done it that one time, but in most trials that's the way it's done all the time.

MR. VAN NEST: You said "pretty close," your Honor.

THE COURT: I think you didn't use the word "Answer," but that -- it was clear enough. But you should always say "Question," "Answer." Because they don't know where the question starts and stops or the answer starts and stops unless you do that, but in context it was close enough.

So I ask you to pay attention to this CD thing so that -- or I guess it's called DVD thing, so that it doesn't

get out of control and our record is maintained properly.

And then while we're on the record, I started with 2939, but I know that you had those deposition exhibits and I still need to have the list of the ones that were moved into evidence this morning. So at the break if you can hand that to Dawn, I will add it to my list.

Okay. So we'll take a 15 minute --

MR. VAN NEST: Your Honor, can I ask a quick question?

THE COURT: Of course.

MR. VAN NEST: With respect to the CD part of it, should we simply put on one CD everything that we played that day, or do you want a separate CD for each witness?

THE COURT: Everything for one day would be even better, because you could probably put the entire trial on one CD if you wanted to. But if you wait that long, it will get out of control. You'll get into an argument. You ought to do this in short pieces. Like, one day is good. One week is good. That will be fine. But if you wait til the very end, I promise you, you all will be disagreeing on what was actually played and then there will be a fight.

MR. VAN NEST: Then should we give it to the court reporter? Is that what she likes?

THE COURT: That's correct. It's actually -- it needs to be identified as excerpts played on April 16th or

17th, as the case may be. All right?

MR. VAN NEST: We'll do it.

MR. JACOBS: Briefly, your Honor, this may have been covered with the jury in the jury assembly room.

We're avoiding contact with them in the elevators and the hallways. It would be perhaps helpful to explain to them that it's not because we're unfriendly, but because that's the proper protocol.

THE COURT: All right. I always tell them that and if I haven't already, I will tell them that.

Anything more?

MR. VAN NEST: No, your Honor.

THE COURT: All right. 15 minutes. Thank you.

(Whereupon there was a recess in the proceedings from 11:08 a.m. until 11:26 a.m.)

THE COURT: Please be seated.

Thank you. Back to work. Call in the jury.

You can bring in your witness.

(Thomas Kurian enters the courtroom.)

THE COURT: Please have a seat. We can swear you in after the jury is present. Thank you. Welcome to the Court.

(Jury enters courtroom at 11:27 a.m.)

THE COURT: Welcome back. Be seated, please.

Is everything on over there now?

(Jury nodding affirmatively.)

THE COURT: Okay, great.

Okay. Oracle may call its next witness.

MR. JACOBS: Your Honor, Oracle calls Thomas Kurian.

THE COURT: Mr. Kurian, please stand and raise your right hand.

THOMAS KURIAN,

called as a witness for the Plaintiff herein, having been first duly sworn, was examined and testified as follows:

THE WITNESS: Yes.

THE CLERK: Okay. Thank you.

THE COURT: All right. Very well. Welcome. Please be seated.

You see how close I am to this? You've got to be about that close, and it will move back and forth. Make it easier for you.

So why don't you say your name and let's see if you're coming through?

THE WITNESS: Thomas Kurian.

THE COURT: Very good.

Thank you, counsel.

DIRECT EXAMINATION

BY MR. JACOBS:

Q. Mr. Kurian, could you introduce yourself again by name and title to the jury, please?

A. My name is Thomas Kurian. I'm Oracle's executive

vice-president for software development.

Q. How long have you worked for Oracle?

A. Since 1996.

Q. And what are your responsibilities now?

A. I run -- I am responsible for the development and the design and delivery of Oracle software products.

Q. Who do you report to?

A. I report to Oracle's CEO, Mr. Larry Ellison.

Q. How does Java fit into your job responsibilities?

A. The development team that builds our Java products reports into my organization. And we -- I'm also responsible for a number of Oracle's software products that are themselves built in Java.

Q. What's the family of products called that is based on Java?

A. So there are two sets of products that we, as Oracle, build using Java technology. There is a product family called Oracle Fusion Middleware.

And we also sell a number of business applications, things to do, things like customer relationship management, supply change, managing accounting. They are called package applications that are also written in Java.

Q. How do you happen to find yourself working at Oracle in 1996?

A. I have a computer science and electrical engineering

undergraduate degree from Princeton University. I was a software developer myself for a number of years in college and before college.

I have a master's degree in business administration from Stanford.

I joined Oracle from a company called McKinsey Consulting, and I have been at Oracle since '96.

Q. And how did you happen to find yourself at Princeton?

A. I came to the United States from India on a scholarship. I learned programming before college in high school. I came to study my undergraduate degree in computer science and electrical engineering at Princeton.

Q. I would like to talk with you about Oracle's use of Java before the acquisition of Sun, so Oracle, as a customer or licensee of Java technology.

Just to get us oriented, on our timeline here Oracle acquires Sun in January of 2010. So I would like to ask you about the period before that.

Can you tell us how it came to be that Oracle used Java in its own business before that acquisition occurred?

A. Oracle started building Java products as early as 1996. And there are four places that we use Java within the Oracle product line.

First of all, in our data base. Second, in our development tools. Third is in our middleware product.

Fourth, in our business applications.

In our data base we integrated Java inside of our data base so that -- you know, a data base is a piece of software that allows you to manage information and then ask questions from that information. And we put Java into the data base so that developers who are writing applications accessing data bases to ask questions and query information could do that in Java, because it's much more open than the other languages that people traditionally use to program databases with.

Second, we then got requests from customers to build a development environment so that programmers could have a nice interface which they could write programs in Java. And so we introduced a product called J Developer, or Java developer in other words, to allow developers, programmers to write programs in Java.

Then as the internet became the way that people built applications and built websites, we introduced a product called Oracle Fusion Middleware. And what it really is is a collection of technologies to allow people to build websites, very sophisticated websites using Java.

And then, finally, we also took our packaged applications. When I say "packaged applications," people want -- you know companies want to do accounting. They want to do -- manage how they manufacture products. They want to have software that handles how they distribute products and manage

their warehouses. Retailers want to have software that manages how you buy things at the point of sale. We deliver a lot of packaged applications to enable companies to do that and those packaged applications, a number of them, are also written in Java.

Q. What was it about Java that made it so attractive for these various uses that Oracle was making of it?

A. There were three reasons we chose to use Java to build these solutions. One is the availability developers. Second is the interest from corporate customers. And third is the cost for Oracle. So I'll explain maybe each of them.

Q. Please.

A. So the availability of developers. You know, by the early 2000s Java had by far become the most popular programming language in the industry. And so when we built software, it was easy for us to find developers who new Java to build our own engineering teams. And, also, when we sold software to companies, the information technology departments of companies often said, "Can you guys implement your software in Java because then it's easier for us to find people to build applications using Java?" So that was reason number one.

Reason number two is Java enforced a set of standards that allowed corporate customers to have choice across different vendors. Okay? So take as an example if you buy middleware, there is a component called an application server

or web server. It's the thing that if you click on a website today, most websites are served up using an application server or web server. And Java enforces a set of standards that multiple vendors support; oracle, IBM, Red Hat, a number of players. So why corporate customers like that is they can get their internal departments to build applications using Java to the standards and then one day if they decide they don't want to use Oracle software, but they want to go and switch over to IBM, or they don't like IBM software and they want to switch over to somebody else, they have that choice and they are not locked into a particular vendor.

So the second reason was corporate customers demanded us to deliver solutions in Java and many of them have, as part of their requisition process, when they put out a bid for software they say: Are you compliant with X version of the Java specification in order to get -- you know, to have this freedom of choice.

The third reason we chose to use Java was cost. And there were two reasons for cost for Oracle. One is, Java has a capability to -- they call it write once, run anywhere. And what it does is it essentially means that you don't bear the heavy cost in porting our software to run across different operating systems or processors.

Q. You better explain porting and how it relates to operating systems and processors.

A. So, normally when you write a piece of software, you write it in a language called a programming language. You then take that programming language and you run it through something called a compiler. Okay. A compiler takes that, let's call it English type syntax, and converts it into machine readable code.

With the vast majority of programming languages, when you convert it from that English syntax to machine readable code, you have to write it for -- there's a specific compiler and specific machine readable code for each hardware platform. So if you were to compile it for Windows, for example, you used one compiler for Microsoft to do that. If you compiled it to run on IBM's processor called the Power Processor, you used a different compiler to process that. If you chose to do it on HP's computer, you had to use yet another processor to do that or compiler to do that.

With Java, what Java has is something called a bytecode, a portable bytecode. So in Java what you do is you take the source code, which is written in English like syntax, and you compile it using a Java compiler and you get something called a bytecode, and that bytecode is portable both across operating systems and hardware platforms. So once you compile it once, it works on Intel's processor. It works on IBM's processor. It works on the Sun Spark Processor. It works on HP's processor. And it also is portable across Java virtual

machines that are compliant with the standard.

And so as a result, we don't have to have hundreds of people responsible for taking our software when it's written in Java and certifying it on every different hardware platform and operating system. So that's what I mean by lowering the cost of porting.

The other factor for us is because of the migrant Java ecosystem, there is a lot of third-party tools and utilities and other things that we can use in building our own software. And so the forth factor is also lower cost.

Q. How did the Java APIs and class libraries fit into this picture of the value to Oracle when you were making these decisions to base your products on Java?

A. So the APIs and class libraries -- maybe I should explain what an API is or a class library.

When you write an application program, you write it in a language called a programming language. And a programming language is just a descriptive set of syntax that says -- you know, it's an expressive syntax to say, here is how you give -- you describe a set of instructions that you want to give to a computer to then have the computer operate.

Class libraries are different. What class libraries are are packaged programs that are encapsulated with an interface called an Application Programming Interface or API. And what they provide is essentially prepackaged functionality

so that developers don't have to write all of that functionality every time themselves. Okay?

So when you look at these class libraries, they are basically prepackaged utilities that provide a lot of useful functions for programmers to build applications with. And they are abstracted through a blueprint or a design interface called the programming interface that says if you're calling this piece of logic, you call it through this well-specified interface and then this piece of code will execute some logic and return a set of results to you. Okay?

Now, for us, as a company building applications in Java, the availability of these class libraries is very important for two reasons.

One is it improves the productivity of our developers, which means we can build applications much more quickly.

And, second, in the Java specification they specify a standard set of these libraries that the vendors who offer solutions need to comply with. And, therefore, it also means that we can depend on a set of standards when we build applications to these APIs.

Q. So explain that last point just a little bit further. You're talking now about the ability of the Oracle applications to run on multiple companies' versions of the Java runtime environment, the Java virtual machine?

A. Correct. So as an example, if you look at our accounting software today, we make that available on at least five different operating systems and hardware platforms.

Windows -- so on Windows and Linux, which are two operating systems running on the Intel processor, we use our own Java virtual machine to run on those platforms this accounting software.

Now, on IBM's computer, which is called the Power Computer running their operating system called AIX, they have a far more sophisticated virtual machine than we do and our same accounting software runs on that platform. And because both virtual machines implement the same specification and have the same class libraries, we don't have differences. You know, our software can run across these operating systems without any issues.

Q. There is a particular specification in the Java world that Oracle made its products consistent with, compliant with on the application side.

A. So on the technology side, we -- there are multiple specifications that we support in the Java world. Our application programs depend on various versions of Java Standard Edition, which is -- Java Standard Edition is just a specification for Java programs that run on a desktop computer, okay?

Java Enterprise Edition is a specification to say how

programs should run on a server or an enterprise computer, as opposed to a desktop.

Micro Edition, or ME, is for it to run on a -- you know, a micro device, like a phone or a tablet computer.

And so our software packages, depending on the kind of software, comply with these standards. So we support Java EE, Enterprise Edition, 5 and 6 in our middleware and business applications. Our tools, for example, support Java Standard Edition 5. And today they support both Standard Edition 6 and 7. So our products always comply with a specific version of one of these standards.

Q. Again, before the acquisition of Sun, did Oracle take a Java license from Sun?

A. Yes, we did.

Q. And why did you do that?

A. We took a Java license from Sun for three reasons. One is our engineers used to read the specifications from the Sun Java standards specifications in order to implement, for example, our Java virtual machine or our middleware.

Number two -- so, we took a specification license.

Number two. We certified our products to comply with the certain versions of these standards. I mentioned Java SE5 and 6 and Java EE5. And so we took the Test Compatibility Kit license, or TCK license, to test and certify that our products were compatible with the specific version of the specification.

And, third, we also took a commercial license because we distributed, you know, Java technology as a vendor of Java technology.

So we took all three licenses and we did that before we acquired Sun for a number of years.

Q. Just spend a little bit more on the TCK license and how that fits into the overall picture here. That terminology may be a little confusing.

What exactly did Oracle get from Sun with respect to the TCK license?

A. So maybe I can be specific. When we implemented Oracle Fusion Middleware, we, as a vendor of the technology, did two things.

First is our engineers read the specifications of various parts of the standard in order to -- you know, because they are the blueprint or the design specifications for how the software should be implemented. And so we, you know, use the click-through license called the spec license or specification license to help our engineers implement the software.

Now, once you implement the software, there is a separate license called the Test Compatibility Kit, TCK kit. What you get with it are three things.

First is, you get a set of tests that are literally test suites that you can then run and test your software with and either you pass or you fail, and you technically have to

pass those test suites. Okay? So part one is you get the test kit.

Part two is you get the ability to submit the results of your testing to say you've passed and you submit that to the Java standards process within something called the JCPO, Java Community Process Organization, to say I qualified.

And third, if you pass or qualify, then you get some of the other capabilities, like the trademarks, the copyright and the ability to brand your solution as being compatible with that version of Java.

So you get these three things with the TCK.

Q. So to review, before the acquisition the three licenses that were -- that you have referred to that Oracle used in its business were the specification license, the TCK license, and the commercial license?

A. Correct.

Q. And are those three licenses -- after the acquisition, what did Oracle do to that basic licensing structure?

A. They are all the same. They are available today.

Q. And then is there -- are there royalty-free licenses that are available for Java technology?

A. There are two additional licenses. There is a royalty-free license. The royalty-free license is meant to be used by non-profit organizations, like universities and research organizations, so that they can both teach their

students Java, so that you get more developers coming out of college knowing Java, and, also, research institutions that want to use Java for non-commercial purposes.

There is also the open source license under something called the GNU public license, or GPL, that's available.

Q. And can you explain how that works?

A. How the GPL works?

Q. Yes.

A. So, there's a version -- there is a Java implementation called the Open JDK. Open JDK is the so-called reference implementation of Java. It's implemented in an open source project called the Open JDK in which a number of companies participate, eight or nine companies participate: Oracle, IBM, a company called Canonical, Red Hat, a number of others. They all contribute source code. Their engineers participate in it. They contribute source code. And the resulting implementation of a virtual machine is called Open JDK.

Now, developers who want to implement their own Java virtual machines can take source code from Open JDK under something called the GPL, GNU public license, with one requirement; that the source code that you use and if you implement a virtual machine or some other technology derived from that source code, you have to contribute it back to the Open JDK source base.

And the reason for that is very simple. The Sun

organization and Oracle and IBM and others who participate in Java want to make sure that Java does not fragment. And by enforcing the ability for people to get access to the source, but to contribute back any modifications, you ensure essentially that Java is not fragmented.

Q. I'd like to return to the specification license for a minute.

(Whereupon, document was tendered to the witness.)

Q. And I've placed in front of you Exhibit 610.1 in the folder. And can you please identify this document?

A. I have it.

(Document displayed)

Q. Can you tell us what this is?

A. This is a document which is a specification license for Java 2 Platform Standard Edition Development Kit 5.0 specification.

MR. JACOBS: 610.1 offered into evidence, your Honor.

MR. VAN NEST: No objection.

THE COURT: Received in evidence. You may publish, if you wish.

(Trial Exhibit 610.1 received in evidence)

BY MR. JACOBS:

Q. You referred to this earlier just a bit. You referred

earlier to the specification license, but let's walk through it.

THE COURT: Is it coming through in the jury box?

(All jurors respond affirmatively.)

THE COURT: Okay. Great thank you. Go ahead.

BY MR. JACOBS:

Q. What in general does this license grant the licensee the right to do? What permissions does it give?

A. The paragraph, the second paragraph I think is the one that tells you what it grants:

"Sun also grants you a perpetual non-exclusive, worldwide, fully paid-up royalty-free limited license (without the right to sublicense) under any applicable copyrights or patent rights it may have in the specification to create and/or distribute an Independent Implementation of the Specification that" --

Q. Stop right there. Let's break it down a little bit.

First of all, how does this specification relate to the creation of class libraries?

A. So if you're a developer who is implementing class libraries, the class library is a part of this specification. So typically the class library design blueprint is available on a website. And as a developer when you go to that website, you

click and accept a -- you know, a specification license that allows you to then read the blueprints and decide to implement according to that blueprint.

And so this specification license is the one that governs the way in which developers can understand the design of the blueprints.

Q. And just to be clear, is this a developer of an implementation of the class libraries or of Java applications?

A. This is a developer who is implementing the class libraries.

Q. Okay. And then it says there:

"Applicable copyrights or patent rights."

And it says:

"You can create, distribute an independent implementation of the specification."

And then it goes on to give us some more requirements. Can you walk us through that, the additional requirements?

A. Okay. That -- and then I'm reading from, you know, (i):

"Fully implements the spec or specs including all of its required interfaces and functionality."

Q. What is that driving at? As Oracle administers these licenses, what is Oracle trying to make sure that people do?

A. In the Java standards organization and the Java

specifications one of the very important clauses has always been that an implementation has to -- an implementation of Java has to fully implement the specification. And by fully implementing the specification means you can't implement a part of the design that constitutes the specification.

And this -- and it's tied to the important requirement that a vendor cannot implement part of the specification because if they implement just part of the specification, then standard Java programs will not run on their version of the -- of Java.

And, secondly, if you write to their specific implementation, then it may not work across other vendor's implementation of Java.

So by definition to ensure that Java is portable across implementations for multiple vendors, it's very important that any developer or vendor implementing Java should fully implement the specification.

Q. And then if you look at the next item, item (ii), can you explain what that's all about?

A. Item (ii):

"Does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected damages, classes, Java interfaces, fields or methods within the Licensor Name Space other than

those required/authorized by the specification or specifications being implemented."

Q. So in a nutshell, what's that all about?

A. So that's just a further clarification of this issue. What it says is if you're implementing these APIs, you cannot subset or superset.

By "subsetting" we mean you can't say there are 40 APIs and I'm only going to implement 25.

By "supersetting," you can't say there's 40 and I'll implement all 40, but I add two of my own within what they call the Name Space. Name Space is just a way of saying how you separate what's delivered as part of the Java standard and is considered part of the foundation and what each vendor then has to comply with.

And then there is a separate way that you can add your own classes for areas that do not conflict with what's part of the specification and that's in a separate what they call Name Space, which is a way of just demarcating what's within the standard versus what's outside the standard.

Q. And by making that demarcation, does the license aim to ensure that developers will not add or subtract from the Java APIs and associated class libraries?

A. Correct. It's done in that specific way to protect Java from vendors who want to, you know, add or subtract. And some

of this has come about because in -- in the early 2000s, Microsoft extended Java and they created a version of Java that did technically supersetting and they have -- and, you know, to protect against that kind of, you know, developing a version of Java that's not standards compliant.

Q. Do you have an understanding of what Sun did when Microsoft came out with that supersetted version?

A. My understanding is --

MR. VAN NEST: Objection, your Honor. Motion in limine.

MR. JACOBS: Consistent with the motion, your Honor.

THE COURT: It's hearsay anyway. His understanding is not direct personal knowledge. Sustained.

MR. VAN NEST: Lacks foundation.

THE COURT: Sustained.

BY MR. JACOBS:

Q. Let's go on to the next category, item three.

A. Category (iii) is:

"Passes the TCK (including satisfying the requirements of the applicable TCK users guide) for such specification."

Then it goes on to say:

"The foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for

any other purpose."

Q. Now, you mentioned the TCK earlier. How does this provision relate to the TCK?

A. What this says is if you're a developer who is reading the specification and implementing software based on the design blueprint in the specification, one of the requirements is that you have to then subsequently take the piece of software you built and then run and pass the TCK, or Test Compatibility Kit, associated with that version of Java, including satisfying the requirements in the applicable user's guide that describes how to run the test.

Q. And in order to get the TCK, do you have to take another license or does that just come with the spec license?

A. No. The spec license is free of charge. It's available as a click-through license on the website. And it's typically done that way because in implementing a piece of software, multiple developers in an organization may be reading specifications. And so each developer can -- should accept and can accept the click-through license on the website at no charge.

Now, when the vendor is going to sell that piece of software, distribute it, even at no cost, they then have to take a second license called the Test Compatibility Kit license and that one has a nominal charge and associated with that you get the test kits and then the ability to run and submit that

you're certified to the standards organization.

MR. JACOBS: Now, if you scroll down a little bit, Mr. Lee, to the independent implementation?

(Document displayed)

BY MR. JACOBS:

Q. So one of the requirements you read was that the implementation of the specification be an independent implementation.

What is an independent implementation under the license?

A. I will read the paragraph, if you don't mind, and then explain it.

Q. Uh-huh.

A. (As read)

"For the purposes of this agreement Independent Implementation shall mean an implementation of the specification that neither derives from any of Sun's source code or binary code materials nor, except within appropriate and separate license from Sun, include any of Sun's source code or binary code materials; and Licensor Name Space shall mean the public class or interface declarations whose names begin with 'Java, Java X, Com.Sun' or their equivalents in any

subsequent naming conventions adopted by Sun through the Java community process or any recognized successors or replacements thereof."

Q. So let's actually just take that in reverse order. The "Licensor Name Space" you explained that earlier?

A. Yes. The Licensor Name Space is when people write software in Java, there needs to be a clear way of saying, demarcating what does a specific version of Java provide out of the box as part of the reference implementation?

And then what is a vendor or developer going to build as part of their own class libraries, because a developer building an application can also build their own class libraries.

So the way that Java handles this is through something called a Name Space. A Name Space basically says when you're writing a program, your class libraries can be named the following things. And the vendor who supports the version of Java that's compliant with the standard has all their class libraries within Java, Java X, Com.Sun. So it provides a clean demarcation between a company.

Pick an example, CitiBank. They are building Java applications. They can say, We can build our own class libraries and put them in a Name Space called CitiBank.java.something or CitiBank.X, as long as we are not

putting it into Java or JavaX or Com.Sun. And the advantage there is both it allows companies and developers to build class libraries without stepping on and bifurcating what's the reference implementation and the standards compliant implementation of the spec. That's what this Name Space separation is. It's just a simple way of partitioning what's in the standard that a vendor needs to comply with and where does a developer or a pool of developers building Java applications, where can they put their own class libraries?

Q. Back to Independent Implementation. What's that clause driving at?

A. Independent Implementation basically means that there are two kinds of developers in the -- who might be building Java technology. One kind of developer is somebody who says in addition to reading the specification documents, I'm actually going to access some of the source code that's in the Open JDK project and use that source code when I write my program. That's type number one.

What Independent Implementation means is, I read the blueprint that's part of the specification, but then I write my own implementation in what's called a clean room, which means I don't take any source code or binary code. I write it myself.

What this document says is that the specification license applies, and the TCK compatibility requirements apply whether you did an independent implementation or you took

source code from the Open JDK project. That's what it means.
Q. Now how does a clean room, the concept of a clean room fit into the idea of an Independent Implementation?
A. A clean room is basically a word used to refer to an engineering project. You know, think about clean rooms. They typically came from the semiconductor industry where people said -- you know, you went into a room and you had specific, you know, clothing on that meant you didn't contaminate any, you know, semiconductor chips.

And by "clean room" in software, it means that your engineers who wrote the code looked at the design specifications so that they could understand the blueprint that was in the design, but didn't look at any source code and wrote the code by their own, you know, design. Meaning, they complied with the blueprint, but they wrote the code themselves. That's what we mean by Independent Implementation.
Q. Now, if we scroll down to the trademarks provision of this license. Can you just read the first three lines of that?
A. It says:

> "Trademarks: No right, title, or interest in or to any trademarks, service marks, or trade names of Sun, Sun's licensors, specification lead or the specification lead's licensor is granted hereunder."

Q. And then it goes on to talk about a couple of trademarks.

Do you see that?
A. Yes.
Q. Okay. So as you administer the licenses and as you work with them at Oracle, if you have met the requirements of the specification license, do you have the right based just on that to use the Java logo?
A. No, you do not.
Q. What do you need to do in order to get that right?
A. You have to purchase the Test Compatible Kit license, or TCK license. You have to run the test suites. You have to pass the test suites. You then have to submit your results to the certification organization and then once they are comfortable that you, in fact, did pass correctly, then you get the ability to use the brand and the logos.
Q. What is the underlying purpose, then, of these requirements of, for example, not supersetting, not subsetting, implementing the whole interface? What's that all driving at?
A. It's all driving at the important notion that if you want Java to be write once, run anywhere. Write once, run anywhere means write once. Run anywhere means you can depend on software to run across different hardware operating systems and you can depend on software to run in a consistent way across Java stacks from different vendors. All of these requirements are essentially protecting that Central Valley proposition for Java.

Q. Are there companies that have complied with the specification license, taken the TCK license, paid the fee, gotten the brand for an Independent Implementation?

A. Yes. There are many companies that have done so.

Q. Did Google do that?

A. No, they do not.

Q. Let's just spend a minute more on commercial licenses, royalty-bearing licenses.

I'm going to ask you about companies that -- and ask you what kind of -- what their license is for and how they are using Java in their business.

So Sony?

A. Sony uses Java inside their blue re DVD players. There's Java on the micro controller inside the blue Ray DVD player that does a bunch of functions that allows the DVD player to, you know, play the DVD as well as record, and the intelligence within the DVD player is managed using Java.

Q. How about Panasonic?

A. Panasonic puts it on some of their -- both televisions, smart televisions, as well as in their own version of DVD players.

Q. Cisco?

A. Cisco Systems sells the -- like a SoftPhone, which is the phone that you use, but it's got a lot of software in it to allow you to do intelligent things, like conference calling,

three-way conferencing, speed dialing, remembering people's telephone numbers, all of that. That's -- so in order for the phone to be able to do that, it needs a program that stores all that information and then manages it, and it uses a version of Java called Java ME to do that.

Q. How about RIM, Research in Motion, Blackberry?

A. The Blackberry, you know, device actually runs Java. When you use Blackberry's mobile device, it's actually running Java. It runs a version of Java called Java ME.

Q. GE?

THE COURT: GE or ME?

MR. JACOBS: I asked a new company. He said ME and then I asked about General Electric.

THE COURT: So your answer was ME?

THE WITNESS: ME.

THE COURT: Now, you're going to GE. You mean General Electric.

MR. JACOBS: Yes. Sorry your Honor.

A. General Electric sells home appliances, things like refrigerators, dishwashers, laundry machines and so on.

Within the refrigerator, for example, they use Java on a microcontroller and the value that they find with Java is that they can store a bunch of programs.

So as an example, if you leave and go on vacation and you want the refrigerator to be at a certain temperature and

when you come back on a particular date, you want ahead of time for the refrigerator to become even colder because you want to have cold drinks when you come back, for example.

In order for it to do that, it needs to have a stored program that saves the calendar and can manage whether you take the temperature up or down, and it does that using Java.

Q. Nokia?

A. Nokia's phones run on -- they have an operating system called Symbian, and on top of that they run programs in Java.

Q. EBay?

A. EBay is an auction site. You know, the website of eBay is served up using a technology in Java called Java Server Pages, a very simple thing. What it is is a way to build web pages using Java. So they took a license for what's called Java Server Pages Version 1.2 because they were doing a customer implementation of that version of the standard in order to be -- to make it really fast and their website really interactive with users.

Q. Visa?

A. Visa uses a technology called Java Card.

Q. Okay.

A. So when you use a Visa card, some of the Visa cards have a chip on them and on the chip they actually store things like a -- call it an electronic wallet and the ability to swap money and do transactions using that.

Visa specifically uses a component of the Java Card called a Java Card trimming tool and what it is is it allows you to take this Java Card, which they license, but then to cut off and trim so that no other program can run on that outside of what visa provides. And that's largely for security reasons, because they don't want somebody by mistake to get some software downloaded on it that may steal money from their bank account.

Q. And last is Amazon?

A. Amazon has a product called the Kindle. A Kindle is an electronic book reader. And Kindle is essentially running Java.

When you open the Kindle and read books, the technology that allows you to actually read the books and then download stuff from the Amazon site and handle all of that is actually based on Java.

Q. Were all these companies that you mentioned, are they licensees under what we call the commercial licenses for Java?

A. Correct. They all are.

Q. And then earlier you mentioned companies that had taken the spec and TCK license. Could you just briefly review some of those companies again?

A. So IBM is an example of a company that takes both the spec and TCK license. IBM is a, you know, big computer company that implements technology and Java. They sell a -- they offer a

Java virtual machine. They have their own middleware called Web Sphere that implements the Java Enterprise Standard or Java EE, and so they have got different Java products.

And they take both a spec license, because their engineers read the design specifications. They take a TCK license, because when they implement the software, they certify that their software is compliant with certain versions of the specification. And they also take a commercial license.

Q. And then we refer to the open source versions, the OpenJDK.

A. Right.

Q. Can you tell us more how that's actually used by users of Java, by companies that have access to that code base?

A. The OpenJDK is a project that has multiple participants. There's about 300 people who have contributed to OpenJDK from 40 different companies, okay.

What it is, is, really, it's a project where the development of Java Standard Edition, or as we call it the Java virtual machine for desktops and for, you know, devices is done through a community of people working in open source. Okay. So you can actually see the source code. You can contribute source code to it. You can petition for membership. And, you know, once granted membership, you can contribute source code to it.

And it's a joint -- it's an organization with a

governance structure managed by Oracle, IBM, Red Hat, and a number of players within that community.

Now, once you -- when source code is contributed to that, you can also take source code from OpenJDK, but you take it under a license called the GNU public license, or GPL. Okay.

The requirement of the GNU public license is, basically, that you can look at the source code, you can actually take the source code. But if you take the source code, anything that you do to modify the source code or build what they call a derivative work based on that source code, you have to contribute back.

Q. And, in fact, is the GPL used widely for commercial purposes? I'm sorry. I didn't ask that right.

Is OpenJDK under the GPL used widely for commercial purposes?

A. Most commercial companies don't use the OpenJDK because of the basic requirement that they want to -- commercial companies want to so-called add value. They want to take a Java virtual machine and they want to optimizes it in a certain way. Right. And "optimize it" is specialize it for their processor, tailor it to their device, optimize it for their hardware. And that optimization they consider as their own unique intellectual property.

So they -- under the conditions of the GPL, they

would have to contribute that back to the OpenJDK project. So, instead, what vendors do is they take a commercial license for that purpose.

Q. So we've talked now about the commercial license, the spec/TCK combination, and the GPL.

Is Android under any of those as Android relates to Java?

A. No.

Q. Post acquisition now, Oracle buys Sun. At some point you actually assume some responsibility for Java. And I'd like you to talk about your responsibilities and what some of the highlights of Oracle's further development of Java have been since the acquisition. Let's break that down a little bit.

After the acquisition, did Oracle have to play a different kind of role than it was playing in the Java Community Process?

A. Yes. After -- prior to the acquisition of Sun, we were a participant in the Java Community Process. And we implemented software that complied with Java standards. But we didn't -- actually were not the primary steward of the Java Community Process and the Java, you know, community organization.

So we acquired Sun in January 2010. Once we acquired Sun, we changed our role in three additional ways.

Number one, we became the steward and primary sponsor for the Java standard setting organization, called the JCP or

the Java Community Process, number one.

Number two, we also took over the engineering of the -- what we call the reference implementations and design specifications for many of the Java standards. Things like the Java Standard Edition, the design specifications for Java Enterprise Edition, et cetera.

And, third, we invested additionally, we also became the sponsors for the Java user groups, what's called the Java, you know, developer community, which has a number of forums and other things where we go out and do events to get developers excited about where Java is going and also train new developers on Java technology so that we can keep the developer ecosystem vibrant.

Q. Do you have an estimate for how much Oracle has invested in Java since the acquisition?

A. Annually, we spend hundreds of millions of dollars on Java.

Q. And any plans to change that?

A. Not -- no.

Q. Did you at some point get involved in analyzing Android for purposes of determining what Oracle would do about Android?

A. I evaluated Android to understand largely from the point of view of trying to understand, one, how we could make Android compliant with the Java specification, and, secondly, what technically would be involved to help make that possible.

Q. What did you learn?

MR. VAN NEST: Objection, Your Honor. Calls for expert testimony. Undisclosed.

THE COURT: Sustained.

BY MR. JACOBS:

Q. Did you actually conduct this analysis yourself, as part of your business responsibilities at Oracle?

A. I did, along with Mark Reinhold and some other people from our Java engineering group.

Q. And you did this not as -- in association with this lawsuit, but as part of your business duties, correct?

A. Correct, yes.

Q. What did you learn?

MR. VAN NEST: Objection, Your Honor. Same objection. It's expert testimony.

THE COURT: Well, it is, but it does not have -- it's okay as long as it's -- it's like a treating physician. As long as he was not retained, and did this as part of his job, it's okay. Isn't it?

MR. VAN NEST: I thought that had to be disclosed, Your Honor, under the rules.

THE COURT: Well --

MR. JACOBS: This is a -- laying a foundation for the discussions that he subsequently had with Google, in which this issue was addressed.

THE COURT: There may be a disclosure.

MR. VAN NEST: Discussions I have no problem with. It's this portion here, Your Honor --

THE COURT: I do think it's required that you identify even fact witnesses who are going to give opinion evidence.

MR. JACOBS: We're not asking him for any ultimate opinions, Your Honor. We're asking him for what he did, what he learned, and what he did with that information.

THE COURT: We're going to skip this part and come back to it after we have a hearing out of the presence of the jury. So see if you can work around it without having to get into this piece of it.

BY MR. JACOBS:

Q. Did you have meetings with Andy Rubin to discuss concerns Oracle had about Android?

A. Yes, we did.

Q. When did you first meet with him?

A. May 28, 2010.

Q. Did you meet with Alan Eustace of Google on a similar topic?

THE COURT: You need to be a little closer to the microphone.

THE WITNESS: May 28, 2010, was the date, since you asked.

And then we did meet with Alan Eustace sometime in June or early July 2010.

BY MR. JACOBS:

Q. And what did you discuss with them?

A. Discussions were around two topics.

Number one was, we proposed a mechanism through which Google could get Android to be compliant with Java. And there were three pieces of that discussion. The first is Android has a virtual machine inside of Android called Dalvik. And Dalvik runs a nonJava standards compliant bytecode. Meaning the way that you take a Java program -- a program for Android and then convert it is you first compile it to standard Java bytecode, and then you run a second step to convert it from Java bytecode to something called a dex bytecode.

And so step one was a discussion on how to either eliminate the Dalvik VM and replace it with the CDC virtual machine, which is compliant with Java Standard or, two, to get Dalvik itself to comply with the Java Standard. So that's one.

The second was to get the class libraries compliant with the Java 2, Standard Edition specification.

And the third was a discussion around if they had applications already written to run on Android, how could those applications be made to work on a standard Java virtual machine.

Q. Let's focus on the second of those three elements. What

were you driving at there?

A. Android implements -- you know, doesn't implement --

MR. VAN NEST: Objection, Your Honor. Same objection.

THE COURT: Sustained.

At this point, all you can do is if it came up in a direct conversation with someone at Google, he can recount the conversation. That would be fine. But as phrased that question calls for an opinion under Rule 26, and apparently was not disclosed.

So we will have to get at it -- he can testify to it only if it was actually spoken.

BY MR. JACOBS:

Q. With respect to the second item, Mr. Kurian, Mr. Kurian, what did you say to Google?

A. I specifically discussed -- we specifically discussed with Mr. Rubin that the Dalvik implementation and their implementation of Java needed to comply with the Java Standard specifications, which meant you had to implement the class libraries completely and also pass the TCK.

Q. Did Google accept that?

A. No.

Q. Did you also discuss with Google the fact that Google was using some class libraries in Android that had been imported from the Apache Harmony project?

MR. VAN NEST: Objection. Leading.

THE COURT: Well, again, in light of your earlier objections, counsel is trying to avoid that by focusing on the word discussion. So I'm going to let him lead to that limited extent.

Please, answer the question.

Overruled.

THE WITNESS: Yes, we did discuss the fact that within Android there is an implementation of Java class libraries taken from a project, an open source project called Apache Harmony.

And we pointed out that Apache Harmony had never been granted a Test Compliance Kit certification, and had never passed certification; and, as a result, in order to comply with Java they would have to replace those libraries.

BY MR. JACOBS:

Q. And what did Google say to that?

A. They did not accept that.

Q. How did they -- your discussions with Google end?

A. Oracle's president Ms. Safra Catz and I met with a gentleman, Mr. Alan Eustace. It's either in June or July 2010, early July 2010. We indicated the same two issues.

Number one, that they needed to eliminate their implementation of the Dalvik virtual machine and replace the class libraries that they had taken from Apache Harmony with a

compliant version from the Java community.

And they -- and that was the last discussion.

MR. JACOBS: I have no further questions.

THE COURT: Cross-examination.

CROSS EXAMINATION

BY MR. VAN NEST:

Q. Good afternoon, Mr. Kurian.

I want to start with your discussions with the Google representatives in 2010.

THE COURT: People are leaving and it's distracting. Anyone else want to leave? Be my guest. But after Mr. Van Nest starts, please stay here until the end.

MR. VAN NEST: I can't even get an audience.

(Laughter)

MR. VAN NEST: I can't keep one.

THE COURT: No, that's fine.

All right. The floor is yours.

BY MR. VAN NEST:

Q. You mentioned, Mr. Kurian, that you had spoken with Mr. Rubin about Apache Harmony --

A. Yes.

Q. -- is that right?

And you told him that some of the libraries in Android were based on Apache Harmony?

A. I told him that it was our understanding that some of the

class libraries in Android running on top of the Dalvik virtual machine were from Apache Harmony.

Q. Okay. Now, Apache Harmony, that wasn't owned by Sun or Oracle?

A. No, it was not.

Q. Apache Harmony, that's an independent project?

A. That is an independent project, yes.

Q. And Apache Harmony never took a license from Sun, right?

A. Correct. They were never granted a license from Sun.

Q. And Apache Harmony first published their product in 2005, right?

A. I'm not sure of the date, sir.

Q. Well, did you do any investigation to determine how long Apache Harmony had been out there without a license from Sun?

A. The specific version of Java that was implemented within the Dalvik virtual machine, the class library version was available as early as 2005, from Sun.

The primary contributions to Apache had been made by Intel and IBM post that timeline.

Q. Okay. But Apache had a group of class libraries, correct? You have to answer verbally, Mr. Kurian.

A. Yes, sir.

Q. And many of those class libraries were in Android. At least that's what you told the Google folks?

A. I said some of the class libraries were in Android, yes.

Q. Did that include all of the class libraries in the 37 APIs that are accused here in this case?
A. Yes, sir.
Q. So all 37 of those you told Mr. Rubin were in Android?
A. Correct, yes, sir.
Q. And those 37 were also in Apache --
A. Yes, sir.
Q. -- right?

And they had been in Apache since when, 2005?

A. As I said, I don't know exactly when they were in Apache. I said that the Java Standard version that had been published by Sun with those class libraries was as early as 2004.
Q. And Apache had been using those same libraries since 2004, without a license from Sun, right?
A. Again, as I said, Intel and IBM were the primary contributors to Apache Harmony.
Q. Excuse me, Mr. Kurian. I had a different question. I apologize for interrupting you.

These libraries were out there being used by Harmony with no license from Sun since approximately 2005, correct?

A. Yes. There had been many discussions between Apache and both Sun and the Java Community Process about resolving this matter. But no license had been granted.
Q. So in 2005, 2006, 2007, 2008, 2009, Apache was using the same libraries and the same APIs we're talking about in this

case, without license from Sun, correct?
A. Correct. And Sun had been trying to negotiate with Apache about a license because Apache had petitioned Sun for such a license.
Q. Now, you mentioned the APIs. I think you said they are the structure that allows you to access the libraries, the source code in the libraries, correct?
A. I said the APIs are the blueprint of the design for how a piece of program logic can be encapsulated for access by a calling program.
Q. Okay. But they are different from the source code that's in the library, correct?
A. There's -- an API consists of three parts. There's a blueprint, which is --
Q. Excuse me, Mr. Kurian. I had a more simple question.

They are different than the library and the source code in the library; are they not?
A. Let me explain myself, if you don't mind.

THE COURT: Let's let the witness explain his answer, and you can do a follow-up. Give us your answer.

THE WITNESS: An API consists of three important pieces.

There's the blueprint, which is the design of the API itself. The specification, which says exactly how the blueprint should be -- you know, what's the rationale for the

design of it and the relationship between this API and other pieces. And a third piece, which is the implementation, or reference implementation or source code that says, here's an example of how this API is implemented.

BY MR. VAN NEST:

Q. Fair enough.

And with respect to all three of those elements, Apache was using those as of, roughly, 2004, 2005?

A. Perhaps, yes.

Q. And in 2007, they were still using them with no license from Sun?

A. Yes. They had petitioned for a license. Sun had talked with them, but they had not been granted a license.

Q. So they were still selling and still using Harmony as late as 2009, without a license from Sun, right?

A. Apache does not sell APIs.

Q. Well, making --

A. They were an open source project, and there were people working in that open source community. But they were not a commercial institution that was selling software to anybody.

Q. Now, at the time you talked to Mr. Rubin, Apache still didn't have a license, correct?

A. Correct.

Q. And Mr. Rubin told you Google didn't need a license to use Android, right?

A. Yes.
Q. He said, We don't need a license --
A. Yes.
Q. -- correct?
A. Yes.
Q. Mr. Eustace said, We don't need a license?
A. Yes.
Q. Now, you mentioned the Dalvik virtual machine. That's a source code implementation in Android, correct?
A. That's a virtual machine that's implemented in Android, yes.
Q. And you were asking in these meetings to replace the Android virtual machine with the Java virtual machine that Oracle made, right?
A. Correct.
Q. And the Java virtual machine, that's different from the Dalvik virtual machine. At least that's what you told Mr. Rubin?
A. That's correct.
Q. It's a different implementation from the Java virtual machine, right?
A. You mean -- did you mean the Dalvik virtual machine is a different --
Q. Let me ask it again.
The Dalvik virtual machine is a different

implementation than the Java virtual machine, correct?
A. Yes.
Q. And you wanted Mr. Rubin to replace his virtual machine with yours, right?
A. Correct.
Q. And pay money for that?
A. We discussed technically replacing it. And we said there would be a separate discussion on the commercial terms.
Q. Commercial terms means you wanted money for that, right?
A. Sure.
Q. You would be asking for money if your product was being used in Android?
A. Sure.
Q. Which at the time you spoke to Mr. Rubin it was not. They were not using the Java virtual machine in Android, right?
A. Yes.
Q. They had built their own?
A. Yes, sir.
Q. And it was different?
A. Yes, sir?

MR. VAN NEST: I have nothing further, Your Honor.

THE COURT: Redirect, please.

REDIRECT EXAMINATION

BY MR. JACOBS:

Q. Mr. Kurian, you referred to this briefly. I just want to

make sure it was clear. What exactly is the -- is the commercial status of Apache Harmony?

Apache Harmony has not been granted a TCK license. There is a very specific technical reason for it. When Sun talked to Apache, and subsequently Oracle talked to Apache, one of the basic requirements of TCK compliance of an implementation of Java is that the -- the implementer of the specification has to guarantee that not only are they compliant with the specification in their implementation, but any organization downstream from them that uses that version of Java or modifies that version of Java will also remain compliant with the specification.

Now, Apache Harmony could not guarantee that downstream users from Apache would remain compliant with their implementation because of the way that the Apache software license is designed.

A downstream company -- let me take a simple example. Apache Harmony project may itself comply with today and going forward with the Java Standard Edition specification and say, we certify that our test suites are passed, et cetera.

But an organization downstream from Apache -- pick an example -- a telecommunications vendor who takes the source code from Apache, and they are free to then modify that source code under the Apache license without any constraints around contribution back. As a result of which, downstream

organizations will not be able -- you know, there's no enforcement mechanism to say anybody downstream from Apache could be granted that license and would remain compliant.

That's the central reason why Apache itself has never been granted a TCK license from the Java community organization.

Q. Are you aware of any commercial uses of Apache Harmony other than the packages of class libraries that are in Android?

A. I am not. Additionally, IBM and Intel, who were the primary contributors of the source code in Apache Harmony, have now moved their source code contributions to OpenJDK.

MR. JACOBS: Thank you very much.

MR. VAN NEST: Just one or two, Your Honor.

THE COURT: Fine.

RECROSS EXAMINATION

BY MR. VAN NEST:

Q. Mr. Kurian, during the period before Oracle acquired Sun, was Mr. Schwartz running Sun?

A. Yes, sir.

Q. Was he responsible for Sun's relationship with Apache Harmony?

A. I'm not sure of that, sir.

Q. He was the boss?

A. He was the CEO of Sun Microsystems.

Q. Was Mr. Schwartz ultimately responsible for Sun's

relationship with Android, as CEO, as well?

A. Perhaps, yes.

MR. VAN NEST: Nothing further, Your Honor.

MR. JACOBS: Nothing further, Your Honor.

THE COURT: All right. May the witness step down and be excused, not subject to recall?

MR. VAN NEST: Yes, Your Honor.

THE COURT: All right. Thank you, sir. You're free to go.

THE WITNESS: Thank you.

(Witness excused)

THE COURT: All right. I think we could take on another witness. Let's try that.

MR. BOIES: Your Honor, our next witness that we call is Mr. Larry Page.

THE COURT: All right.

MR. JACOBS: May I retrieve an exhibit, Your Honor, from the witness stand?

THE COURT: Yes, you may.

Now, let me be clear. This is Mr. Page of Google?

MR. BOIES: Calling him as an adverse witness, Your Honor.

THE COURT: So I want to be clear that --

MR. VAN NEST: I understand.

THE COURT: -- he's on cross-examination now,

Mr. Van Nest, so that our house rules now apply.

All right. Mr. Page will come forward.

Welcome, sir. Please stand in the witness box and raise your right hand.

LARRY PAGE,

called as a witness for the Plaintiff herein, having been first duly sworn, was examined and testified as follows:

THE WITNESS: I do.

THE COURT: Welcome. Please have a seat. This microphone moves around. Do you see how I'm moving mine? You've got to point it this close. Say your name.

THE WITNESS: I'm Lawrence Page.

THE COURT: Perfect.

Counsel.

CROSS EXAMINATION

BY MR. BOIES:

Q. Good afternoon, Mr. Page.

A. Good afternoon.

Q. You are currently the chief executive officer of Google; are you not, sir?

A. That's correct.

Q. And you are one of the founders of Google?

A. That's right.

Q. And since the founding of Google in 1998, you have been one of the top three officers of Google, correct?

A. That's correct.

Q. And you are either the largest or one of the two largest shareholders of Google, correct?

A. That's correct.

Q. I'd like to direct your attention to the period of 2005. And, in 2005, Google was considering acquiring Android, correct?

A. That is correct.

Q. And Android had not been initially developed by Google, but it had been developed by another small company, correct?

A. Android was a startup that we were working on acquiring, yes.

Q. And when you say "a startup," you mean a separate company?

A. Yeah, a small, a small company.

Q. And who was in charge at Google of the Android project?

A. Well, once we acquired the company Android, was Andy Rubin.

MR. BOIES: May I approach, Your Honor, with Exhibit 1?

THE COURT: Yes. Did you say "1"?

MR. BOIES: Trial Exhibit 1.

THE COURT: 1. All right. Go right ahead.

BY MR. BOIES:

Q. This is a document you are familiar with, correct, sir?

A. Yes.

Q. And --

MR. BOIES: Your Honor, I would offer this exhibit.

THE COURT: Any objection to number 1?

MR. VAN NEST: No objection, Your Honor.

THE COURT: Number 1 is in. Please publish it, if you wish.

(Trial Exhibit 1 received in evidence.)

MR. BOIES: Publish it.

(Document displayed.)

BY MR. BOIES:

Q. And this was prepared on or about July 26, 2005, correct?

A. That is correct.

Q. And this was prepared for a discussion of Key Strategic Decisions Around the Open Source Android Project, correct?

A. I think this presentation was about Android, and that's the title of it, yes.

Q. And you received this at or about the time it was prepared, correct, sir?

A. I believe so, yes.

Q. And who prepared this?

A. Looks like it was prepared by the Android team.

Q. And that would have included who -- who would have been the head of that Android team?

A. As I already mentioned, that would be Andy Rubin as the head of the team.

Q. Now, let me -- let me direct your attention to page 8 of 10 of this exhibit. It says at the top "Why Java?" Do you see that?

A. Yes.

Q. And it begins, "Carriers require it." Do you see that?

A. Yes, I do.

Q. And the carriers there were the telecoms that provided cell service, correct?

A. Yeah. Like Verizon or T-Mobile, or something like that.

Q. And it also says under "Why Java?" "Existing pool of developers and applications." Do you see that?

A. Yeah, I see that.

Q. And that was an existing pool of developers to do Java programming, correct?

A. Yeah. Java programming is taught in schools and many places.

Q. Let me ask you to go to the next page, where it says -- do you see "Current Scenario"?

A. Yes.

Q. And the first item there says "Developing a clean room implementation of a JVM."

And that stands for Java virtual machine, correct, sir?

A. Yes.

Q. And a clean room would be a project that had in it people

that did not reference, use, know the intellectual property of somebody else, correct?

A. Uhm, I think a clean room, as I understand this, would be a process that was used to develop software where, you know, you carefully control the information you use to do that.

Q. And in carefully controlling the information that you use to do that, you understand that you cannot use people to do that that know and are referencing somebody else's intellectual property, correct, sir?

A. I think in a clean room, obviously -- I assume you're referring to developing a clean room implementation of Java programming language, or free Java, which obviously you would need to use a description of that in order to develop, and there's nothing wrong with that. You carefully control any proprietary technology that was used to develop anything.

Q. I just want to be sure that the question and answer are meeting when we talk about "carefully control."

You understood that you might be able and you were able to use something that was publicly available in the clean room, but that you could not use somebody else's copyrighted material, and you could not use developers who referenced or knew somebody else's copyrighted material. Correct, sir?

A. Uhm, I'm not an expert on what we did in our clean room. I think we did nothing wrong, and I don't know details about what -- what you use or not, whether it was copyrighted or not,

or whatever.

Q. Well, sir, you know what a clean room is; don't you?

THE COURT: Mr. Boies, you can pursue this, but be mindful of the -- there are decisions on reverse engineering and fair use that if you're going to ask a legal question, which you are, you ought to build that into it.

Now, if you're just asking what he understood and how they set up their clean room --

MR. BOIES: Yes, Your Honor.

THE COURT: -- and what the guidelines were, that's okay. You can do that. But you're coming very close to saying -- you telling the jury what is copyrightable, what is not copyrightable, when fair use is okay, when reverse engineering is okay. And that's my job, not your job. So be mindful of the -- you're treading very close to getting into issues of law. Thank you.

MR. BOIES: Thank you, Your Honor.

BY MR. BOIES:

Q. Let me try to be very clear what I'm saying.

Within the software industry, there is a recognized concept of what a clean room is. Not a legal concept, but something that is recognized within the software industry. Correct, sir?

A. Yeah. I have no reason to think that we did anything that wasn't in accordance with that.

Q. Well, let me try to proceed maybe in order, first to get what you understand the clean room to be, and then try to get whether you proceeded in accordance with that.

First, let's talk about what you understand a clean room to be in the common industry practice of the software industry.

And do you understand that in the common industry practice of the software industry a clean room means that you are using people to develop software that do not have access to the proprietary intellectual property of other people?

I think that's a yes or no question.

A. Yes.

Q. Okay. Now, in terms of the clean room implementation that Google had, what steps did Google take to make sure that the people who participated did not have access to Sun and later Oracle's intellectual property?

A. I'm not familiar with the steps that were taken. Like I said, I have no -- I have no knowledge of any issues around that.

Q. When you say you have no knowledge of what steps were taken, did you, in your capacity as a high executive at Google, ever make any effort to determine what those standards were?

A. No. Nor did I have any reason to.

Q. Whose responsibility was it to find that out? Was that Mr. Rubin?

A. I assume it would be Mr. Rubin as head of the project, yeah.

Q. Now, Mr. Rubin reported to you with respect to this project, correct, sir?

A. Uhm, actually, I don't think he reported to me. He probably reported to another executive.

Q. He reported to you as well as other executives on this project, correct, sir?

A. Sorry, I was referring to just formal reporting.

But, you know, occasionally I would talk to him about it, so I had some knowledge about what was going on, yeah. I just didn't mean -- he didn't have a formal reporting relationship to me.

Q. And is it your testimony that you never inquired of him as to what procedures or standards they were using to make sure that the Google developers did not have access to Sun's intellectual property?

A. Yeah, I don't recall ever talking to him about that.

Q. Now, you know now that some of the developers did have access to Sun's intellectual property, correct, sir?

A. I don't know anything about that.

Q. Is it your testimony that you are unaware that certain lines of code in Android were copied, symbol for symbol, literally copied from Sun's intellectual property?

A. I know there's some disputes about a few files. And, you

know, I know that Mr. Ellison told me -- when I met with him in his house, he told me that they had some, you know, copied material in Android. And I disputed that with him. And he said he would send me that copied material, and he never did. Probably because it wasn't very strong evidence.

So -- and I'm not familiar with any of the other detail around that.

Q. Did you ever ask anybody in Google whether some lines of code had just actually been copied symbol for symbol, exact, literal copying, from Sun's intellectual property?

A. I don't recall asking anybody that.

Q. And as you sit here today, it is your testimony that no one at Google -- leave Mr. Ellison aside -- no one at Google has ever told you that Android includes some literal lines of code that were copied from Sun's intellectual property?

MR. VAN NEST: Objection, Your Honor.

BY MR. BOIES:

Q. Is that your testimony?

MR. VAN NEST: Privilege.

THE WITNESS: I already testified --

THE COURT: Wait, wait, wait.

Well, you may exclude from your answer a lawyer representing Google. But, otherwise, it is a proper question whether anyone else ever told you that or you had a conversation with anyone other than a lawyer on that subject at

Google.

So, please, answer the question.

THE WITNESS: Yeah, I don't recall anybody else.

BY MR. BOIES:

Q. Now, if you discovered that Android included some lines of code that had been literally copied from Sun's intellectual property, is that something that you would feel was a violation of Google policy?

A. Uhm, that seems like a hypothetical question. I imagine it would depend on the exact circumstances of that, but, in general, yeah, we would take that seriously.

Q. Is there any circumstance that you can think of that's consistent with a clean room where you could have literal line-for-line copying from Sun's intellectual property and be consistent with what Google's standards for a clean room are?

A. Again, it's hard to answer a hypothetical, but I -- I see there's no reason that wouldn't be possible.

Certainly, if it were very short, for example, there might only be -- you know, there's only one way to write one plus one, for example. So if it's a very short piece of code, it might appear to be the same but not actually be copied.

Q. Yes, I understand something might appear to be the same.

THE COURT: I think we're getting into hypotheticals. This witness says he has no direct knowledge. And now it's become an argument. And I think we should move to something

else. He says he has no information on the subject. So it's best to move to something new.

MR. BOIES: Yes, Your Honor. Are you finished?

THE COURT: Do what?

MR. BOIES: I just want to be sure the Court had finished.

THE COURT: I had. I think it's become an argument.

MR. BOIES: Okay.

THE COURT: You made your point.

MR. BOIES: Thank you, Your Honor.

BY MR. BOIES:

Q. Let me turn back to Exhibit 1, in the same page that we were looking on. Do you see that the third bullet there is "Must take license from Sun"?

A. Yeah, I see that.

Q. And that's what Mr. Rubin and his team told you in July of 2005, correct, sir?

A. That's what's in the slides here.

Q. And then down under proposal it says:

"Proposal: Google/Android, with support from Tim Lindholm, negotiates the first OSS J2ME JVM license with Sun."

Do you see that?

A. Yeah.

Q. And you know what OSS refers to, do you not, sir?

A. Open source software, I would assume.
Q. And J2ME was one of the versions of Java, correct?
A. Yeah, I think that was the mobile version.
Q. "M" actually stands for Micro Edition, does it not, sir?
A. I'm not familiar -- I'm not sure.
Q. And then JVM is the Java virtual machine, correct, sir?
A. That's my understanding, yeah.
Q. And you know who Mr. Lindholm is, don't you, sir?
A. I think -- I don't know him well. I think he's somebody who works for Google.
Q. Beyond thinking that he's somebody that works for Google, he's somebody who you have asked questions of and give instructions to, is that not --
A. I don't recall him particularly.
Q. Do you recall generally?
A. Uhm, I, don't remember him well. If I met him, I assume I would know who he was.
Q. Do you remember giving him some instructions and asking him some questions?
A. Not particularly, no.
Q. Do you recall asking him to do some work in connection with Java?
A. I've already said I don't recall him.
Q. Let me ask you to look at Trial Exhibit 2, if I could, sir.

THE COURT: Use the time as you wish, but in four minutes we will break. So, go ahead. We have four minutes.

MR. BOIES: We could break --

THE COURT: Is this a good point?

MR. BOIES: This is a good time to break.

THE COURT: This is a good time to break. Then this is the time we're going to break.

All right. The jury will remember the admonition. I'm going to repeat it. No research about the case. No looking at news stories about the case or reading stories or listening to stories. No talking with friends, neighbors, loved ones about the case, nor with each other about the case. It will be your duty to do that soon enough, but not yet.

Thank you for being on time this morning. We were able to start earlier because you were on time. And, hopefully, we can continue to do that. So your close attention is noted and most appreciated. We'll see you back here tomorrow. Thank you.

THE CLERK: All rise.

(Jury out at 12:57 p.m.)

THE COURT: All right. Everyone else, be seated.

Mr. Page, you're free to go, but you're under a direct order not to talk with any counsel or anyone else about your testimony. All right?

THE WITNESS: All right.

THE COURT: All right. Thank you.

I have a few questions for counsel that I would like to ask. And I know these -- these are not points yet in evidence.

Mr. Boies, you can stay right there because you may know the answer to these questions. How many methods -- I'm thinking back to the file, Mr. Van Nest's file box thing which helped me understand the problem. Maybe you disagree with that.

How many methods are in the 37 APIs? Are we talking about thousands, or a couple hundred, or just 37 altogether? Sounded to me like there were more than 37.

MR. JACOBS: According to Dr. Astrachan, Google's expert, there are 8,693 methods in the 37 packages.

THE COURT: Are those the exact same ones -- putting aside the source code -- that are in the Oracle version? Or are there differences in those methods?

MR. JACOBS: I cannot represent to you as to all 8,693. These are the ones that he identified -- I'm sorry. I want to be -- I may be misunderstanding my notes here. Let me take just a minute.

I'm sorry, Your Honor. I want to be sure we have this right, so we'll get this to you.

THE COURT: All right. I would like to know what differences there are. I know there's differences in source

code, but I would like to know if the methods are the same ones, whether there have been any added or subtracted.

MR. JACOBS: You bet.

THE COURT: And then, in addition, next question is -- this has all got to be proven up in due course. But I am thinking about these problems.

Did Sun or did third parties or did some combination develop these 8,000 -- I'm talking about the ones that were now on the -- on the Sun side of the equation, were they all developed by Sun, or were some of these done by third parties? Contributed to the file.

MR. JACOBS: I'll have to get the answer to that, as well.

The way it works, Your Honor, is if people participate in the development of the specifications, the spec lead obtains ownership of the intellectual property.

And as to these 37 packages, there's no question about ownership of the --

THE COURT: No, no --

MR. JACOBS: -- specification.

THE COURT: I am not questioning ownership.

I am wondering how they got put into the file in the first place. Can a third party develop something and say, I want this in the file? Or can a third party say, I want this in the file, and just get it in? Or does somebody have to

approve it, or does Sun have to approve it, or maybe third parties having nothing to do with developing those methods?

MR. JACOBS: So I think that would probably be best explained by someone who has actually done this, Your Honor.

THE COURT: All right. Next question. In one of the things you gave me earlier, both sides referred to something called a declaration, declaration of API elements.

What is a declaration? That's in your statement of issues regarding copyright that I've been studying, trying to -- what is the -- what do you mean by "declaration"?

MR. JACOBS: We'll get you a formal definition of the declaration in the Java class structure, but it is roughly the first line of the Application Programming Interface and the corresponding class library.

THE COURT: I'm going to let you over there have your response if you have a different response.

My last question is, whether something is a derivative work is that a question of fact for the jury?

MR. JACOBS: I don't think so, if there's an acknowledgment of substantial similarity. I think the -- if there was a dispute about substantial similarity, then there could be a fact question.

THE COURT: All right. Let's hear what the other side says. Take them in any order you wish.

MR. KWUN: Your Honor, Michael Kwun for Google.

Starting with the last one, the derivative work question, I think, depends a little bit on what exactly the claim is.

To the extent that the claim is that the -- that it's a derivative work because it has the same structure, selection and organization, then if the Court concludes that the structure, selection and organization isn't copyrightable, it can't be a derivative work. Because to be a derivative work, it must be derived from something that's protectable.

To be perfectly frank --

THE COURT: I understand that -- no. Let me give you a different example.

As I read these write-ups that you've given me, one of the contentions out of several, one of the contentions is that even though Google says it used its own source code -- and looks like, from what I've heard so far, that with the exception of some lines of code, that's true. However, the come back by Oracle to that is, well, nonetheless, it's a derivative work because Google had the specification.

By that I mean the plain English language manual that said what the inputs were, what the outputs were, and which other files and methods were borrowed along the way, and that, therefore, even though different source code -- it was written in different source code, the 37 APIs and the 8,000 methods are derivative works of the plain English description of those

files, and so forth.

MR. KWUN: Yes, Your Honor. With that formulation, I think that it just runs completely afoul of 102(b).

I mean, you just can't have something be a derivative work because Google's code does what they describe. I mean, that is -- that's -- I mean, that's directly contrary to *Baker v. Selden*.

THE COURT: All right. Maybe. Contrary to which case?

MR. KWUN: *Baker v. Selden*. That's the 1879 --

THE COURT: Oh. 18- --

MR. KWUN: I mean, it would be contrary to many other cases, as well.

THE COURT: All right. Assume that you're wrong on that for the sake of argument. Is the question of derivative work something that is a jury question or a judge question?

MR. KWUN: It could certainly be a jury question. There are a number of legal questions that are entangled with it that might well render it -- there are no facts left to the jury.

THE COURT: Well, before we get to the point that this goes to the jury, you lawyers have to take a gutsy position and either say yes, it is, or, no, it's not.

But, okay. I want you to know it's on my mind. And sooner or later you've got to, you know, make your mind up.

What's the answer --

MR. VAN NEST: We will, Your Honor. We'll address --

THE COURT: All right. Nobody -- everyone is going to send me a postcard on these issues.

(Laughter)

THE COURT: I need something -- what is a declaration?

MR. KWUN: So a declaration, there is a more -- a more formally defined term. It's called a method signature. And I believe that at least as a -- for lawyers' purposes, that the declaration would be the same as the method signature.

I don't know if I have a disagreement with that from the Oracle side or not.

THE COURT: Look. Here's your document right here. Paragraph 2. It says that, "Oracle accuses the following: Number 2, the declarations of the API elements," blah blah blah.

What did you mean when you used the word "declaration"? It's your own word.

MR. KWUN: We meant a method signature. We also meant to try to come as close as we could to their language.

But a method signature means the name of the method, the variable type that it returns, and the variable types and order of the parameters it accepts. And it would also include the exceptions that it throws, the errors that it throws.

THE COURT: What would one of these items look like?

MR. KWUN: So we've talked about in the past the max method, which identifies the maximum of two numbers.

The method signature for one version of the max method would be int, which is to say that it returns an integer, max, which is the name of the method, and then parentheses int,int. So that means that it take two parameters that are both integers.

Depending on who you ask, some people will -- would actually identify it as max open parentheses int A comma int B closed parentheses.

THE COURT: Do you know anything about any other programming languages?

MR. KWUN: Sure, some.

(Laughter)

THE COURT: Now, do you know anything about a language called QBasic, that's very simple.

MR. KWUN: I know something about Basic, which QBasic --

THE COURT: C or C+. May be the same way.

But take a statement called or command called ABS parentheses closed parentheses with something in there. It always returns the positive, the absolute value of whatever that number is. Like ABS paren minus 19 closed paren would return 19.

MR. KWUN: Yes, Your Honor.

THE COURT: All right. So using that as an example, is ABS an API? Is it a method? Is it a library? Or is it a class? Or is it none of the above?

MR. KWUN: It would be a method; although, it might be called a function in QBasic. But those terms are semantically equivalent. So it would be a method.

I would say that it would also be an API. It's an API one can use to access the absolute value functionality. It is not a class.

THE COURT: All right. So what is, then, a class versus the method? Is a method simply just a collection of methods? Is that what a class is?

MR. KWUN: It can be a collection of -- it can be a collection of methods. It can also contain fields that store data.

So, classes come from a particular type of program called object-oriented programming. And in object-oriented programming you have objects. And objects are created out of a template of sorts called a class. And the class has a way of organizing data that you might need for that type of object.

So, for example, if you had an object that described a bicycle, you might have some data fields that say what color the bike is, what size the tires are, and so forth.

So it can have data in the class. And then it can

also have methods that are going to be useful to do things to the data that it stored in the class.

THE COURT: When you use object-oriented programming, the word "object" there is not the same thing as object code; is it?

MR. KWUN: No.

THE COURT: Those are two different uses of the word "object"?

MR. KWUN: Yes.

THE COURT: Are we going to have witnesses who are going to be good at explaining all of this?

MR. JACOBS: Promise, Your Honor.

THE COURT: Good.

How about Google, will you have a witness who --

MR. KWUN: We have a expert witness who can answer these sorts of questions, yes.

THE COURT: How many methods are in all of these 37 APIs, in your view?

MR. KWUN: I don't know the precise number. It's somewhere in the thousands.

THE COURT: Now, if we went to some other company and looked at their APIs, would they be organized roughly the same way?

MR. KWUN: Well, the APIs are there to accomplish a purpose. So if they are accomplishing some other purpose, they

would have no resemblance to the APIs in this case.

THE COURT: All right. Well, I forgot to tell the jury that you were not being rude when you didn't talk to them. I will remember to do that tomorrow. Okay.

All right. Thank you for the help on those things. And anything more today before we break?

MR. JACOBS: No, Your Honor.

MR. BOIES: No, Your Honor.

THE COURT: I do have my criminal calendar in 45 minutes, so I'll need the tables. You can leave all your other stuff over there on the side of the room.

MR. VAN NEST: Thank you, Your Honor.

MS. ANDERSON: Thank you.

(At 1:12 p.m. the proceedings were adjourned until Wednesday, April 18, 2012, at 7:30 a.m.)

- - - - -

I N D E X

– – –

E X H I B I T S

TRIAL EXHIBITS	IDEN	VOL.	EVID	VOL.
1			407	2
7			297	2
401			297	2
610.1			370	2
2041			316	2
2042			321	2
2043			324	2
2044			329	2
2362			350	2
2939			314	2

– – –

CERTIFICATE OF REPORTERS

We, KATHERINE POWELL SULLIVAN and DEBRA L. PAS, Official Reporters for the United States Court, Northern District of California, hereby certify that the foregoing proceedings in C 10-3561 WHA, **Oracle America, Inc., vs. Google, Inc.**, were reported by us, certified shorthand reporters, and were thereafter transcribed under our direction into typewriting; that the foregoing is a full, complete and true record of said proceedings at the time of filing.

/s/ Katherine Powell Sullivan

Katherine Powell Sullivan, CSR #5812, RPR, CRR
U.S. Court Reporter

/s/ Debra L. Pas

Debra L. Pas, CSR #11916, RMR CRR

Tuesday, April 17, 2012