>> The classes we've been using are part of the Java Class Library, and it's part of the JDK, which comes with Java. And it's documented in the Java API, or Application Programming Interface. These classes are not part of the Java language, but we rely on them heavily. You can think of them as an extension to the language.

We've already used the system class scanner and string classes. And they're all part of the Java Class Libraries. Other libraries can be obtained through a third party vendors, or you can create your own class library. But buy and large those in the Java Class Library are sufficient for lots of work.

They're organized as packages. And here's some examples. Here's a java.lang for general support. Here's java.applet for creating applets. The next two, java.a awt, and javax.swing, are what we use to do GUI development, as in graphical user interfaces. And there's java.net, java.util, for utilities, and of course we've already used the scanner class from that.

Now these are described in detail in the Java API. And you can just Google Java API, or go to the website. You can also get to it from the jGRASP help menu by clicking Help and then Java API. Let's take a look at the API. The top page will look like this.

You'll see packages on the on the left here and then you can see lots of packages and below here are the classes. There are literally thousands and thousands of classes and let me see if I can get down here to string, which is where we're at. Let's see here.

Here it comes. So here's the string class, and I've actually positioned this on it already. If you look at this, up at the top, you see a summary of fields, constructors, methods and so on, and notice this is part of java.lang. This is the package that it comes in.

The name of the class is String. In fact its full, fully qualified name is java.lang.String. And then if we click on methods, this takes us to the complete list of String methods. Notice the first one is charAt. It takes an int index and returns the char value at the specified index.

And notice here is the return type on the, on the left side here of all these is the return type. If you concatenate a string you get back a string as you see. And looking down through here let's see what else we got. Here's index of, where did it go?

Here's index of. Here, we send it a character, and we get back an int. So, again, the return type, easy to spot here, and so on. Here's length, gives back an int as well. Let's see some others here. Two lower case we get back a string, replace, trim.

Trim is another one we've used here, you get back a string, and so on. So plan to make frequent use of the API. Anytime you need to use a method for a particular object, that you're using, you can, certainly go here and and look it up. So, let's talk about the import declaration.

When you want to use a class from a package, you can use it fully qualified, (no import statement required). Here's an example. We've been using the scanner class, but if you wanted to not import it you could say java.util.Scanner, and then the varium name, we're using scan here, new java.util.Scanner.

Notice we're fully qualifying it on both sides. So it's a little bit longer to write that way. If you want to write just the class name. You just import the class up at the top. Like we've done. Import java.util.Scanner. And the data in the method you can create a a scanner like we are doing here with this single statement.

To import all classes in a package you can use the asterisk or wild or wildcard character, also called star, and it might look like this, import java.util.*;. And this would give your program access to all of the classes in there, sort of with a single line. Generally this is not a good practice.

In most cases it's best to name each class used for documentation purposes and most style-checkers have rules that that actually don't allow you to use the the wildcard. Now you might wonder why we can use the String class without importing its package, recall it's in java.lang. Well, the java.lang, which is general support is automatically imported.

We don't have to import it. In fact if up at the top we said import java.lang.* it would be redundant because we already have access to all of those. Scanner class on the other hand is not in java.lang so we have to actually import the java.util package.