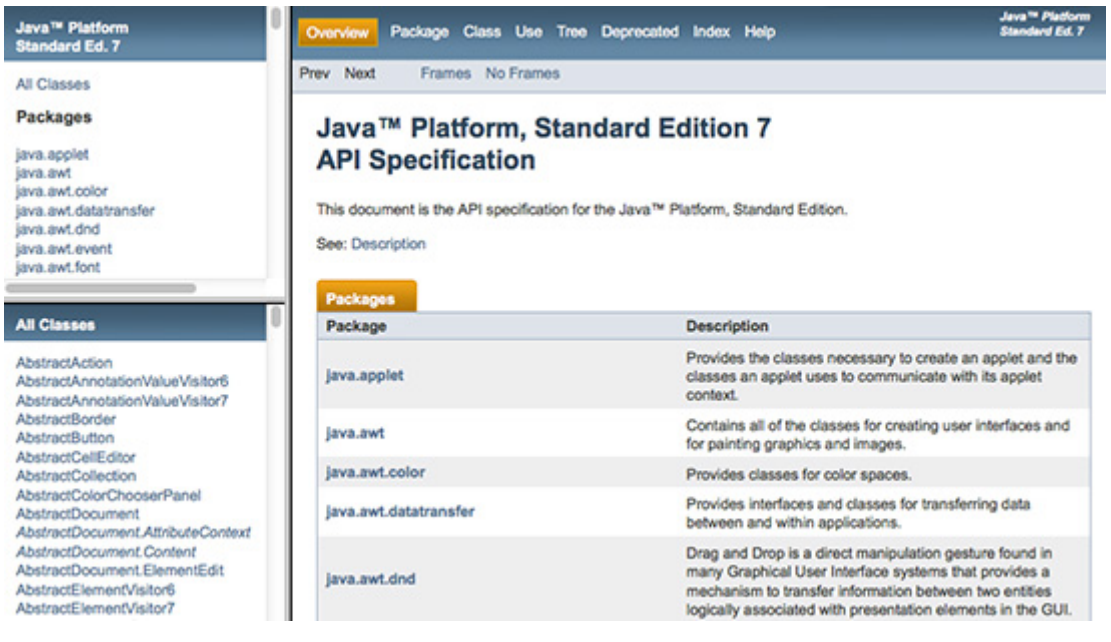
Chapter 3: The Java API

The Java API

Java's API, or Application Program Interface, is where you can get all the information you need to use the classes that come with Java. (As you learned back in Lesson 2, a class is like a blueprint that you can use to create different objects.)

The amount of information can be daunting at first, so we're going to look at the different parts of the API, and then we'll examine the information that's available for each class. We'll dig into the API for a couple of frequently used classes so we can see how to use the information that's there.

Let's look at the API home page and what it shows us. I've included the link in the Supplementary Material in case you want to follow along on the site rather than using the images I've provided in this chapter.



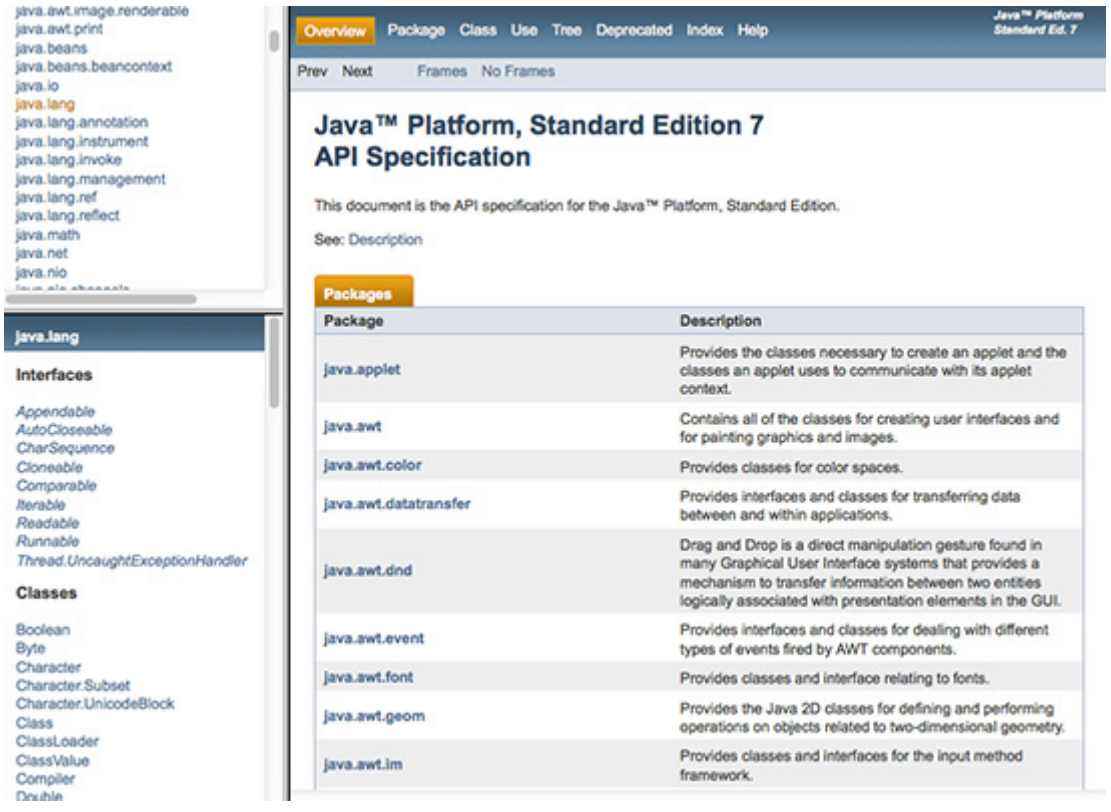
The Java API home page

When the API opens, you'll see three panels in the screen. The top-left panel shows a list of Java *packages*. In Java libraries, a package is a group of related classes, subpackages, or both. This list shows several: java.applet, java.awt, java.awt.color (a subpackage of java.awt), and so on. Over 200 packages deal with input and output (java.io), databases (java.sql), user interface components (java.awt and javax.swing), and many more aspects of Java. Clicking a package name will display the list of classes in that package in the lower-left panel.

On this screen, the lower-left panel shows a list of all the classes in the selected package. The opening view shows all the classes in all the packages. To get back to that view, just click **All Classes** in the upper-left panel.

The larger panel on the right shows the details of any class selected in the lower-left panel. The opening view in this larger panel is another list of all the packages, along with a brief description of the purpose of each.

Just to see how some of this works, scroll down in the upper-left package panel until you see the **java.lang** package, and then click it. You'll see the lower-left panel change to this:



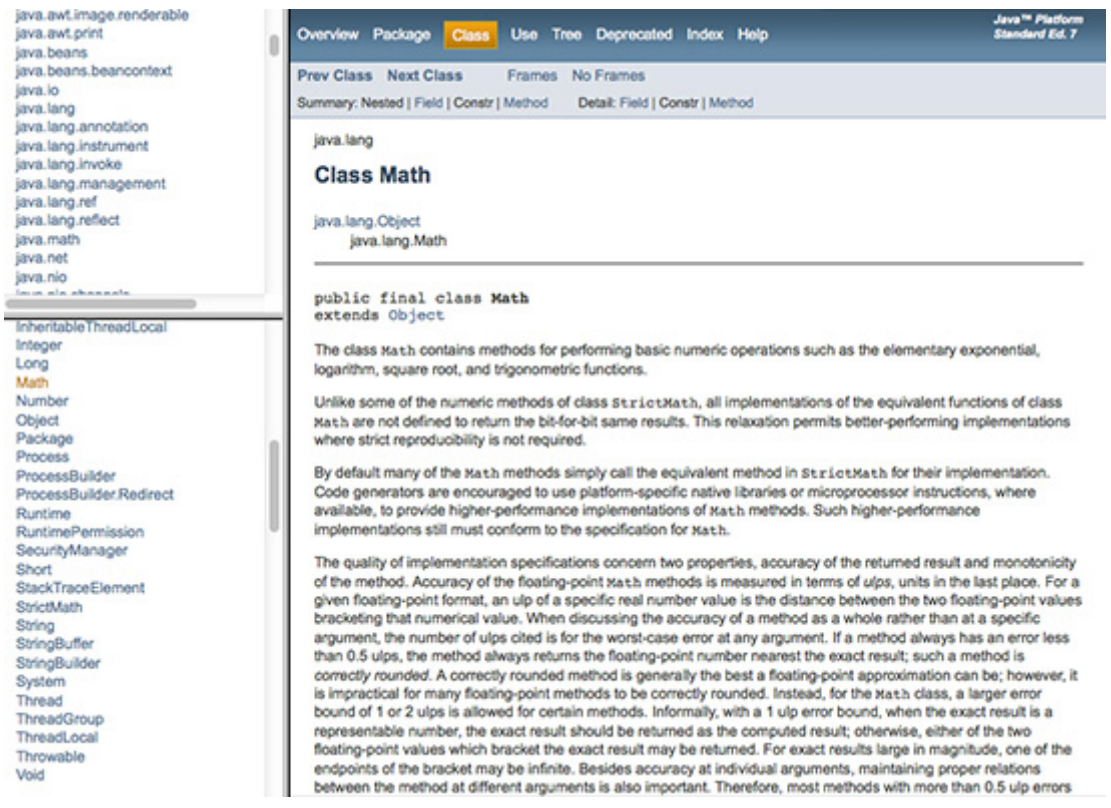
contents list in the lower-left panel

The package contents list has several categories, including Interfaces, Classes, Enums, Exceptions, and a few others. We don't have time or space to get into all the categories or classes here, but you can find them here.

The java.lang package automatically loads whenever you compile a Java class or program. These are some of the most commonly used Java classes, like String, System, and Math, along with 25 or so others.

Let's see what information's available about a single class. To do that, we'll look at a couple of classes we've used already.

Scroll down in the lower-left panel until you see the **Math** class, and click on its name. The Math class documentation in the right-hand panel will open.



The java.lang.Math class documentation

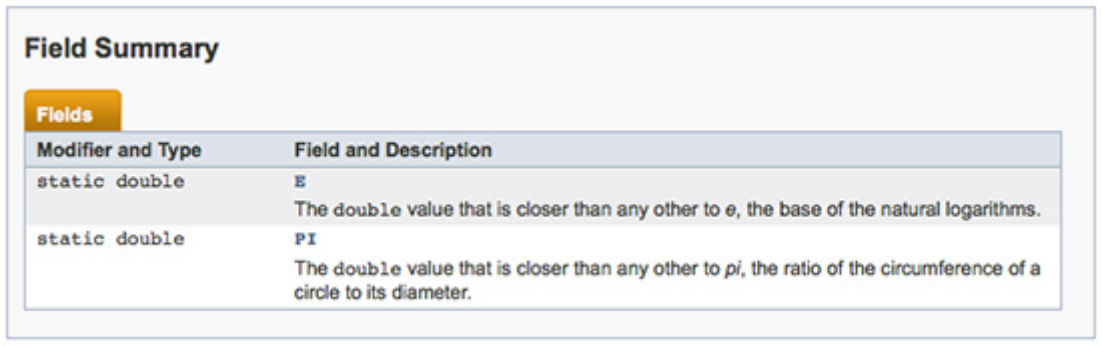
The first line in that right-hand panel, "java.lang", tells us what package the class is from. The second line is the class name.

The next two lines show the *inheritance hierarchy* for the class. We discussed about inherited attributes and methods in Lesson 8. Ultimately every class in Java inherits from Object, but some classes have several other classes in between. In this case, Object is the only class that Math inherits from. If you ever need to know what classes any class inherits from, this is where to find out.

The next few paragraphs are a general description of the Math class and its use. Some classes have only a short description, if any. But others have quite a bit, including examples of uses of the class and its methods.

Finding the Field Summary

If you scroll down in the large panel, you'll get to detailed interface information. It begins with a *field summary*, which is a section that summarizes public fields of the class. The Math class has two public fields: E and PI. (This is one of the rare occasions that data fields are declared with the public access modifier.) Their summaries look like this:



Field Summary

Fields

Modifier and Type	Field and Description
`static double`	`E` The `double` value that is closer than any other to *e*, the base of the natural logarithms.
`static double`	`PI` The `double` value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

java.lang.Math public data fields

These two fields represent important mathematical values that you might use if you perform calculations involving circles (PI) or logarithms (E). The documentation for each field shows its type, its name, and a brief description of what it is. Both of these values are public, static, and double. They're also constants, so their values can't change. We use them in our programs by using the class name followed by the field name, like this: Math.E and Math.PI. You can click on each name to see details about that field.

Method Summary

Scroll down again to the method summary section. It looks like this for Math:

Method Summary

Methods

Modifier and Type	Method and Description
`static double`	`abs(double a)` Returns the absolute value of a `double` value.
`static float`	`abs(float a)` Returns the absolute value of a `float` value.
`static int`	`abs(int a)` Returns the absolute value of an `int` value.
`static long`	`abs(long a)` Returns the absolute value of a `long` value.
`static double`	`acos(double a)` Returns the arc cosine of a value; the returned angle is in the range 0.0 through *pi*.
`static double`	`asin(double a)` Returns the arc sine of a value; the returned angle is in the range -*pi*/2 through *pi*/2.
`static double`	`atan(double a)` Returns the arc tangent of a value; the returned angle is in the range -*pi*/2 through *pi*/2.
`static double`	`atan2(double y, double x)` Returns the angle *theta* from the conversion of rectangular coordinates (`x`, `y`) to polar coordinates (r, *theta*).
`static double`	`cbrt(double a)` Returns the cube root of a `double` value.
`static double`	`ceil(double a)` Returns the smallest (closest to negative infinity) `double` value that is greater than or equal to the argument and is equal to a mathematical integer.

java.lang.Math public methods

These are the first few of a long list of commonly used mathematical methods in the Math class. The list shows each method's type (the data type it returns), its name, its parameter list, and a brief description of what the method does. In the Math class, they're all static, so we can call them by prefixing their names with the class name. For example, Math.abs(-5) returns the absolute value of −5, which is 5. Math.cbrt(8) returns the cube root of 8, which is 2.

Just as it was in the field summary, each method's name is a link that will take you to details about the method if you click it.

The method summary lists four abs() methods listed, one for each parameter list the class provides. The authors overloaded that method for the four most commonly used numeric types. The reason for overloading it is so that the method's result can be the same data type as the argument it's given. That way, if I ask for the absolute value of an int, I get an int result. If I ask for the absolute value of a double, I get a double result, and so on.

Other methods, like cbrt(), have only one version. That's because the cube root of most integers isn't an integer, and the method returns the most accurate answer it can, which is always a double type.

Let's continue with our scrolling . . .

Field and Method Details

The last two sections in the Math class documentation are detailed descriptions about its fields and methods. You'd get to the same information by clicking on the field name or method name in the summary section. Here are the details on the field PI and the integer version of the abs() method:

PI

```
public static final double PI
```

The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

See Also:

Constant Field Values

abs

```
public static int abs(int a)
```

Returns the absolute value of an `int` value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Note that if the argument is equal to the value of `Integer.MIN_VALUE`, the most negative representable `int` value, the result is that same value, which is negative.

Parameters:

`a` - the argument whose absolute value is to be determined

Returns:

the absolute value of the argument.

java.lang.Math field and method details

Each of these sections describes the field or method in detail, starting with the exact details of how it's declared. For methods, there are also details about parameters and the value the method returns. If the method can generate any exceptions, this section lists them after the return value. If any related topics exist in the API, this section provides a "See Also" section that links to them.

Both of these sections link to additional information you might need. The section on PI has a link to a list of Java constant values that will give you the actual value in the field, 3.141592653589793. The section on abs() has a link to the value of the smallest integer that an int type can contain: −2,147,483,648.

That's an overview of the documentation available on the Math class. Let's take a break from reading about it and actually use it. (I know I can only absorb a limited amount at once.)

Now You Try

Try writing a quick program that uses a Math method you haven't used before, and then use printf() to format its output. For example, you could write a program that calculates the cube root of the first 25 even numbers greater than zero and displays all of them with five decimal places.

If you'd like to do more, feel free to come up with your own ideas for a short program.

My solution is "behind" this link. Take a look at it after you write your own program. As always, free to share questions with fellow students in the Discussion Area.

Hide answer

```
/**
 * MathFormatting provides a solution to an exercise in the lesson.
 *
 * @author Merrill Hall
 * @version 1.0
 */
public class MathFormatting {
    /**
     * The main() method calculates the cube root of the first 25
     * positive integers and formats them to display 5 decimal
     * places.
     */
    public static void main(String[] args) {
        int i = 2;

        while (i <= 50) {
            System.out.printf("The cube root of %2d is %7.5f%n",
                              i, Math.cbrt(i));
            i += 2;
        }
    }
}
```

In the next chapter, we'll look at possibly the most widely used class in Java: the String class.