CSC 1051 - Lab 6 - Extra API Exercise

Objectives:
Learn to use the Java API.

Introduction:
Java derives much of its power from the many classes already defined in the Java API. But how are we ever to learn and use these classes if we don’t know about them? Any text on Java can only begin to cover these classes and the methods defined in them. For a complete listing of these classes and methods you will need to visit Java 2 Platform Standard Edition 6.0 API Specification

Although the information covered in the textbook is sufficient to complete all of the programming and lab assignments for this course, you may find yourself wishing for a “better” class or method, or just more information on a known class or method. The JavaTM Platform, Standard Edition, v 6.0 API Specification website is the place to find that information!

All class definitions are found in the Java API Specifications. API stands for application programming interfaces and is more simply a set of existing “building blocks” for programmers to use to develop programs. The API is divided into packages. Packages contain classes. Classes contain methods. Methods contain Java code. We use methods in our Java programs.

Assignment:
Access the Java API web site, using the link above or navigate to it from the Oracle website.

The API Specifications page is divided into 3 sections. The upper left-hand section is used to navigate to different packages (collections of classes). Below this section is a listing of all Java classes in alphabetical order. The largest section of the page displays details about a selected package or class. At present (before selecting a class or package), all Java packages are listed.

Scroll down the main display section of the page until you find the java.lang package. What does it provide?

________________________________________________________________
________________________________________________________________

The java.lang package is automatically provided/imported for all Java programs.
Find the `java.util` package. Can you find a class provided by `java.util` that we learned about in Chapter 3?

Clicking on any package will get a detailed description of the package. Click on `java.util`. This detailed description provides 5 summaries of items contained in this package. List the five summaries (i.e., the headings highlighted in blue):

________________________________________________________________

________________________________________________________________

________________________________________________________________

________________________________________________________________

For now, we are interested in the **Class Summary**. This summary lists the classes that are contained in the package. The left column contains the name of the class. Notice that all class names start with a capital letter. The right column contains the description of the class. Scroll down until you find the `Scanner` class. What does it contain?

________________________________________________________________

________________________________________________________________

Click on the `Scanner` class. You will get a detailed description of what is contained in the `Scanner` class. Notice that the package name - `java.util` - appears (in small print) above the class name. Scroll down a few pages to see the two summaries available for the `Scanner` class. What are they?

________________________________________________________________

________________________________________________________________

Scroll down to the **Method Summary**. The left column indicates the type of information the method will return. The right column contains the method name (underlined), the parameters (in parentheses) and a brief description of the method.
Examine the first method listed for the Scanner class. It is the `close()` method. The left column contains `void`, indicating that this particular method does not return anything. All methods have a return type, even if the return type is simply `void`. The right column tells us the name of the method is `close` and the empty () indicates that this method does not require any parameters to be used. The name of the method is located immediately before the open parenthesis. All methods require parentheses.

Based on this information, you could invoke this method using the programming statement `scan.close();` where `scan` is an already declared and initialized Scanner object.

Let’s look at another Scanner method. Locate the method `findInLine(String pattern)`. This method’s return type is a String. It requires 1 parameter in order to invoke this method. The parameter is of type String and is some kind of pattern. Note that there is also another version of this method that accepts a parameter of type Pattern (another class). We will focus on the version that has a String parameter. The definition tells us that this method “attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.”

Based on this information, you could invoke this method using the programming statement `String result = scan.findInLine("xx");` where `scan` is an already declared and initialized Scanner object. The variable `result` will then reference the String produced/returned by the method.

Click on the name of the Scanner method `findInLine(String pattern)`. This will provide you additional information about the method. Notice the line at the top of the page:

```
public String findInLine( String pattern)
```

This line is known as the method header. This is what we see in two columns on the previous page with the added word `public`. The word “public” indicates that this method is “publically assessible” so that we can use it. The return type follows and is a String. A method only ever retuns one type. The word located immediately before the parentheses is the name of the method. Everything listed inside of the parentheses are the parameter specifications.

Choose your browser’s back button to return to the Scanner class’s Method Summary. Let’s look at one more method of the Scanner class. To date, we have used the `nextInt()` method to capture integer input from the user. Locate the `nextInt()` method. This method is also listed twice. The first appearance of this method does not specify a parameter and the second appearance of the method does. Note that both `nextInt()` methods return an integer. If you have a Scanner object declared and initialized called `scan` and an integer declared and initialized called `num`, the `nextInt()` method could be invoked one of two ways:

```
int inputA = scan.nextInt();
int inputB = scan.nextInt( num );
```
(It might take some detective work to figure out exactly what the \texttt{num} parameter could possibly be indicating here. If you have time, write a little program to investigate this question. Otherwise, you can move on and it will not be a great loss—except to your curiosity!)

Ok, now let’s look at another class – the \texttt{String} class. To locate the \texttt{String} class, use the left hand alphabetical listing of classes. What package is the \texttt{String} class part of?

________________________________________________________________

Under the \texttt{String} class Method Summary, locate the \texttt{String} method \texttt{trim()}. For this method, provide the following:

Method return type: ________________________________

Required parameters for the method: ________________________________

Purpose of the method: __________________________________________

What would be displayed as a result of executing the following programming statements?

\begin{verbatim}
String fname = "Ben", lname = "Franklin";
System.out.println( fname + lname);
System.out.println( fname.trim() + lname);
\end{verbatim}

________________________________________________________________

________________________________________________________________

There are so many great methods to be used from the \texttt{String} class that you will surely return to this class’s API many times! But before you review more or the \texttt{String} methods, let’s take a look a look at a special type of class.

The \textbf{Math} class is a class that only contains static methods. First, locate the \texttt{Math} class. In which Java package can you find the \texttt{Math} class?

________________________________________________________________

Scroll down to the Method Summary section of the \texttt{Math} class. Examine the first method called \texttt{abs()}. The left column contains \texttt{static double}. The word \texttt{double} tells us that the return type of the method is \texttt{double}. But what does \texttt{static} mean? Static tells us that this method does not act on an object from the \texttt{Math}
class but that we can just call this method whenever needed. First, answer these
questions about abs():

Method return type:______________________________

Required parameters for the method:______________________________

Purpose of the method:
_________________________________________________________________

Because abs() is a static method, to invoke the method you would use the class
name and then the method. For example, executing
System.out.println( Math.abs( 2 * (10 - 35) );
would result in printing 50.

Review the Math method ceil() and answer these questions:

Method return type:______________________________

Required parameters for the method:______________________________

Purpose of the method:
_________________________________________________________________

Example of invoking the method to print something:
_________________________________________________________________

What gets printed?
_________________________________________________________________

What have you learned?
• The Java API is divided into packages
• Packages contain classes
• Class names start with a capital letter
• Classes contain methods
• The name of the method is directly to the left of the open parenthesis
• All methods require parenthesis
• Parameters are specified with a type followed by an identifier
• All methods have a return type
• The return type of the method is located directly to the left of the method
  name