
Using Objects from the Java API

In this laboratory you will:

1. Learn how to read a method header
2. Learn how to declare, create and use objects
3. Use a `javax.swing.JOptionPane` object as an alternate way to do Input/Output
4. Learn how to generate random numbers using a `java.util.Random` object
5. Explore using `java.lang.String` objects and their methods

In Session 1, we learned that a *class* is a template for an object in that it describes the object's characteristics and behaviors. Objects are built using the information in these templates. Thus, we say an *object* is an instance of a class. In this session, we are going to create and use objects defined by three of the classes in the Java API. They are `JOptionPane`, `Random` and `String`.

When using an object defined in the API we are essentially using a black box. For example, without knowing about or understanding the inner workings of a television, we do know how to turn it on or off, how to change the channel, and how to increase or decrease the volume. That is, a television is essentially a black box that provides us with a way to use it. Similarly, when we create an object defined in the Java API, we do not need to know how it works. We only need to know how to use it by invoking one of its methods. The header of a method tells us how to properly invoke the method.

Understanding a Method Header

To use an object, we need to know what `public` methods are found in its defining class. The header of a method tells us how to use the method. The form of a method header is

```
optionalAccessModifiers returnType methodName( optionalParameterList )
```

For example, the `Scanner` class has a method `nextLine`, whose header is

```
public String nextLine()
```

The *methodName* is `nextLine`. The *returnType* is `String`, meaning that this method returns a `String` that the program can then store or manipulate. This method has no *parameterList*, meaning that it does not need any additional information. The term `public` is an *accessModifier* that tells us that this method is available for the public to use. Thus, to use this method on an object named `scan` of type `Scanner` we write

```
line = scan.nextLine();
```

which assigns the returned `String` to the `String` variable `line`.

Another method that we've used is the `parseInt` method from the `Integer` class. The header for the method is

```
public static int parseInt(String s)
```

The method header tells us that the method `parseInt` has one *formal parameter*, `s`, of type `String`. Therefore, when the method is invoked it must be passed a `String` by placing either a `String` variable or `String` literal between the parentheses. The `String` that is passed is assigned to the formal parameter `s`. Inside the black box method, processing that uses `s` takes place, and when done, the method returns an `int` value to the user. Furthermore, the access modifier `public` tells us that this is one of the available methods, and the access modifier `static` tells us that the method can be invoked on the name of the class, `Integer`, instead of on an `Integer` object. When executed, the combination of statements

```
line = scan.nextLine();
num = Integer.parseInt(line);
```

passes the `String` variable `line` to the `parseInt` method and assigns the returned `int` value to the `int` variable `num`. The variable `line` is the *actual parameter* that is passed to the method and assigned to the formal parameter, `s`.

Looking at the code from the previous labs, the statement

```
num = Integer.parseInt(scan.nextLine());
```

first invokes the `nextLine` method on the `Scanner` object `scan`. The `String` that is returned by `nextLine` is passed to the `parseInt` method. Finally, the `int` value returned by `parseInt` is assigned to the `int` variable `num`.

Declaring, Creating, and Using a `JOptionPane` Object

The `JOptionPane` class is defined in the package `javax.swing`. Because a `JOptionPane` is used to give us a graphical user interface with our program, we will actually be able to see the object when we use it.

When an object is used in a program, it must first be declared and then created. To declare a `JOptionPane` variable, we use a variable declaration statement:

```
JOptionPane myGUI;
```

To create the object we use the `new` operator with a constructor, `JOptionPane()`. This combination constructs, or creates, a new `JOptionPane` object that is then assigned to the `JOptionPane` variable, `myGUI`.

```
myGUI = new JOptionPane();
```

These two lines of code can be combined into a single declaration-initialization statement:

```
JOptionPane myGUI = new JOptionPane();
```

Once an object exists, it can be used. The `JOptionPane` class, contains the method

```
public void showMessageDialog(Component c, String s)
```

which tells us that the `public` method `showMessageDialog` has two formal parameters: `Component c` and `String s`. The return type of the method is `void`, which means that the method returns nothing. To use the method, invoke it on the object in the statement

```
myGUI.showMessageDialog(null, "I love Java!!");
```

Variables that are passed to a method are called the *actual parameters*; values that are passed to a method are called *arguments*. The first argument, `null`, is assigned to the formal parameter `Component c`. Simply put, `null` indicates that this `JOptionPane` is not attached to another `Component` or window. The second argument, `"I love Java!!"`, is assigned to the formal parameter `String s`.

Experiment 6.1

Step 1. Compile and execute `J06E01.java`. Record the results.

```
import javax.swing.*;
class J06E01
{
    public static void main(String[] args)
    {
        JOptionPane myGUI = new JOptionPane();
    }
}
```

Step 2. Modify the program in Step 1 by adding this line of code to the `main` method.

```
myGUI.showMessageDialog(null, "I love Java!!");
```

Compile and execute the modified program. Describe what appears. Record what happens when the button is clicked.

Step 3. Modify the string to be printed, changing it to `"I\nlove\nJava!!!"`. Compile and execute the program. Record the results.

Step 4. Remove the statement `import javax.*;` from the program. How does the compiler inform you of this error?

Experiment 6.2

The `JOptionPane` class also defines a method that allows the user to input information. The method header

```
public String showInputDialog(String prompt)
```

tells us that, when called, the method must be passed a `String`. The returned `String` can be saved, for future use, by assigning it to a `String` variable.

Step 1. Compile and execute the program `J06E02.java`. Describe what appears.

```
import javax.swing.*;
class J06E02
{
    public static void main(String[] args)
    {
        JOptionPane myGUI = new JOptionPane();
        String answer, greeting;
        answer = myGUI.showInputDialog("Enter your name");
        greeting = "Hi " + answer;
        myGUI.showMessageDialog(null, greeting);
    }
}
```

Step 2. Enter your name and click the OK button. Record the results.

Step 3. Execute the program again, this time respond by just clicking the Cancel button. Record the results.

Step 4. Execute the program again, this time respond by entering your name and then clicking the Cancel button. Record the results.

Step 7. Execute the program, entering a non-integer value at the prompt and clicking the OK button. Record the results.

Step 8. Execute the program, entering nothing at the prompt and clicking the OK button. Record the results.

Declaring, Creating, and Using a Random Object

Random numbers are used in computer games and in simulations. For example, if five dice are rolled in a game of Yahtzee, the upturned face of each die must be a randomly chosen number between 1 and 6. Computer simulations are often used to predict behaviors in a complex system such as airport traffic. Random numbers can be used to generate arrival, departure, and waiting times to test the air traffic patterns. The computer simulation can provide valuable information to justify changes to the design of the airport.

The Java API defines a class `Random` in the `java.util` package. The `Random` class defines methods that return random `int` values and others that return random `double` values. The random numbers that are generated are actually calculated using sophisticated formulas that make them appear to be random. Therefore, these numbers are called pseudo-random numbers.

To declare and create a `Random` object, use the constructor `public Random()`.

```
Random rand = new Random();
```

To generate a random integer value, use the method

```
public int nextInt()
```


Step 2. Modify the program to generate double values by adding the line

```
double realNum;
```

and replacing the line

```
num = rand.nextInt();
```

with

```
realNum = rand.nextDouble();
```

Compile and execute the program. List samples of values that are generated.

Experiment 6.4

Usually the pseudo-random values that are generated must be modified to make them usable. For example, if we are using the random number to represent the face of a die, the number must be an integer between 1 and 6. The `Random` class has a second `nextInt` method:

```
public int nextInt(int n)
```

Having more than one method with the same name is called *method overloading*. Method overloading is allowed as long as the methods have different parameter lists. This new method has a single formal parameter, `int n`. The original `nextInt` method has no formal parameter.

Step 1. Compile and execute `J06E04.java`. Record the results by listing samples of values that are generated. Comment on the output.

```
import java.util.*;
class J06E04
{
    public static void main(String[] args)
    {
        Random rand = new Random();
        int num;
        for(int j = 0; j < 20; j++)
        {
            num = rand.nextInt(100);
            System.out.println(num);
        }
    }
}
```


Declaring, Creating, and Using String Objects

We have been using `String` objects in every program that we have written. The `String` class is in the `java.lang` package, which is imported automatically into every program. To declare and create a `String` object we can follow the pattern established when we created the `JOptionPane` and `Random` objects.

```
String mouse = new String("Mickey Mouse");
```

The `String` class defines many interesting methods that we can use with `String` objects. First, let's look at these two methods:

```
public String toUpperCase()
```

and

```
public String toLowerCase()
```

To use either of these methods, invoke the method on a `String` object. No arguments are needed and a `String` object is returned. Can you predict what `String` is returned by each of these methods? Let's check it out.

Experiment 6.5

Step1. Compile and execute `J06E05.java`. Record the results.

```
class J06E05
{
    public static void main(String[] args)
    {
        String mouse = new String("Mickey Mouse");
        System.out.println(mouse);
        System.out.println(mouse.toLowerCase());
    }
}
```

Step 6. Because `String` objects are used so frequently, Java makes an exception in the creation of `String` objects. Java allows a `String` object to be created without using the `new` operator and a `String` constructor. The `String` class is the only class that has this special exception.

Replace the line

```
String mouse = new String("Mickey Mouse");
```

with

```
String mouse = "Mickey Mouse";
```

Compile and execute the program. Record the result.

Experiment 6.6

Every `String` object has a `length` representing the number of characters in the string. If

```
String river = "Nile";
```

then the `length` of `river` is 4. The `String` class defines the method

```
public int length()
```

that returns the `length` of this `String`. This code assigns 4 to `int len`:

```
len = river.length();
```

Each of the characters stored in a `String` object has an associated number called an index. The index of the first character in the string is always 0. The indices of the second, third and fourth characters are 1, 2 and 3, respectively. Note that a string with length 4 has characters with indices 0,1,2 and 3.

The `String` class also defines the method

```
public char charAt(int index)
```

that returns the character at `index`. This code assigns 'N' to `char c`:

```
c = river.charAt(0);
```

Step 1. Compile and execute the program J06E06.java. Record the results.

```
class J06E06
{
    public static void main(String[] args)
    {
        String river = "Amazon";
        char c;
        int len = river.length();
        System.out.println(river + " has length " + len);
        c = river.charAt(0);
        System.out.println(c);
    }
}
```

Step 2. Modify the code by passing 3 to the `charAt` method instead of 0. Predict the results, then compile and execute the program. Record the results.

Step 3. Modify the existing code by replacing the statement

```
c = river.charAt(3);
```

with the statement

```
c = river.charAt(len);
```

Predict the results, then compile and execute the program. Record the results.

Step 5. Modify the program by assigning the string "Nile" to `river`. Make no other changes. Compile and execute the modified program. Record the results. If the program does not work correctly, go back to Step 4.

Experiment 6.7

The `String` class also defines a method with header

```
public String substring(int index)
```

Let's use this method in a program to determine what `String` is returned.

Step 1. Compile and execute the program `J06E07.java`. Record the results in the first line of the table below.

```
import javax.swing.*;
class J06E07
{
    public static void main(String[] args)
    {
        String river = "Mississippi";
        JOptionPane myGUI = new JOptionPane();
        myGUI.showMessageDialog(null, river.substring(1));
    }
}
```

index	river.substring(index) returns
1	
3	
5	
8	
10	
0	

Step 2. Modify the program by replacing the 1 in `river.substring(1)` with each of the integers listed in the above table. Record the results in the table above.

Step 3. What is returned by the method `public String substring(int index)`?

Step 4. The method `substring` is an overloaded method. The second method has header

```
public String substring(int beginIndex, int endIndex)
```

Replace the statement

```
myGUI.showMessageDialog(null, river.substring(1));
```

with the statement

```
myGUI.showMessageDialog(null, river.substring(1,5));
```

Compile and execute the program. Record the results in the first line of the following table.

beginIndex	endIndex	<code>river.substring(beginIndex, endIndex)</code> returns
1	5	
1	7	
4	4	
4	7	
5	9	
0	4	

Step 2. Modify the program by replacing the 1 and 5 in `river.substring(1,5)` with each of the integers listed in the above table. Record the results in the table.

Step 3. What is returned by the method `public String substring(int index)`?

Step 2. Add the code that creates both the `JOptionPane` and `Random` objects in the appropriate places in the existing code. What did you add?

Step 3. Compile and execute the program. Respond with a `3` at the first prompt and a `y` at the second prompt. Continue with these responses at least four times. Record the results.

Step 4. Correct the code to assign to `num` a value in the correct range. Compile and execute the modified program. Report the results.

Step 5. Execute the program using different integer responses to verify that the program works correctly. Record the results.

Step 6. Until now, we have not learned how an individual character is retrieved from the keyboard. Explain how a single character is read in this program.

Step 7. Modify the code so that the program continues if the user enters either a 'y' or a 'Y'. Record your solution. Compile and execute the program. Record the results.

Optional Post-Laboratory Problems

- 6.1.** Write a program that uses a `JOptionPane` object to read in a user-entered Fahrenheit temperature, calculates the Celsius temperature, and reports the results using the `JOptionPane` object.
- 6.2.** Write a program that uses a `JOptionPane` object to read a user-entered measurement in meters (a double). Convert the measurement to centimeters, and report the results using the `JOptionPane` object.
- 6.3.** Write a program that uses a `JOptionPane` object to read a user-entered measurement in meters (a double). Convert the measurement to feet and inches, and report the results using the `JOptionPane` object. Note: 1 meter = 39.37 inches.
- 6.4.** Write a program that implements the game "Guess my number". Generate a random number between 1 and 100. For each user guess, respond with either "higher", "lower" or "You guessed it". When the number is guessed, ask the user if he would like to play again. If the answer is yes, a different random number must be generated.
- 6.5.** Write a program that first prompts the user for a word and then prints the first letter of the word on the first line, the first two letters of the word on the second line, the first three letters of the word on the third line, etc. For example: if the user enters `Java`, then the program should print:

```
J
Ja
Jav
Java
```

- 6.6.** Write a program that generates a random 3-letter word. Each letter should be randomly chosen by generating a random integer between 0 and 25 and adding that integer value to the letter 'a'.
- 6.7.** Modify problem 6.6 to generate 50 random 3-letter words. Run the program several times. Are any of the generated words actual English words?
- 6.8.** The `String` class has an overloaded method with method headers

```
public int indexOf(char letter)
public int indexOf(char letter, int beginIndex)
```

Write a program that can be used to test these two methods. Write a report that documents your test cases, listing both the arguments and the values returned. Draw a conclusion indicating what these two methods do.

- 6.9.** In addition to the two methods listed in problem 6.8, the `String` class defines two more methods named `indexOf`.

```
public int indexOf(String s)
public int indexOf(String s, int beginIndex)
```

Write a program that can be used to test these two methods. Write a report that documents your test cases, listing both the arguments and the values returned. Draw a conclusion indicating what these two methods do.