Lab 5  Name:__________________________  Checked:_____

Objectives:
• Learn about the Java API
• Practice using Math, Random, String, and other classes from the Java API
• Practice writing code to do basic String processing

Preparation: Exploring the Java API

Read and follow the instructions below and fill in your answers in this worksheet. You do NOT need to submit anything for this preparation, but in order to receive credit you must bring the completed worksheet to check with your partner in class.

Java derives much of its power from the many classes already defined in the Java Application Programming Interface (aka Java API). But how are we ever to learn and use these classes if we don’t know about them? Any textbook 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 the Java 8 API: https://docs.oracle.com/javase/8/docs/api/

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 Java API website (see link above) 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.

Access the Java API at the link above. Why is it abbreviated to Java SE (what does the SE stand for)?

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. What does it provide?
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 four summaries which are written in the orange background:

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

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(). As you can see, there are two versions of this method, both of which return a String. Look at the version with a parameter of type String named pattern. 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(). 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 the same information that we saw in the method summary with the added word `public`. The word `public` indicates that this method is “publically accessible” so that we can use it. The return type follows and is a String. A method only ever returns 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 interesting because it is 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 call `num`, the `nextInt()` method could be invoked one of two ways:

```java
int inputA = scan.nextInt();
int inputB = scan.nextInt(num);
```

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

```
________________________________________________________
```

Under the String class Method Summary, locate the String method `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?

```java
String fname = "Ben        ", lname = "Franklin";
System.out.println( fname + lname);
System.out.println( fname.trim() + lname);
```

```
________________________________________________________________________
```

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

The `Math` class is a class that only contains static methods. First, locate the Math class. In which Java package can you find the Math class?

```
________________________________________________________________________
```

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

---

Villanova University     CSC 1051     www.csc.villanova.edu/~map/1051     Dr. Papalaskari
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(396 - 400)); would result in 4.

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: ________________________________________________

Summary - What have you learned?

- Understand effects of binary data representation and the Java data types
- Explore automatic conversions, casts, and the Math class
- Understand the difference between String and char
- Explore the char type and Unicode representation
- Learn about some and String methods
- 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 starts with a lowercase letter and is directly to the left of the open parenthesis
- All methods require parentheses for their parameters
- 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

Partner name:_________________________________________

Signature:__________________________
Part A: jGrasp Interactions - String methods

*Type the expressions and statements in the interactions pane and note the results.*

String word = "evolve"

```
word.length() _____ word.charAt(1) _____ word.charAt(0) _____ 
word.toUpperCase() _____ word __________
word.replace("e","E") _____ word __________
word.replaceAll("e","E") _____ word __________
String mutation = word.replaceAll("e","")

mutation __________ word __________
word.substring(3) __________
word.substring(2,4) __________
```

Notice that word has NOT changed, throughout all these transformations. To effect a change, try:

```
word = word.toUpperCase() _____ word __________
```

Some more examples involving Strings:

String answer = "Yes"

```
answer == "yes" _____ answer == "Yes" _____ 
answer.equals("yes") _____ answer.equalsIgnoreCase("yes") _____ 
answer.charAt(0) == "y" _____ answer.charAt(0) == 'y' _____ 
answer.charAt(0) == 'y' | | 'Y' _____ 
answer.charAt(0) == 'y' | | answer.charAt(0) == 'Y' _____ 
int n = 0;
while (n < word.length())
{
    System.out.print(word.charAt(n) + "*");
    n++;
}
```

Partner’s signature: ____________________________

Villanova University      CSC 1051      www.csc.villanova.edu/~map/1051      Dr. Papalaskari
Part B: Work with Strings

1. Write a Java program `Initials.java` that asks your first name and last name and then prints a greeting using your initials. For example, an interaction might look like this:

   Please enter your first name: Grace
   Please enter your last name: Hopper
   Great meeting you, G.H., have a nice day.

2. Write a Java program `StringAcrobatics.java` based on the example from Part A:

   ```java
   String word = "evolve";
   int n = 0;
   while (n < word.length())
   {
       System.out.print(word.charAt(n) + "*");
       n++;
   }
   ```

   Modify it so that it prints
   - each letter in a separate line
   - with a $ in front of it and
   - three asterisks after.

   So the output should be as shown to the right:

3. Create a Java program `AllCaps.java` that asks your name and then prints it out one letter per line, with all letters printed in UPPERCASE.
   For example, an interaction might look like this:

   Please enter your name: Grace
   Hello...
   $ G ***
   $ R ***
   $ A ***
   $ C ***
   $ E ***

4. Modify `AllCaps.java` so that it works as before, but keeps repeating, until the user types QUIT (or “quit” or “Quit”- it should not be case sensitive), at which point the program should print a “thanks, goodbye” message and terminate.

   Partner's signature: __________________________

Villanova University       CSC 1051       www.csc.villanova.edu/~map/1051       Dr. Papalaskari
Part C. : jGrasp Interactions - Focus on numbers

1. Practice using the NumberFormat class.
   **Note:** For classes that are not in java.lang, we need to issue import directive:
   ```java
   import java.text.NumberFormat;
   ```
   ```java
   NumberFormat money = NumberFormat.getCurrencyInstance()
   NumberFormat percent = NumberFormat.getPercentInstance()
   double cash = 3.8;
   double rate = 0.028;
   money.format(cash) __________ percent.format(rate) __________
   ```

2. Use the DecimalFormat class to control printing:
   ```java
   import java.text.DecimalFormat;
   DecimalFormat fmt = new DecimalFormat("0.####");
   DecimalFormat fmtAlt = new DecimalFormat(".####");
   fmt.format(Math.PI) __________ fmtAlt.format(Math.PI) __________
   fmt.format(Math.PI/4) __________ fmtAlt.format(Math.PI/4) __________
   ```

3. Practice using the Random class. Be sure to enter each expression repeatedly and note the values generated.
   **Note:** Random is also not in java.lang, so you need the import directive:
   ```java
   import java.util.Random;
   ```
   ```java
   Random rand = new Random()
   rand.nextInt(4) __________ __________ // repeat a few times
   (use up-arrow on keyboard to avoid having to re-enter the Java code)
   ...
   __________ __________ __________ __________
   Range of values for rand.nextInt(4) __________________________
   rand.nextFloat() __________ __________ __________ __________...  
   Range of values for rand.nextFloat() __________________________
   ```
   
   ➢ Similarly, experiment with the following expressions and note the range of values for each one:
   ```text
   Range of values for rand.nextInt(6) + 1 __________________________
   Range of values for rand.nextInt(50) + 100 __________________________
   Range of values for rand.nextInt(10) - 5 __________________________
   ```

   **Partner’s signature:** __________________________
Part D: Work with Randomly generated integers

1. Formulate expressions to produce each of the following ranges. Be sure to test each one to make sure you are getting values in the range.

<table>
<thead>
<tr>
<th>Range</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 12</td>
<td></td>
</tr>
<tr>
<td>1 to 20</td>
<td></td>
</tr>
<tr>
<td>15 to 20</td>
<td></td>
</tr>
<tr>
<td>-10 to 0</td>
<td></td>
</tr>
<tr>
<td>100000 to 999999</td>
<td></td>
</tr>
<tr>
<td>2,000 to 3,000,000</td>
<td></td>
</tr>
</tbody>
</table>

2. Write a Java program `RandomJackpot.java` that prints a random winning number in the range 100000-999999 and followed by a random monetary amount for the winner of the jackpot in the range $2,000-$3,000,000. For example, the interaction could look like this:

Congratulations! you have the winning ticket: 820673
You have won $501,790.00 !!!

Partner’s signature: ___________________________
Lab 5 Comments

Name: __________________________

Comments on this lab, please:

What was the most valuable thing you learned in this lab?

What did you like best about this lab?

Was there any particular problem?

Do you have any suggestions for improving this lab as an effective learning experience?