Lab 5

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:
Complete all warmup exercises (Data types, Casts, Strings, Java API – separate handout).
*Be sure to have handout with you to check with your partner in the lab.*

Part A. Focus on Strings

1. Using your code that you submitted for the preparation, modify it to print the word in UPPERCASE and with spaces around each letter.
   So the output should look like this:
   
   Write the code here:

   ```java
   String word = "evolve";
   $ E ***
   $ V ***
   $ O ***
   $ L ***
   $ V ***
   $ E ***
   
   Hint: Use the String method toUpperCase().
   ```

2. 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.

3. Write a Java program **AllCaps.java** that asks your name and then prints it out one letter per line, similar to 1, above, ALL CAPS.
   For example, an interaction might look like this:

   Please enter your name: Grace
   Hello...
   $ G ***
   $ R ***
   $ A ***
   $ C ***
   $ E ***
4. Let's try some more code snippets (Use jGrasp interactions pane.)

```java
String message = "";
int n = 1;
while (n <= 10)
{
    message += "#";
    n++;
}
```

Explain in your own words what the above loop does

(we will be using something similar in part C).

```java
String funds = "$402.63"

funds.length() funds.charAt(1) funds.charAt(0) funds.substring(1)
```

In general, what is the effect of .substring(1)?

```java
funds.charAt(_______________) = 3 (fill in a number to get last char in String)
```

In general, the last character in a String will have index ________________

```java
double amount = Double.parseDouble(funds.substring(1))

funds + 1 amount + 1
```

```java
int dollars = (int) amount
int cents = (int) ((amount - dollars) * 100)
```

```java
Integer.toString(536202) Integer.toString(536202).length()
```

5. Write a Java program `DollarsAndCents.java` that asks you input an amount and then prints it out as dollars and cents. For example, an interaction might look like this:

How much? $203.56
That is 203 dollars and 56 cents.

*Hint:* Use the String method `trim()` to get rid of extra spaces and the space between `$` and `2`. 
Part B. Focus on numbers
1. Run this code, explain what happens.

```java
double number = 0.0;
while (number != 1)
{
    System.out.println(number);
    number += 0.1;
}
```

Explain in your own words what happened and why.

_____________________________________________________________________________________________
_____________________________________________________________________________________________
_____________________________________________________________________________________________
_______________________________

Fix it by changing the condition so that it works as expected, i.e., outputs only the numbers 0.1, 0.2, ... 0.9.
What condition did you use?

2. Practice using the NumberFormat class.
Note: For classes that are not in java.lang, we need to issue import directive:
```
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) ________
```

3. Use the DecimalFormat class to control printing:
```
import java.text.DecimalFormat;
DecimalFormat fmt = new DecimalFormat("0.####");
DecimalFormat fmtAlt = new DecimalFormat(".####");
```

```java
fmt.format(Math.PI) ____________ fmtAlt.format(Math.PI) ____________
fmt.format(Math.PI/4)___________ fmtAlt.format(Math.PI/4)___________
```
4. 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();
rnd.nextInt(4) __________ __________ // repeat a few times
```

(Use up-arrow on keyboard to avoid having to re-enter the Java code)

```java
... __________ __________ __________ __________
```

Range of values for `rand.nextInt(4)` ___________________________

Range of values for `rand.nextFloat()` __________ __________ __________ __________ __________ ...

5. Write a Java program `RandomJackpot.java` that inputs a number N and prints a series of N random letters 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

```
How many random letters?: 50
LDAWTYWLBDPLQCMUXIZRBUSTNBBHNTMCWJCVWRUQISQTMEV
You have won $501,790.00 !!!
```

Villanova University  CSC 1051  www.csc.villanova.edu/~map/1051  Dr. Papalaskari
Part C. Implementing a customer service simulation

1. Download the code for `QueueingTheory_Start.java` from the course website, reproduced below. Observe how it simulates a queue of customers in a queue to be served.

```java
//*************************************************************
//  QueueingTheory_Start.java       Author: M A Papalaskari
//  CSC 1051 Lab 5: Starter code for Lab 5 and Project 5 ***
//*************************************************************
import java.util.*;

public class QueuingTheory_Start {
    public static void main(String[] args) {
        // Randomized arrivals (and more?)
        String queue = ""; // queue starts out empty
        int minute = 1; // unit of time is minutes
        double arrivalProbability = 0.25; // prob of customer arrival
        // on any given minute (try different values!)
        int processingTime = 4; // each customer takes 4 minutes
        char personLetter = 'A'; // symbolizes each customer
        System.out.println("n
***************************************");
        System.out.println("Call center simulation.");
        System.out.println("Calls placed with probability per minute: "+ arrivalProbability);
        System.out.println("Call duration: " + processingTime);

        // main loop of simulation - lasts 60 minutes
        while (minute <= 60) {
            // randomly decide if a customer has arrived and add to queue
            if (rand.nextFloat() < arrivalProbability) {
                // add as many symbols as processingTime (minutes) required
                int n = 0;
                while (n < processingTime){
                    queue += personLetter;
                    n++;
                }
                personLetter++; // change symbol for next customer
            }
            // display the current state of the queue
            System.out.println(minute + " >>> \	" + queue);

            // serve customer at front of queue (if any) for 1 minute
            if (queue.length() > 0)
                queue = queue.substring(1);

            minute++; // tick tock... as time goes by
        }
    }
}
```
2. Change the length of the simulation to 30 minutes and experiment with different the values of the following variables to observe the results

arrivalProbability  (try smaller and larger values between 0 and 1) __________________________

processingTime    (try smaller and larger integer values) _________________________________

personLetter      (try 'a' or other characters) ________________________________

3. Modify the code to display more details, including the queue length and the contents of the queue enclosed in <<< and >>>. The output should look as follows.

Call center simulation.
Calls placed with probability per minute: 0.25
Call duration:  4
minute 1 queue length 0 queue contents: <<<>>>
minute 2 queue length 0 queue contents: <<<>>>
minute 3 queue length 0 queue contents: <<<>>>
minute 4 queue length 0 queue contents: <<<>>>
minute 5 queue length 0 queue contents: <<<>>>
minute 6 queue length 4 queue contents: <<<AAAA>>>  
minute 7 queue length 3 queue contents: <<<AA>>>  
minute 8 queue length 2 queue contents: <<<AA>>>  
minute 9 queue length 1 queue contents: <<<A>>>  
minute 10 queue length 0 queue contents: <<<>>>  
minute 11 queue length 0 queue contents: <<<>>>  
minute 12 queue length 0 queue contents: <<<>>>  
minute 13 queue length 0 queue contents: <<<>>>  
minute 14 queue length 4 queue contents: <<<BBBB>>>  
minute 15 queue length 3 queue contents: <<<BBB>>>  
minute 16 queue length 2 queue contents: <<<BB>>>  
minute 17 queue length 1 queue contents: <<<B>>>  
minute 18 queue length 0 queue contents: <<<>>>  
minute 19 queue length 0 queue contents: <<<>>>  
minute 20 queue length 4 queue contents: <<<CCCC>>>  
minute 21 queue length 3 queue contents: <<<CCC>>>  
minute 22 queue length 2 queue contents: <<<CC>>>  
minute 23 queue length 1 queue contents: <<<C>>>  
minute 24 queue length 0 queue contents: <<<>>>  
minute 25 queue length 0 queue contents: <<<>>>  
minute 26 queue length 4 queue contents: <<<DDDD>>>  
minute 27 queue length 3 queue contents: <<<DDD>>>  
minute 28 queue length 2 queue contents: <<<DD>>>  
minute 29 queue length 1 queue contents: <<<D>>>  
minute 30 queue length 0 queue contents: <<<>>>  

3. Examine the sample output above and answer these questions:

• How many idle minutes? (minutes the queue was empty) __________

• What is the maximum length of the queue? __________

• What is the average length of the queue in the last 10 minutes? __________
Lab 5 Comments

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?