CS 469 / CS 569 – Special Topics in Computer Science: Human-Computer Interaction

Assignment 2: Lab Exercises: Classes, Conditionals and Loops

Recalls

Some exercises focus on the String, Random, and Math classes defined in the Java Standard Class Library. The goals of the lab are for you to gain experience with the following concepts:

- Declaring a variable to be a reference to an object—for example, the following declares the variable *quotation* to be a reference to a String object:

```
String quotation;
```

- Declaring a variable to be a reference to an object and creating the object *(instantiating* it) using the *new* operator—for example,

```
String quotation = new String("I think, therefore I am.");

Random generator = new Random();
```

- Invoking a method to operate on an object using the *dot operator*—for example,

```
quotation.length()
```

invokes the length method which returns the length of the quotation String or

```
quotation.toLowerCase()
```

quotation except all letters are lower case. These invocations would be used in a program in a place appropriate for an integer (in the first case) or a String (in the second case) such as an assignment statement or a *println* statement.

- Invoking *static* or *class* methods—these are methods that are invoked using the class name rather than an object name. The methods in the Math class are static methods (basically because we don't need different instances of Math whereas there are lots of different String objects). Examples are

```
Math.sqrt(2)  (which returns the square root of 2)
```

and

```
Math.pow(3, 2)   (which returns 3^2)
```

- Importing the appropriate packages—usually when you use classes from a library you need to put the *import* declaration at the top of your program. The exception is for classes defined in the java.lang package (this includes String and Math) which is automatically imported into every Java program.

Exercise 1

1. Fill in the blanks in the program below as follows:
   (a) declare the variable *town* as a reference to a String object and initialize it to "Anytown, USA".
   (b) write an assignment statement that invokes the *length* method of the string class to find the length of the *college* String object and assigns the result to the *stringLength* variable
   (c) complete the assignment statement so that *change1* contains the same characters as *college* but all in upper case
   (d) complete the assignment statement so that *change2* is the same as *change1* except all capital O's are replaced with the asterisk (*) character.
   (e) complete the assignment statement so that *change3* is the concatenation of *college* and *town* (use the *concat* method of the String class rather than the + operator)

```
// ************************************************
//  StringPlay.java
//
//  Play with String objects
// ************************************************

public class StringPlay
{
   public static void main (String[] args){
      String college = new String ("PoDunk College");

      _____; // part (a)
      int stringLength;
      String change1, change2, change3;

      _____; // part (b)
      System.out.println (college + " contains " + stringLength + "
      characters.");

      change1 = _____; // part (c)
      change2 = _____; // part (d)
      change3 = _____; // part (e)
      System.out.println ("The final string is " + change3);
      }
}
```

2. The following program should read in the lengths of two sides of a right triangle and compute the length of the hypotenuse (recall that the length of the hypotenuse is the square root of side 1 squared plus side 2 squared). Complete it by adding statements to read the input from the keyboard and to compute the length of the hypotenuse (you need to use a Math class method for that).

```
//********************************************************************
//  RightTriangle.java
//
//  Compute the length of the hypotenuse of a right triangle
//  given the lengths of the sides
//********************************************************************

 import java.util.Scanner;

public class RightTriangle
{
   public static void main (String[] args)
   {
      double side1, side2;  // lengths of the sides of a right triangle
      double hypotenuse; // length of the hypotenuse

      Scanner scan = new Scanner(System.in);


      // Get the lengths of the sides as input
      System.out.print ("Please enter the lengths of the two sides of " +
                        "a right triangle (separate by a blank space): ");
      ------------------------------------------------------------;

      ------------------------------------------------------------;

      // Compute the length of the hypotenuse

      ------------------------------------------------------------;

      // Print the result
      System.out.println ("Length of the hypotenuse: " + hypotenuse);
   }
}
```

3. In many situations a program needs to generate a random number in a certain range. The Java Random class lets the programmer create objects of type Random and use them to generate a stream of random numbers (one at a time). The following declares the variable *generator* to be an object of type Random and instantiates it with the *new* operator.

```
Random generator = new Random();
```

The *generator* object can be used to generate either integer or floating point random numbers using either the *nextInt* method (either with no parameter or with a single integer parameter) or *nextFloat* (or *nextDouble)* methods, respectively. The integer returned by *nextIn(t* could be any valid integer (positive or negative) whereas the number returned by *nextInt(n)* is a random integer in the range 0 to n-1. The numbers returned by *nextFloat()* or *nextDouble()* are floating point numbers between 0 and 1 (up to but not including the 1). Most often the goal of a program is to generate a random integer in some particular range, say 30 to 99 (inclusive). There are several ways to do this:

- **Using nextInt():** This way we must use the % operator to reduce the range of values—for example,

```
Math.abs(generator.nextInt()) % 70
```

will return numbers between 0 and 69 (because those are the only possible remainders when an integer is divided by 70 - note that the absolute value of the integer is first taken using the *abs* method from

the Math class). In general, using % N will give numbers in the range 0 to N -1. Next the numbers must be shifted to the desired range by adding the appropriate number. So, the expression

```
Math.abs(generator.nextInt()) % 70 + 30
```

will generate numbers between 30 and 99.

- **Using nextInt(70):** The expression

  ```
  generator.nextInt(70)
  ```

  will return numbers between 0 and 69 (inclusive). Next the numbers must be shifted to the desired range by adding the appropriate number. So, the expression

  ```
  generator.nextInt(70)    +30
  ```

  will generate numbers between 30 and 99.

- **Using nextFloat:** In this case, we must multiply the result of *nextFloat* to expand the range—for example,

  ```
  generator.nextFloat() * 70
  ```

  returns a floating point number between 0 and 70 (up to but not including 70). To get the integer part of the number we use the cast operator:

  ```
  (int)   (generator.nextFloat()   *  70)
  ```

  The result of this is an integer between 0 and 69, so

  ```
  (int)   (generator.nextFloat()   *    70)+    30
  ```

  shifts the numbers by 30 resulting in numbers between 30 and 99.

  The method *nextFloat* can be replaced by *nextDouble* to get double precision floating point numbers rather than single precision.

Fill in the blanks in the following program to generate the random numbers as described in the documentation. NOTE that that *java.util.Random* must be imported to use the Random class.

```
// ************************************************
//  LuckyNumbers.java
//
//  To generate three random "lucky" numbers
// ************************************************

import java.util.Random;

public class LuckyNumbers
{
   public static void main (String[] args)
   {
      Random generator = new Random();
      int lucky1, lucky2,  lucky3;

// Generate lucky1 (a random integer between 50 and 79) using the
```

```
// nextInt method (with no parameter)
        lucky1 = _____;
// Generate lucky2 (a random integer between 90 and 100) using the
// nextInt method with an integer parameter
        lucky2 = _____;

// Generate lucky3 (a random integer between 11 and 30) using nextFloat

        lucky3 = _____;
        System.out.println ("Your lucky numbers are " + lucky1 + ", " +
        lucky2+ ", and " + lucky3);
      }
   }
```

Exercise 2: Working with Strings

The following program illustrates the use of some of the methods in the String class. Study the program to see what it is doing.

```
// ************************************************************
//   StringManips.j ava
//
//   Test several methods for manipulating String objects
// ************************************************************
import java.util.Scanner;

public class StringManips

{

    public static void main (String[] args)

    {

      String phrase = new String ("This is a String test.");

      int phraseLength;  // number of characters in the phrase String int

      middleIndex;     // index of the middle character in the String

      String firstHalf;  // first half of the phrase String

      String secondHalf; // second half of the phrase String

      String switchedPhrase; //a new phrase with original halves switched


      // compute the length and middle index of the phrase

      phraseLength = phrase.length();

      middleIndex = phraseLength / 2;


      // get the substring for each half of the phrase

      firstHalf = phrase.substring(0,middleIndex);

      secondHalf = phrase.substring(middleIndex, phraseLength);
```

```
        // concatenate the firstHalf at the end of the secondHalf
        switchedPhrase = secondHalf.concat(firstHalf);

        // print information about the phrase System.out.println();
        System.out.println ("Original phrase: " + phrase);
        System.out.println ("Length of the phrase: " + phraseLength +
                            " characters");
        System.out.println ("Index of the middle: " + middleIndex);
        System.out.println ("Character at the middle index: " +
                           phrase.charAt(middleIndex));
        System.out.println ("Switched phrase: " + switchedPhrase);
        System.out.println();
    }
}
```

The file *StringManips.java* contains this program. Save the file to your directory and compile and run it. Study the output and make sure you understand the relationship between the code and what is printed. Now modify the file as follows:

1. Declare a variable of type String named *middle3* (put your declaration with the other declarations near the top of the program) and use an assignment statement and the *substring* method to assign *middle3* the substring consisting of the middle three characters of *phrase* (the character at the middle index together with the character to the left of that and the one to the right - use variables, not the literal indices for this particular string). Add a *println* statement to print out the result. Save, compile, and run to test what you have done so far.
2. Add an assignment statement to replace all blank characters in *switchedPhrase* with an asterisk (*). The result should be stored back in *switchedPhrase* (so *switchedPhrase* is actually changed). (Do not add another print—place your statement in the program so that this new value of *switchedPhrase* will be the one printed in the current println statement.) Save, compile, and run your program.
3. Declare two new variables *city* and *state* of type String. Add statements to the program to prompt the user to enter their hometown—the city and the state. Read in the results using the appropriate Scanner class method - you will need to have the user enter city and state on separate lines. Then using String class methods create and print a new string that consists of the state name (all in uppercase letters) followed by the city name (all in lowercase letters) followed again by the state name (uppercase). So, if the user enters Lilesville for the city and North Carolina for the state, the program should create and print the string

```
NORTH CAROLINAlilesvilleNORTH CAROLINA
```

Exercise 3: Counting and Looping

The program in *LoveCS.java* prints "I love Computer Science!!" 10 times. Copy it to your directory and compile and run it to see how it works. Then modify it as follows:

```
// ************************************************************
//   LoveCS.java
//
//   Use a while loop to print many messages declaring your
//   passion for computer science
// ************************************************************

public class LoveCS
{
    public static void main(String[] args)
    {
      final int LIMIT = 10;

      int count = 1;

      while (count <= LIMIT){
          System.out.println("I love Computer Science!!");
          count++;

      }
    }
}
```

1. Instead of using constant LIMIT, ask the user how many times the message should be printed. You will need to declare a variable to store the user's response and use that variable to control the loop. (Remember that all caps are used only for constants!)
2. Number each line in the output and add a message at the end of the loop that says how many times the message was printed. So, if the user enters 3, your program should print this:

```
1 I love Computer Science!!
2 I love Computer Science!!
3 I love Computer Science!!
Printed this message 3 times.
```

3. If the message is printed N times, compute and print the sum of the numbers from 1 to N. So for the example above, the last line would now read:

   ```
   Printed this message 3 times. The sum of the numbers from 1 to 3 is 6.
   ```

   Note that you will need to add a variable to hold the sum.

Exercise 4: Powers of 2

File *PowersOf2.java* contains a skeleton of a program to read in an integer from the user and print out that many powers of 2, starting with 20.

1. Using the comments as a guide, complete the program so that it prints out the number of powers of 2 that

the user requests. **Do not use Math.pow to compute the powers of 2!** Instead, compute each power from the previous one (how do you get 2^n from 2^{n-1}?). For example, if the user enters 4, your program should print this:

```
Here are the first 4 powers of 2:
1
2
4
8
```

2. Modify the program so that instead of just printing the powers, you print which power each is, e.g.:

```
Here are the first 4 powers of 2:
2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
```

```
// ********************************************************
//   PowersOf2.java
//
//   Print out as many powers of 2 as the user requests
//
// ********************************************************

import java.util.Scanner;

public class PowersOf2
{
    public static void main(String[] args)
    {
        int numPowersOf2;      //How many powers of 2 to compute
        int nextPowerOf2 = 1;  //Current power of 2
        int exponent;          //Exponent for current power of 2 -- this
                               //also serves as a counter for the loop
        Scanner scan = new Scanner(System.in);

        System.out.println("How many powers of 2 would you like printed?");
        numPowersOf2 = scan.nextInt();

        //print a message saying how many powers of 2 will be printed
        //initialize exponent -- the first thing printed is 2 to the what?
        while ( )
        {
            //print out current power of 2

            //find next power of 2 -- how do you get this from the last one?

            //increment exponent
        }
    }
}
```

Exercise 5: Factorials

The *factorial* of n (written n!) is the product of the integers between 1 and n. Thus 4! = 1*2*3*4 = 24. By definition, 0! = 1. Factorial is not defined for negative numbers.

1. Write a program that asks the user for a non-negative integer and computes and prints the factorial of that

integer. You'll need a while loop to do most of the work—this is a lot like computing a sum, but it's a product instead. And you'll need to think about what should happen if the user enters 0.

2. Now modify your program so that it checks to see if the user entered a negative number. If so, the program should print a message saying that a nonnegative number is required and ask the user the enter another number. The program should keep doing this until the user enters a nonnegative number, after which it should compute the factorial of that number. **Hint:** you will need another while loop **before** the loop that computes the factorial. You should not need to change any of the code that computes the factorial!

Exercise 6: A Shopping Cart Using the ArrayList Class

In this exercise you will implement a shopping cart using the ArrayList class. The file *Item.java* contains the definition of a class named *Item* that models an item one would purchase (this class was used in an earlier lab). An item has a name, price, and quantity (the quantity purchased). The file *Shop.java* is an incomplete program that models shopping.

1. Complete Shop.java as follows:
   a. Declare and instantiate a variable *cart* to be an empty ArrayList.
   b. Fill in the statements in the loop to add an item to the cart and to print the cart contents (using the default *toString* in the ArrayList class). Comments in the code indicate where these statements go.
   c. Compile your program and run it.
2. You should have observed two problems with using the default printing for the cart object: the output doesn't look very good and the total price of the goods in the cart is not computed or printed. Modify the program to correct these problems by replacing the print statement with a loop that does the following:
   a. gets each item from the cart and prints the item
   b. computes the total price of the items in the cart (you need to use the *getPrice* and *getQuantity* methods of the Item class). The total price should be printed after the loop.
3. Compile and run your program.

```
// ********************************************************
//   Shop.java
//
//   Uses the Item class to create items and add them to a shopping
//   cart stored in an ArrayList.
// ********************************************************

import java.util.ArrayList;
import java.util.Scanner;

public class Shop
{
   public static void main (String[] args)
   {
```

```
      Item item;
      String itemName;
      double itemPrice;
      int quantity;

      Scanner scan = new Scanner(System.in);

      String keepShopping = "y";

      do
         {
            System.out.print ("Enter the name of the item: ");
            itemName = scan.nextLine();
            System.out.print ("Enter the unit price: ");
            itemPrice = scan.nextDouble();
            System.out.print ("Enter the quantity: ");
            quantity = scan.nextInt();
            // *** create a new item and add it to the cart

            // *** print the contents of the cart object using println

            System.out.print ("Continue shopping (y/n)? ");
            keepShopping = scan.nextLine();
         }
      while (keepShopping.equals("y"));

   }
}
```