csTask15 Assignment

The activities in this assignment require you to create and modify Java programs using BlueJ. Create a new BlueJ project named csTask15.

All programs (class files) must be in your new project, and the project must be copied to your shared Dropbox folder before the end of class today.

When questions are posed, write your responses as full sentences in the Readme.txt file in your project.

There are seven parts to this assignment:

I. Background
II. Manipulating Strings
III. Using the Pythagorean Theorem
IV. Generating Random Values
V. Working with Strings
VI. Rolling Dice
VII. Computing Distance

I. Background

The exercises in this task focus on the String, Random, and Math classes defined in the Java Standard Class Library. The main concepts are in the text in Chapter 2. 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 `quotation` or

```
quotation.toLowerCase()
```

which returns `quotation` except all letters are lower case. These invocations would be used in a program in a place appropriate for an `int` (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.

II. Manipulating Strings

1. You are to copy and paste the following Java program into a new class in your project. Use Edit / Auto-layout to format the program and then fill in the Javadoc comment space at the top of the class. Then 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` 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)
System.out.println ("The variable change1 is " + change1);
change2 = ___________________________________________; // part (d)
System.out.println ("The variable change2 is " + change2);
change3 = ___________________________________________; // part (e)
System.out.println ("The final string is " + change3);
}
}
```

III. Using the Pythagorean Theorem

The following program should read in the lengths of two sides of a right triangle and compute the length of the hypotenuse in accordance with the Pythagorean Theorem. You are to copy and paste the following Java program into a new class in your project. Use Edit / Auto-layout to format the program and then fill in the Javadoc comment space at the top of the class. Then 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);
}
}
```

IV. Generating Random Values

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 `nextInt()` 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 the `random` method of the `Math` class: The expression `Math.random()` returns a `double` value between 0 and 1 (up to but not including 1), similar to the `nextDouble` method of the `Random` class. Notice that `Math.random()` is an invocation of a static method whereas `nextDouble` must be invoked with an instance of a `Random` object. Use multiplication, casting, and addition to return an integer in a given range. For instance, `(int) (Math.random() * 100)` returns an integer between 0 and 99, inclusive.

You are to copy and paste the following Java program into a new class in your project. Use Edit / Auto-layout to format the program and then fill in the Javadoc comment space at the top of the class. Then fill in the blanks in the program to generate the random numbers as described in the documentation.

NOTE 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, inclusive)
// using the nextInt method (with no parameter)
lucky1 = _____________________________________________________;
// Generate lucky2 (a random integer between 90 and 100, inclusive)
// using the nextInt method with an integer parameter
lucky2 = _____________________________________________________;
// Generate lucky3 (a random integer between 11 and 30, inclusive)
// using the random method of the Math class
lucky3 = _____________________________________________________;
System.out.println ("Your lucky numbers are " + lucky1 + ", " + lucky2
+ ", and " + lucky3);
}
}
```

V. Working with Strings

The following program illustrates the use of some of the methods in the String class. You are to copy and paste the following Java program into a new class in your project. Use Edit / Auto-layout to format the program and then fill in the Javadoc comment space at the top of the class.

```
// ****************************************************************
// StringManips.java
//
// 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();
}
}
```

Compile and run it. Study the output and make sure you understand the relationship between the code and what is printed. *Continued on next page…*

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 city name (all in lowercase letters) followed by a comma and a space and the state name (all in uppercase). So, if the user enters `Lilesville` for the city and `North Carolina` for the state, the program should create and print the string

```
lilesville, NORTH CAROLINA
```

VI. Rolling Dice

Write a complete Java program that simulates the rolling of a pair of dice. For each die in the pair, the program should generate a random number between 1 and 6 (inclusive). It should print out the result of the roll for each die and the total roll (the sum of the two dice), all appropriately labeled. You can use either the Random class or the Math class.

VII. Computing Distance

The following program contains an incomplete program to compute the distance between two points. You are to copy and paste the following Java program into a new class in your project. Use Edit / Auto-layout to format the program and then fill in the Javadoc comment space at the top of the class.

Recall that the distance between the two points (x1, y1) and (x2, y2) is computed by taking the square root of the quantity $(x1 - x2)^2 + (y1 - y2)^2$. The program already has code to get the two points as input. You need to add an assignment statement to compute the distance and then a print statement to print it out (appropriately labeled).

Test your program using the following data: The distance between the points (3, 17) and (8, 10) is 8.6023... (lots more digits printed); the distance between (–33, 49) and (–9,–15) is 68.352.…

```
// ************************************************************
// Distance.java
//
// Computes the distance between two points
// ************************************************************
import java.util.Scanner;

public class Distance
{
public static void main (String[] args)
{
double x1, y1, x2, y2; // coordinates of two points
double distance; // distance between the points
Scanner scan = new Scanner(System.in);
// Read in the two points
System.out.print ("Enter the coordinates of the first point " +
"(put a space between them): ");
x1 = scan.nextDouble();
y1 = scan.nextDouble();
System.out.print ("Enter the coordinates of the second point: ");
x2 = scan.nextDouble();
y2 = scan.nextDouble();
// Compute the distance
// Print out the answer
}
}
```