ASSIGNMENT 2

COMP-202, Fall 2014, All Sections

Due: October 8^{th}, 2014 (23:59)

Please read the entire pdf before starting.

You must do this assignment individually and, unless otherwise specified, you must follow all the general instructions and regulations for assignments. Graders have the discretion to deduct up to 10% of the value of this assignment for deviations from the general instructions and regulations. These regulations are posted on the course website. Be sure to read them before starting.

Question 1:	40 points
Question 2:	40 points
Question 3:	20 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment through automated tests. While these tests will not determine your entire grade, it will speed up the process significantly, which will allow the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. Marks can be removed if comments are missing, if the code is not well structured, and if the problems your solution does not respect the assignment requirement.

Assignment

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions.*

Warm-up Question 1 (0 points)

Create a file called `Counting.java`, and in this file, declare a class called `Counting`. This class should ask the user when the computer should stop counting.

```
When should I stop counting to?
10 <---- User typed this
I am counting until 10: 1 2 3 4 5 6 7 8 9 10
```

Warm-up Question 2 (0 points)

For this question you have to generalize the last question. The user will give you the number they want the computer to count up to and the step by which it will do so.

```
When should I stop counting to?
25 <----
Which step should I use?
3 <----
I am counting to 25 with a step of 3:
1 4 7 10 13 16 19 21 24
```

Warm-up Question 3 (0 points)

This program should ask the user how big she wants the square to appear. Using two loops, you should be able to describe the outline of a square like the following.

```
How big do you want your square to be?
10 <---- User typed this
##########
#        #
#        #
#        #
#        #
#        #
#        #
#        #
#        #
##########
```

N.B. It is normal that it does not output a perfect square as the width and the length of the characters are not equal.

How would you extend your program to output a rectangle with width and length specified by the user?

Warm-up Question 4 (0 points)

Write a program that asks the user to enter an odd number. If the user enters an even number, the program should ask the user again to input an odd number. In order to achieve this, you will have to use the modulo operator[1].

Part 2

The questions in this part of the assignment will be graded.

Question 1: Binary Converter (35 points)

You will write a program that converts binary numbers to base 10 numbers. This program will ask the user to enter a binary number. You will have to verify that what is entered by the user only has 0s and 1s. In the case the user entered letters, you have to keep asking the user for another input. When the input is valid, you will convert the binary number to a base 10 number. Please use the `Question1.java` file provided in the `A2.zip` file.

Valid input - In order to check if the input is valid your program should call the `CheckInputCorrect` method, which takes a **String** as an input parameter and returns a **boolean** value. An input string is considered valid if it corresponds to a number in binary representation.

More specifically, in the `CheckInputCorrect` method, you should scan this string to make sure it only contains '0' or '1' characters. As soon as you find a character that is not '0' or '1', the method should returns `false`. If the method reaches the end of the input string (i.e. all characters are '0' or '1') the method should return `true`.

[1] `http://mathbits.com/MathBits/Java/DataBasics/Mathoperators.htm` or `http://youtu.be/HN_oBvDUQlg?t=8m44s`

Converter - At this point we assume that the input string is valid (checked with the `CheckInputCorrect` method). To convert the input from binary to base 10, you must implement the `BinaryToNumber` method. The `BinaryToNumber` method should take as parameter a `String` and return an `integer`, which corresponds to converted value in base 10.

The binary conversion should work as follows. For each digit in the binary number, if the digit is '1' you should add the corresponding decimal value (2^0 for the rightmost digit, 2^1 for the next digits to the left, 2^2 for the next one, and so on) to a variable that will hold the final result to be returned. This can be easily accomplished by using a loop.

Here is an example of how an interaction with your program should look like,

```
Marvin: Please enter a binary number.
No!                  <-------- This is entered by the user
Marvin: This will all end in tears. Can you just please enter a number?
Well maybe.          <-------- This is entered by the user
Marvin: This will all end in tears. Can you just please enter a number?
101010               <-------- This is entered by the user
Marvin: I've calculated your binary number, but I don't think you'll like it.
Marvin: The binary number 101010 is 42 in base 10.
```

Question 2: Drawing polynomials curve (45 points)

You will write a program which will allow you to draw curves corresponding to functions of the form $y = ax + b$ (straight lines) and $y = ax^2 + bx + c$ (parabolic curves) using `System.out.print("")` statement and conditional statements. Please use the `Question2.java` file provided in the `A2.zip` file.

Start by drawing the x-axis and y-axis, where each quadrant has 10 by 10 grid size. The end of both axes in the positive direction should display an arrow head and the origin of the plot should display a period, *e.g.*

```
          ^
          |
          |
          |
          |
          |
          |
          |
          |
          |
----------.--------->
          |
          |
          |
          |
          |
          |
          |
          |
          |
          |
```

To achieve this, you will have to use two loops, one counting the x position and the other one counting the y position. For each position (x, y) in the grid, you need to use conditional statements (i.e. an imbrication of `if-else if-else` statements) to determine what to draw at that particular position. Pay attention to the order of importance for drawing the symbols [2]. Symbols with higher importance should appear higher up in the `if-else` structure of conditional statements.

[2] e.g. The axis arrow head `>` and `^` have higher importance than the axis lines `-`.

This part should be implemented in the `DrawAxis` method, in the file `Question2.java`. The `DrawAxis` method should receive two integer numbers `x` and `y` as input parameters, corresponding to a position (x, y) in the 2D plane. The method should **return**, not print, a **String** with a single character. This character is either an empty string ("") or one of the axis characters, depending on the position (x, y). This character will be returned to the `main` method, where the it will be printed.

Linear Function (Straight lines) - To draw straight lines, you must implement the `DrawLine` method. In this method, the (x, y) positions where the line should be drawn are calculated using the equation $y = a(x) + b$, where a and b specify the slope and intercept of the line[3]. The values of a and b should be set in the `main` method; you can imagine that these could be defined by the user using the `Scanner`. When a position (x, y) satisfies the equation $y = ax + b$, the method should return the line character `*`.

Since our drawing system is not continuous (i.e. it is a 20 by 20 grid), you need to relax the equality in the equation $y = ax + b$. Instead of drawing the line character in the line when the equality is satisfied, you should draw a point in the line whenever $y - \epsilon < ax + b < y + \epsilon$, where ϵ represents *thickness* of the line being drawn. Thus the `DrawLine` method receives two integer numbers `x` and `y` as input parameters, corresponding to a position (x, y). If the previous inequality evaluates to `true`, the method returns the line character, `*`. Otherwise it returns the empty string.

Quadratic Function (Parabola) - Similar to the previous method, you must implement the `DrawParabola` method to determine if the position (x, y) is part of the parabola or not. The points on a parabola satisfy the equation $y = a(x^2) + b(x) + c$, where a, b and c are parameters that control the shape of the parabola [4]. These parameters need to be defined in the `main` method. Make sure to use different variable names to the ones used to define for the straight line parameters.

As before, you need to relax the condition for drawing the parabola by using the ϵ thickness parameter. When that inequality (that you have to find) evaluates to `true`, the method should return the parabola character `#`. Otherwise it should return the empty string "". The order of importance in the `main` method is: 1) Axes, 2) Straight line, 3) Parabola. This means that the axis lines should alway be visible, and that the straight line should be drawn on top of the parabola.

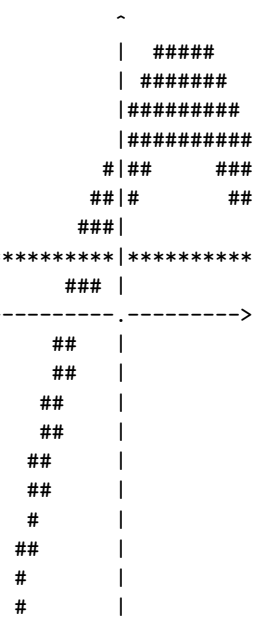
Here are some examples. Depending on the value you use for the ϵ parameter, you might get different results (line thickness). Nevertheless, your results should look quite similar to these.

Line: $y = x + 2$ and parabola: $y = 0.3x^2 + x - 4$

```
 #        ^    #  *
  #       |    # *
  #       |    #*
  #       |    *
  #       |   *
   #      |  *#
   #      | * #
   #      |* ##
   ##     |  #
    #    *| ##
----------.--------->
    ## *  |##
     #*   |##
     *#####|#
    * ####|#
   *  ####|
  *    ###|
 *        |
*         |
          |
          |
```

[3] `http://en.wikipedia.org/wiki/Linear_equation`
[4] `http://en.wikipedia.org/wiki/Parabola`

Line: $y = 2$ and parabola: $y = -0.1x^2 + x + 5$



Question 3: Comprehension (20 points)

Please write your answer in the `Question3.txt` file. Each line is an answer to a question. If you do not know the answer leave that line blank. In other words, the first line refers to question 1, line 2 refers to the question 2, and so on. Each question is worth 4 points each.

a. Would the following code compile?

```
int x = 0;
for(x+=1; x%20 != 0; System.out.println("hello"))
{
}
```

b. What would be displayed on the screen when the following code is run?

```
int W = 212;
int a = 51;
System.out.println("a" + "W");
System.out.println(W + a + "----");
System.out.println(W + "" + a );
System.out.println((W-211)/2);
System.out.println(1/(a-49.0));
```

c. Lets define the following variables,

```
int x = 20;
double d = 3.1416;
String s ="H";
```

Give the **type** the next lines evaluate to, *e.g.* 3+3 evaluates to an int.

```
1. x+20;
2. d/x + "";
3. s + s + x;
4. 1/2;
5. 1.0/2;
6. 1+2.0 + ""
```

d. The four adjacent digits in the 1000-digit number generated by the seed 514419 using `nextInt(9)` that have the greatest product are 7 x 8 x 8 x 6 = 2688. In order to generate the numbers you are to create a random variable as follows, `Random generator = new Random(514419)` and then generate 1000 numbers[5]. In order to operate on them it would be a good idea to store then in an array.

   On that same sequence of numbers, you are to find the 12 digits sequence that have the greatest product and write them down as well as their product value.

e. Each sequence term is generated by adding the previous three terms. By starting with 0, 1 and 1. $T(n) = T(n-1) + T(n-2) + T(n-3)$, where $T(1) = 0$, $T(2) = 1$ and $T(3) = 1$, the first 10 terms are:

   0, 1, 1, 2, 4, 7, 13, 24, 44, 81, ...

   What is the sum of the first 45 terms[6] of this sequence mod the value of the 42^{st} term of this sequence?

What To Submit

You have to submit one zip file that contains all your files to myCourses - Assignment 1. If you do not know how to zip files, please enquire that information from any search engine or friends. Google might be your best friend with this, and for a lot of different little problems as well.

`Question1.java` - Java code
`Question2.java` - Java code
`Question3.txt` - One line: one answer, you can also provide explanations if you wish

`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise he or she would not.

Marking Scheme

Up to 30% can be removed for bad indentation of your code as well as omitting comments, coding structure, or missing files. Marks will be removed as well if the class names are not respected.

[5] If you want to double check the values generated, you can do this on `www.cs.mcgill.ca/~cs202/2014-01/web/A2.html` as well as simply copying and pasting the values in an array.

[6] You should be able to accomplish this using a loop but not mandatory.

Question 1

Used while loops to ask user	5	points
The input is well checked	15	points
The number is well converted	15	points
	35	**points**

Question 2

The variables have different names	5	points
Respect order of printing	5	points
The axis are well drawn	5	points
The line is well drawn	6	points
The parabola is well drawn	8	points
x position refers to horizontal axis and y to the vertical axis	10	points
	45	**points**

Question 3

a.	4	points
b.	4	points
c.	4	points
d.	4	points
e.	4	points
	20	**points**