ASSIGNMENT 3

COMP-202, Winter 2019

Due: Wednesday, March 13^{th}, (23:59)

Please read the entire PDF before starting. You must do this assignment individually.

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, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows 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. :)

Up to 30% can be removed for bad indentation of your code as well as omitting comments, or poor coding structure.

To get full marks, you must:

- Follow all directions below
  - In particular, make sure that all classes and method names are **spelled and capitalized exactly** as described in this document. Otherwise, you will receive a **50% penalty**.
- Make sure that your code compiles
  - **Non-compiling code will receive a 0**.
- Write your name and student ID as a comment in all .java files you hand in
- Indent your code properly
- Name your variables appropriately
  - The purpose of each variable should be obvious from the name
- Comment your work
  - A comment every line is not needed, but there should be enough comments to fully understand your program

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. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Create a method to print the outline of a square made up of * signs. This method must have one parameter, which is the length of the sides in number of *'s. This method should use only two *for loops*, and use *if statements* within the *for loops* to decide whether to draw a space or a star.

Draw the outline of a square as follows:

```
**********
*        *
*        *
*        *
*        *
*        *
*        *
*        *
*        *
**********
```

N.B. It is normal that the square does not appear to be a perfect square on screen as the width and the length of the characters are not equal.

Warm-up Question 2 (0 points)

Change the method you just created to have two parameters, so that you can create rectangles. The first parameter will be the height of the rectangle, and the second parameter will be the width of the rectangle.

Warm-up Question 3 (0 points)

Write a program to display the (x,y) coordinates up to (9,9) of the upper right quadrant of a Cartesian plane. As in the previous warm-up question, your solution should use two nested *for loops*. Your program should also display the axes, by checking to see if the x-coordinate is zero or if the y-coordinate is zero. Note that when both the x and y coordinates are zero, you should print a + character.

For example, the output of your code should look like:

```
^
|(1,9)(2,9)(3,9)(4,9)(5,9)(6,9)(7,9)(8,9)(9,9)
|(1,8)(2,8)(3,8)(4,8)(5,8)(6,8)(7,8)(8,8)(9,8)
|(1,7)(2,7)(3,7)(4,7)(5,7)(6,7)(7,7)(8,7)(9,7)
|(1,6)(2,6)(3,6)(4,6)(5,6)(6,6)(7,6)(8,6)(9,6)
|(1,5)(2,5)(3,5)(4,5)(5,5)(6,5)(7,5)(8,5)(9,5)
|(1,4)(2,4)(3,4)(4,4)(5,4)(6,4)(7,4)(8,4)(9,4)
|(1,3)(2,3)(3,3)(4,3)(5,3)(6,3)(7,3)(8,3)(9,3)
|(1,2)(2,2)(3,2)(4,2)(5,2)(6,2)(7,2)(8,2)(9,2)
|(1,1)(2,1)(3,1)(4,1)(5,1)(6,1)(7,1)(8,1)(9,1)
+----------------------------------------->
```

Note that in the above image, all of the coordinates containing 0's are not displayed, since we are printing axes instead.

Warm-up Question 4 (0 points)

Write a method `generateRandomArray()` that takes as input an integer n and returns an array of size n. The elements of the array should be random doubles between 0 (included) and 5 (excluded). You can use `Math.random()` to do this.

Warm-up Question 5 (0 points)

Write a method `getLargestElement()` that takes an array of integers as input and returns the largest element inside the array. For example, if the input is:
`int[] arr= {1, 5, -3, 15, 4};`
then `getLargestElement(arr)` should return 15.

Warm-up Question 6 (0 points)

Write a method `getSum()` that takes an array of integers as input and returns the sum of all the elements inside the array. For example, if the input is:
`int[] arr= {1, 5,-3, 15, 4};`
then `getSum(arr)` should return 22.

Warm-up Question 7 (0 points)

Write a method `countNegatives()` that takes an array of integers as input and returns the number of negative numbers inside the array. For example, if the input is:
`int[] arr= {1,5,-3, 15, -13};`
then `countNegative(arr)` should return 2.

Warm-up Question 8 (0 points)

Write a method `getVowels()` that takes a String as input and returns an array characters containing all the vowels from the given string. For example, if the input is:
`String s = "kangaroo";`
then `getVowels(s)` should return {'a', 'a', 'o', 'o'}.

Warm-up Question 9 (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 10 (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 22 25
```

Part 2

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

Throughout this assignment you are allowed to use everything we have learned in class up to and including arrays, `Scanner`, and `Random`. This does not mean however that you are allowed to change any of the headers of the methods described below. You need to make sure to follow precisely the instructions provided (for example, you'll be using `Scanner` only in one of the methods for this assignment!).

Question 1: BullsAndCows (60 points)

Bulls and Cows is an old code-breaking mind game. For this question, you will write a Java program that implements a game of Bulls and Cows in which the player needs to guess a randomly generated secret 4-digits number. When the player takes a guess, the program reveals the number of digits that match with the secret number. If the matching digits are in the correct positions, they are called "bulls", if they are in different positions, they are called "cows". After each guess, the program reveals how many bulls and cows the player's guess contains.

To complete this task, you will need to implement all the methods listed below. Note that you are free to write more methods if they help the design or readability of your code. All the code for this question must be placed in a file named `BullsAndCows.java`.

1a) Method to check if an element is contained in a given array

Write a method called `contains()` that takes as input an array of integers and a specific integer. The method returns `true` if the specified integer is an element of the given array, `false` otherwise. For example, let

```
int[] x = {-2, 7};
int[] y = {9, 0, 2, 6};
int[] z = {};
```

Then,

- `contains(x, -3)` returns `false`
- `contains(y, 2)` returns `true`
- `contains(z, 5)` returns `false`

1b) Method to generate the secret number

Write a method called `generateSecretDigits()` that takes as input an integer and returns an array of integers containing the 4 digits that make up the randomly generated secret number. The method uses the input to create an object of type `Random` with the provided seed (remember that to use `Random` you should add the appropriate import statement at the beginning of your file). The method then uses such object to generate a random integer between 0 and 9 (both included). If the integer is not already part of the array, then the method uses it to initialize one of the elements of the array, otherwise a new integer is generated. The process continues until all four elements of the array have been generated. This allows us to generate an array containing four digits that are all different from one another. Make sure to initialize the elements of the array from the first one to the last one.

For example:

- `generateSecretDigits(123)` returns the array {`2, 0, 6, 9`}
- `generateSecretDigits(45)` returns the array {`9, 1, 0, 7`}
- `generateSecretDigits(987)` returns the array {`5, 8, 9, 7`}

1c) Method to extract the digits from a given number

Write a method called `extractDigits()` that takes as input an integer and returns an array containing all the digits of the given number. Note that if the number contains less than 4 digits, then the method returns an array with 4 elements where the missing digits are replaced by 0s and placed at the beginning of the array.

For example,

- `extractDigits(1234)` returns the array {`1, 2, 3, 4`}.
- `extractDigits(75)` returns the array {`0, 0, 7, 5`}.
- `extractDigits(593823)` returns the array {`5, 9, 3, 8, 2, 3`}.
- `extractDigits(-326)` returns the array {`0, 3, 2, 6`}.

1d) Method to get the number of *bulls* in a guess

Write a method called `getNumOfBulls()` that takes as input two integer arrays. The first one represents the digits of the secret number, while the second one represents the digits of the number guessed by the player. The method returns the number of bulls in the guess of the players (see the intro paragraph for the definition of *bull* in this context). If the input arrays have a different number of elements, then the method should throw an `IllegalArgumentException` with an appropriate message.

For example, consider the following arrays:

```
int[] secret = {2, 0, 6, 9};
int[] guessOne = {9, 5, 6, 2};
int[] guessTwo = {2, 0, 6, 2};
int[] guessThree = {1, 2, 3, 4, 5, 6};
int[] guessFour = {1, 3, 4, 4, 0, 5};
```

Then:

- `getNumOfBulls(secret, guessOne)` returns `1`.
- `getNumOfBulls(secret, guessTwo)` returns `3`.
- `getNumOfBulls(secret, guessThree)` raises an exception.
- `getNumOfBulls(guessThree, guessFour)` returns `2`.

1e) Method to get the number of *cows* in a guess

Write a method called `getNumOfCows()` that takes as input two integer arrays. The first one represents the digits of the secret number, while the second one represents the digits of the number guessed by the player. The method returns the number of cows in the guess of the players (see the intro paragraph for the definition of *cow* in this context). If the input arrays have a different number of elements, then the method should throw an `IllegalArgumentException` with an appropriate message.

For example, consider the following arrays:

```
int[] secret = {2, 0, 6, 9};
int[] guessOne = {9, 5, 6, 2};
int[] guessTwo = {2, 0, 6, 2};
int[] guessThree = {1, 2, 3, 4, 5, 6};
int[] guessFour = {1, 3, 4, 4, 0, 5};
```

Then:

- `getNumOfCows(secret, guessOne)` returns `2`.

- `getNumOfCows(secret, guessTwo)` returns `0`.
- `getNumOfCows(secret, guessThree)` raises an exception.
- `getNumOfCows(guessThree, guessFour)` returns `3`.

1f) Method to simulate a game

Write a method `playBullsAndCows()` which takes an integer as input and simulates a game of Bulls and Cows. The method creates *one* object of type `Scanner` to retrieve the inputs from the user (remember that to use `Scanner` you should add the appropriate import statement at the beginning of your file). The method should then do the following:

1. Generate a random secret number to be guessed using `generateSecretDigits()` and the integer received as input.
2. Display a welcome message to the player. It is up to you to decide how to welcome the users of your program.
3. Display a message encouraging the player to make a guess. The message should also display the guess attempt.
4. The method retrieves the guess as an integer. Here you can assume that the player will enter an input of the correct type.
   - If the player inputs a negative integer or an integer with more than 5 digits, then the method displays a message letting them know they wasted one guess and that they should enter a positive integer with at most 4 digits.
   - Otherwise, the method displays how many bulls and cows the player's guess contains.
5. If the player guessed the secret number, then the method congratulates the player and let them know how many tries were needed to win the game.
6. If the player has not guessed the secret number after 5 guesses, then the method should start to ask the player if they want to quit the game at each incorrect guess. If the player replies with a 'y' (which is retrieved as a String) the method should displays a message acknowledging that the player wanted to quit and letting them know how many times they tried to guess the secret number.
7. If the player did not guess the number nor decided to quit, then the method goes back to step three.

To complete the program, simply call `playBullsAndCows()` from the `main` method providing an integer of your choice as input. Note that the secret number generated depends on that integer. If you want to test your program with different numbers you should change the integer provided as input. In fact, to play the game you can generate a random integer inside the main method (using `Math.random()`) and provide such integer as input to `playBullsAndCows()`. To debug your program, we highly suggest you fix a specific integer so that it will be easier to understand whether your method is behaving as it should.

Note that you can add additional features of your choice to the game. For example your game could do the following:

- Your method could ask for the name of the player.
- Your method could display different messages depending on the number guess attempts.
- Your method could use a different layout.
- Your method could display the number of seconds the game was played for (if you'd like to do that, you can use `System.currentTimeMillis()`).

- Any other thing you can think of as long as you don't use any another class beside `Scanner`, `Random`, and those that belongs to *java.lang* package.

Note that none of the above features is needed (you won't receive extra marks for it), but it might be fun to make the game a little more original. :)

Below you can find a couple of sample runs of the program. Note that our program displays the number of seconds played, but yours does not have to do that. In all of the examples, the call made from the main method was `playBullsAndCows(123)` (this means that the secret number to guess is `2069`).

Examples

```
> run BullsAndCows
Hey, you! Want to play a game?
Do you think you can crack the code?

Guess #1. Enter a four digit number: 1234
Bulls: 0, Cows: 1

Guess #2. Enter a four digit number: 1298
Bulls: 0, Cows: 2

Guess #3. Enter a four digit number: -7
You must enter a positive integer with at most four digits. Try again!

Guess #4. Enter a four digit number: 567890
You must enter a positive integer with at most four digits. Try again!

Guess #5. Enter a four digit number: 2908
Bulls: 1, Cows: 2

Might be time for you to give up. Do you want to quit? (y/n) n

Guess #6. Enter a four digit number: 0296
Bulls: 0, Cows: 4

Might be time for you to give up. Do you want to quit? (y/n) n

Guess #7. Enter a four digit number: 2069
Bulls: 4, Cows: 0

Congratulations! You cracked the code in only 7 tries. 47 seconds well invested!
```

```
> run BullsAndCows
Hey, you! Want to play a game?
Do you think you can crack the code?

Guess #1. Enter a four digit number: 89
Bulls: 2, Cows: 0

Guess #2. Enter a four digit number: 9001
Bulls: 1, Cows: 1

Guess #3. Enter a four digit number: 62
Bulls: 2, Cows: 1

Guess #4. Enter a four digit number: 1234
Bulls: 0, Cows: 1

Guess #5. Enter a four digit number: 9876
Bulls: 0, Cows: 2

Might be time for you to give up. Do you want to quit? (y/n) n

Guess #6. Enter a four digit number: 3476
Bulls: 0, Cows: 1

Might be time for you to give up. Do you want to quit? (y/n) n

Guess #7. Enter a four digit number: 6543271
You must enter a positive integer with at most four digits. Try again!

Might be time for you to give up. Do you want to quit? (y/n) you quit!

Guess #8. Enter a four digit number: 6500
Bulls: 0, Cows: 2

Might be time for you to give up. Do you want to quit? (y/n) maybe...

Guess #9. Enter a four digit number: 9602
Bulls: 0, Cows: 4

Might be time for you to give up. Do you want to quit? (y/n) alright

Guess #10. Enter a four digit number: 7777
Bulls: 0, Cows: 0

Might be time for you to give up. Do you want to quit? (y/n) y

You gave up after 10 tries. :( You wasted 117 seconds of your precious time!
```

Question 2: Polynomial functions (40 points)

For this question you will write a program which allows you to draw curves corresponding to functions of the form

$$y = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where the a_i's are decimal numbers. Although the general formula might look quite complicated, particular examples are much simpler. For instance, the following equation represents a constant function which is a polynomial function of degree 0:

$$y = 4$$

The following equation represents a line which is a polynomial function of degree 1:

$$y = x + 2$$

The following equation represents a parabola which is a polynomial function of degree 2:

$$y = 0.1x^2 - x - 8$$

And finally, the following equation represents a cubic which is a polynomial function of degree 3:

$$y = \frac{1}{25}x^3 + 0.1x + 0.5$$

You will be able to draw the graphs of the functions using print statements, conditional statements, and loops. The graphs should be drawn on a Cartesian plane with four quadrants, where each quadrant has at least a 10 by 10 grid size. To achieve this, write the following three methods. Note that you are free to write more methods if they help the design or readability of your code. The code for this question must be placed in the file named `PolynomialCurves.java`

2a. A method called `onCurve`

Write a method called `onCurve()` that takes 3 inputs: an integer array representing the coordinates of a point on the plane, a double array representing the coefficients of the polynomial function you want to draw, and a double representing the desired thickness of the curve. The method determines whether or not the given point should be considered to be part of the specified curve given the desired thickness.

In general, given the coefficients of a polynomial function $\{a_n, a_{n-1}, \ldots, a_1, a_0\}$, a specific point (x_0, y_0) is considered to be part of the curve if the following equation holds:

$$y_0 = a_n x_0^n + a_{n-1} x_0^{n-1} + \cdots + a_1 x_0 + a_0$$

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. Instead of checking if the equality is satisfied, you should check if the following inequalities are satisfied

$$y_0 - t < a_n x_0^n + a_{n-1} x_0^{n-1} + \cdots + a_1 x_0 + a_0 < y_0 + t$$

where t represents the *thickness* of the curve being drawn.

Thus, to recap, **the method receives as input the coordinates of a point ($\{x_0, y_0\}$), the coefficients of a curve ($\{a_n, a_{n-1}, \ldots, a_1, a_0\}$), and a desired thickness (t). The method returns `true` if the above inequalities are satisfied, `false` otherwise**.

Consider the following variables

```
double[] line = {1.0, 2}; // y = x + 2
double[] parabola = {0.1, -1, -8}; // y = 0.1x^2 - x - 8
int[] pointOne = {0, 2};
int[] pointTwo = {0, -8};
int[] pointThree = {-5, -1};
int[] pointFour = {3, 5};
double lineThickness = 1;
double parabolaThickness = 1.05;
```

Then,

- `onCurve(pointOne, line, lineThickness)` returns `true`
- `onCurve(pointTwo, line, lineThickness)` returns `false`
- `onCurve(pointThree, line, lineThickness)` returns `false`
- `onCurve(pointFour, line, lineThickness)` returns `true`
- `onCurve(pointOne, parabola, parabolaThickness)` returns `false`
- `onCurve(pointTwo, parabola, parabolaThickness)` returns `true`
- `onCurve(pointThree, parabola, parabolaThickness)` returns `true`
- `onCurve(pointFour, parabola, parabolaThickness)` returns `false`

2b. A method to verify the inputs

Write a method called `verifyInput()` which takes two inputs: an array of doubles representing the coefficients of a polynomial curve, and a double representing the desired thickness of the curve. The method should throw an `IllegalArgumentException` (and display a helpful error message) if the array does not have at least one element, or if the second input is not a positive number. Note that the messages should be different depending on the issue. Otherwise, the method will not do anything. Note that this means in particular that the method does not return any value.

2c. A method to draw the graph of a polynomial curve

Write a method called `drawCurve()` that takes 3 inputs: an array of doubles representing the coefficients of a polynomial curve, a double representing the desired thickness of the curve, and a character representing the symbol with which the curve will be drawn. Both of the methods described above must be called and used appropriately in this method in order to get full points.

The method first verifies that the inputs received are valid and if so, the method proceeds to draw the specified curve.

Remember that the graph of the curve should be drawn on a Cartesian plane with four quadrants, where each quadrant has at least a 10 by 10 grid size.

The rules for displaying the axes (the x-axis and the y-axis) are as follows: to represent the x-axis dashes (`-`) should be used, to represent the y-axis vertical bars (`|`) should be used. The y-axis (in the positive direction) should end in a hat symbol (`^`), and the x-axis (in the positive direction) should end in a right triangle bracket (`>`). You should use the plus symbol (`+`) to represent the origin. This means that there are a total of 21 characters on the x-axis, and 21 characters on the y-axis, with the origin counting as being on both.

Note that, we want the y-axis to extend at least 5 points above and below the y-intercept of the curve (the point on the curve with x-coordinate 0). Therefore, if this does not happen naturally with the set up of the grid, then the y-axis should be extended accordingly.

Finally, if more than one character should be drawn at the same location (eg, if the curve intersects with an axis), the curve gets priority.

To achieve all 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 determine what to draw at that particular position. Pay attention to the order of importance for drawing the symbols [1].

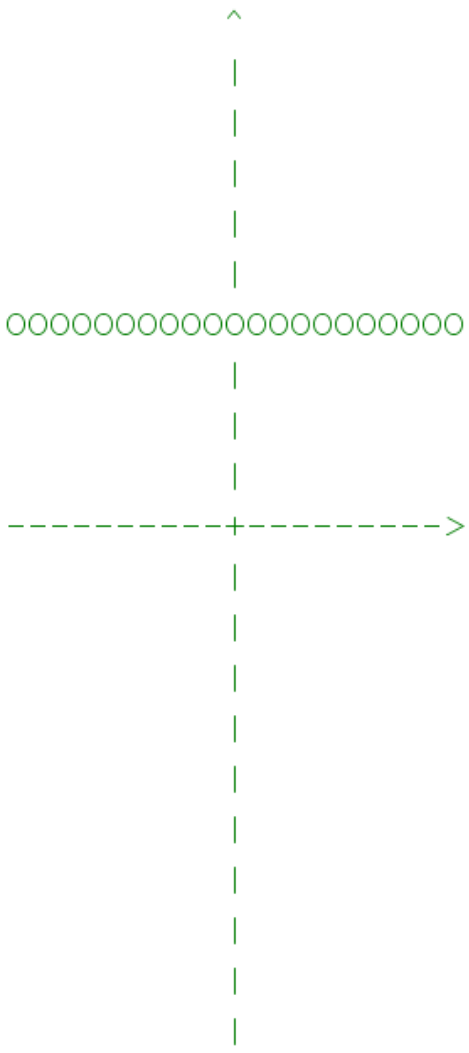
Your main method will not be graded. However, we strongly advise that you write a main method in order to test your code. You may include it in your submission as long as it compiles.

[1] e.g. The axis arrow head `>` has higher importance than the axis lines `-`.

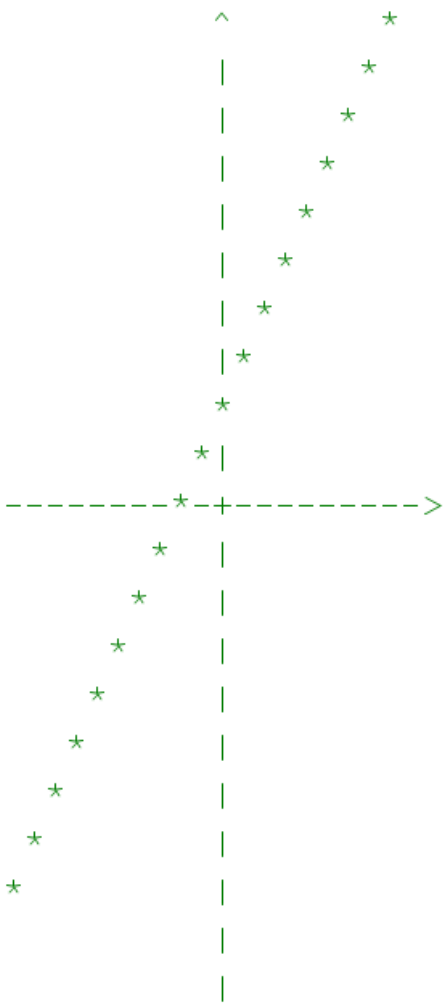
You can find below what would be displayed when each snippet of codes is executed.

Examples

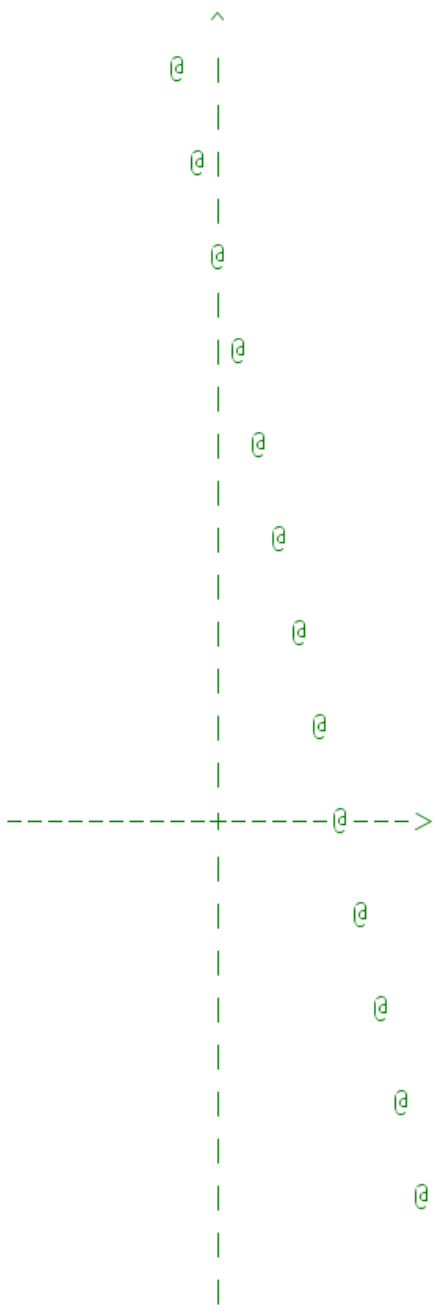
```
double[] constant = {4}; // y = 4
drawCurve(constant, 1, 'O');
```



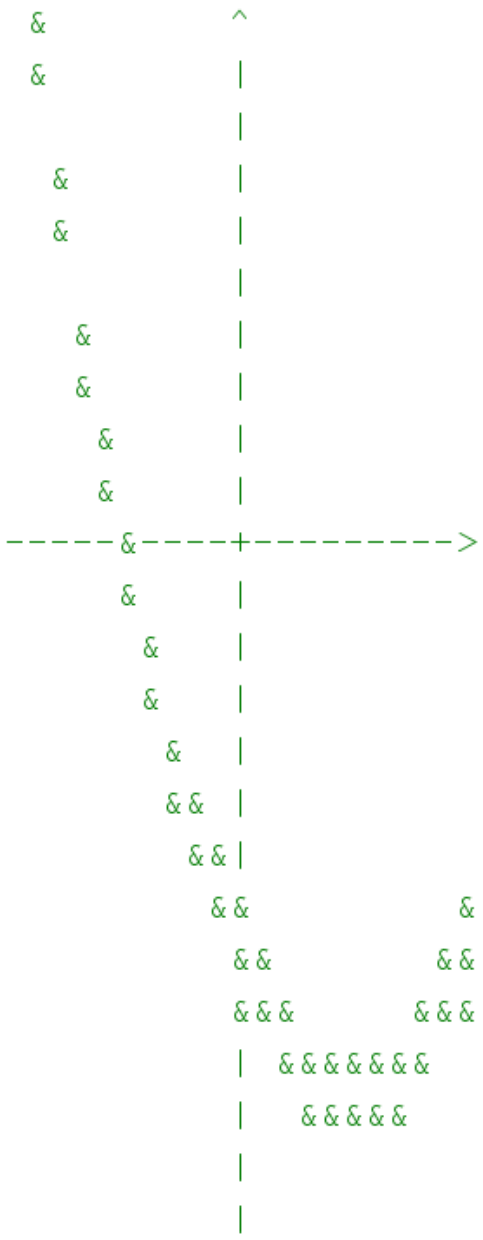
```
double[] lineOne = {1.0, 2}; // y = x + 2
drawCurve(lineOne, 1, '*');
```



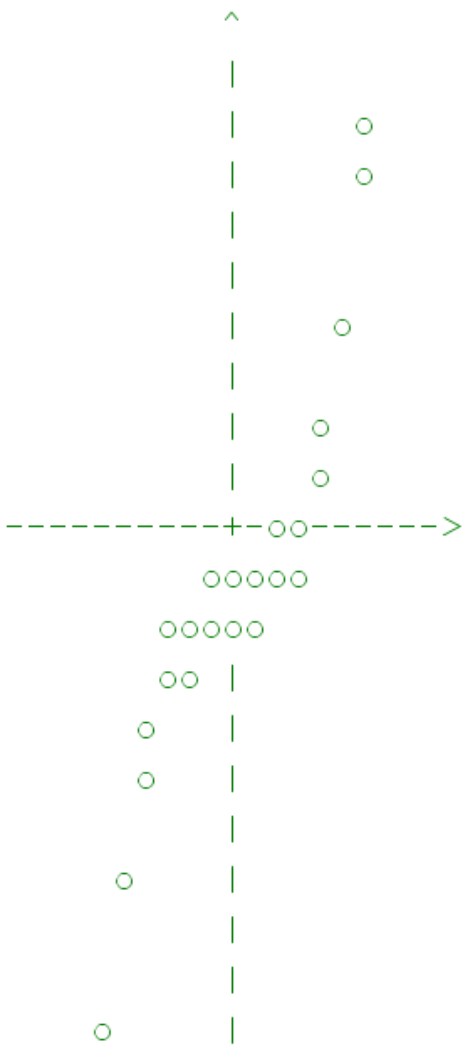
```
double[] lineTwo = {-2.0, 12}; // y = -2x + 12
drawCurve(lineTwo, 1, '@');
```



```
double[] parabola = {0.1, -1, -8}; // y = 0.1x^2 - x - 8
drawCurve(parabola, 1.05, '&');
```



```
double[] cubic = {1.0/25, 0, 0.1, -1.5};
drawCurve(cubic, 1, 'o');
```



What To Submit

Please put all your files in a folder called Assignment3. Zip the folder (DO NOT RAR it) and submit it in MyCourses.

Inside your zipped folder, there must be the following files. **Do not submit any other files, especially .class files.** Any deviation from these requirements may lead to lost marks.

`BullsAndCows.java`
`PolynomialCurves.java`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment.