Assignment 1 : Robots!

COMP-202, Fall 2016 (all sections)

Due: September 23^{rd}, 2016 - 11:55pm

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

Part A, Question 1:	20 points
Part A, Question 2:	40 points
Part B:	40 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, or if your solution does not follow the assignment specifications. Graders have the discretion to deduct up to 15% of the value of this assignment for not adhering to the best practices, as discussed in class and on the syllabus.

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 file called `HelloWorld.java`, and in this file, declare a class called `HelloWorld`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display "`Hello world!`". You can find such a class in the lecture slides; make sure you can compile and run it properly.

Warm-up Question 2 (0 points)

Create a file called `A.java`, and in this file, declare a class called `A`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display the following pattern:

```
    A
   A A
  AAAAA
 A     A
A       A
```

Warm-up Question 3 (0 points)

This question is designed to help you practice with types.

Background: As discussed in the lectures, there are different types of values in programming languages such as Java. For instance, we can have integers like the number 5, real numbers like the number 5.1, boolean values like true and false, characters like 'm', or more complex types like Strings e.g. "we all live in a yellow submarine". We can compare, contrast, display and apply operators to values of these different types.

Recall also that we can assign values to variables that are declared with the appropriate type: char for storing characters, int for integers, double for real numbers, boolean for logical values, and several others.

When you declare a variable in Java, you need to specify the type. This involves thinking about a real world piece of data and figuring out what type you should use.

For each of the following, what type would best represent the data? Are there any that cannot be represented in Java?

1. The name of a day in the week.
2. Your letter grade in the course.
3. Your numeric grade in the course (as a percentage).
4. The name of the planet that your COMP 202 teacher is from.
5. The temperature outside.
6. The mathematical number π
7. The name of the author who wrote the Hitchhiker's Guide to the Galaxy?
8. The (estimated) number of molecules in the universe.
9. The result of multiplying two integers together
10. The result of dividing an integer by another integer.

11. "Is the date today Sept 11?"
12. Does this string "i am happy" start with the character 'i'?

Warm-up Question 4 (0 points)

The following code is designed to put the value of the variable x into the variable y and the value of the variable y into x. What is wrong with it?

```
public class BadSwap {
     public static void main(String[] args) {
           int x = 1;
           int y = 2;
           x = y;
           y = x;
     }
}
```

Warm-up Question 5 (0 points)

Write a program `Adder` that takes as input, using command line arguments (args[0] and args[1]) 2 numbers and adds them together. It should print the resulting value to the screen. See *mycourses* for details on accepting input via the command line.

Part 2: Graded Assignment

The questions in this part of the assignment will be graded.
You are expected to properly indent your code, and to add comments where appropriate. Marks will be deducted for badly formatted and uncommented code.

Part A: Robot City! (60 points)

For this question you have to get a robot (represented by a coloured triangle) to navigate a maze and collect flashing lights. You are provided with a model of the world using cities, robots and flashing lights. Instead of creating your own code from scratch, you will be using software that has already been developed by Byron Becker at the University of Waterloo. (Additional information about the project can be found at `http://www.learningwithrobots.com`)

To complete the assignment, you will need the following files.

- `a4-becker.jar` This is an archive file which contains all of the additional Java classes that you will need for the assignment and is posted on mycourses.
- `RobotsMoveLightsSmall.java`
- `RobotsMoveLightsGreatEscape.java`

In order to get the provided files `RobotsMoveLightsSmall.java` and `RobotsMoveLightsGreatEscape.java` to compile and then to run the programs, you need to tell your computer where to find the `a4-becker.jar` file.

In Dr Java: Go into Edit and then click on Preferences. Then click on Resource Locations. Under 'Extra Classpath' click on Add. You then have to find the location of the `a4-becker.jar` file in your computer, select it and the click on 'select'. If you did not move the file, it should be in your 'Downloads' folder. You then click on Apply, and then Okay. You may have to quit DrJava and reopen it for the changes to take effect. Now that you have set up the `a4-becker.jar` file, you should be able to compile and execute RobotsMoveLightsSmall.java.

When you run the file, you should see a picture that looks like Figure 1.

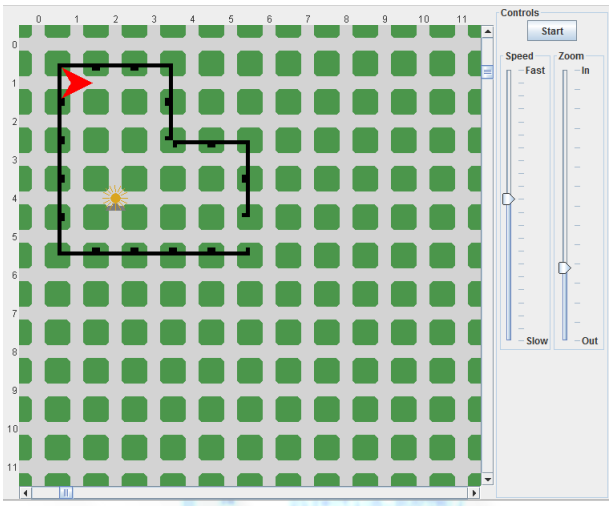

Figure 1: Small Escape Task

This picture is a representation of a Robot (the small triangle) living in a City with very organized blocks (like New York :)). In this city, there is a flashing light. There are also black walls that the robot cannot walk through.

The follow are commands that you can use to make the robot do things:

- `robot.move()` This causes the robot to move forwards one block. If this would cause the robot to crash into a wall, the robot will break and your program will crash.
- `robot.turnLeft()` This causes the robot to turn 90 degrees to the left.
- `robot.pickThing()` This causes the robot to pick up whatever object shares its location. Once the robot is on the same space as the flashing light, it can use this command to pick it up. If there is nothing on this spot, then the robot will break. This will cause your program to crash.

You'll notice there is no `turnRight`. You'll need to turn left several times. (It is the opposite of the character from the movie Zoolander. :))

Question 1: Small Escape (20 points)

Add to the file `RobotsMoveLightsSmall.java` to make the robot navigate to the light in the middle and pick up the light. After picking up the light, the robot should spin in a circle (to do a victory dance), and then exit the maze at the bottom right.

You will add your commands inside of the main method in the RobotsMoveLightsSmall class. When you run your program, nothing will happen until you press the `Start` button on the top right corner of the picture. You can speed up or slow down the robot's movement with the sliding controls.

All of the commands that you put inside of the method will execute back-to-back, from top to bottom. There is no way to change the commands while your program is running.

Question 2: Great Escape (40 points)

Once you have gotten the small escape program to work, you are ready to write code for the great escape! Open the file `RobotsMoveLightsGreatEscape.java`. When you run this file, the picture will now look like Figure 2.

You now have a more complicated problem. This new world has three flashing lights and additional walls.

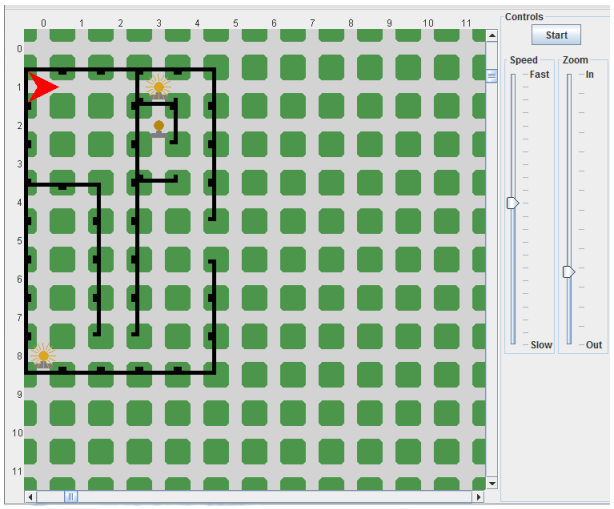

Figure 2: Great Escape Task

Your task is still basically the same: the robot has to pick up all of the lights and then escape via the exit on the right-hand wall.

Part B: Your Own Program (40 points)

The purpose of this question is to give you practice writing a complete program from beginning to end. You will also get practice with accepting arguments via the command line, with manipulating numbers, and using library utility methods. Note: See mycourses *for detailed instructions on accepting and using command line arguments in Dr. Java.*

For this part of the assignment, you will write your own program from start to finish. This program must be saved in a file called `RaceTrack.java`.

You are writing a program for a contractor who sells asphalt for racetracks. The asphalt is sold at 7.49$ per square meter. For the purposes of this assignment, all racetracks are perfectly circular.

This program accepts as input via `args[0]` the large radius of the track in meters. Via `args[1]`, it accepts the number of lanes in the track (see Figure 3 for a sample racetrack). The radius can be any positive number - the numbers of lanes must be a *whole* number.

Each lane is 2.8 meters wide. The program computes the cost of the asphalt needed to pave the track. It then prints the total bill in the following format. (Note: this data is generated when `args[0]` is `10.5` and `args[1]` is 3.

```
Total area: 332.5061664559437
Subtotal: 2490.471186755018$
Sales Tax (15%): 373.5706780132527$
Total: 2865$
```

Ask Google (or Bing) if you have forgotten how to compute the area of a circle. Use `Math.PI` to get the value of the mathematical constant π.

Treat sales tax in Quebec as a flat 15.0% (for both GST and QST).

Due to the present high demand, the contractor doesn't accept payment of fractions of a dollar. After computing the total cost (including tax), he *rounds up*. Do this using the `Math.ceil()` method.

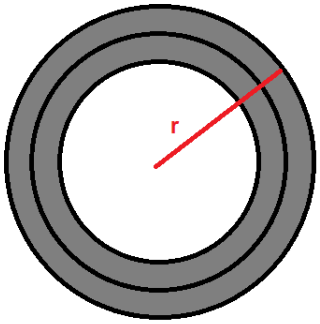

Figure 3: A racetrack with two lanes and radius r

Your program can assume that it is only given valid input (you don't have to worry about negative numbers, non-numerical values, a radius that is too small to accommodate the number of lanes requested, etc).

What To Submit

You have to submit one zip file called `Assignment1_YourName.zip` with all your files in it to MyCourses under Assignment 1. If you do not know how to zip files, please ask any search engine or friends. Google might be your best friend with this, and a lot of different little problems as well.
`RobotsMoveLightsSmall.java`
`RobotsMoveLightsGreatEscape.java`
`RaceTrack.java`