ASSIGNMENT 3

COMP-202, Winter 2016, All Sections

Due: March 18^{th}, 2016 (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 15% 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:	60 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.

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)

Write a method that takes an input an array of integer arrays and checks if all of the numbers in each 'sub-array' are the same. For example, if the input is:

$$\{\{1,1,1\},\{6,6\}\}$$

then it should return true and if the input is:

$$\{\{1,6,1\},\{6,6\}\}$$

it should return false.

Warm-up Question 2 (0 points)

Write a method that takes as input an array of double arrays and returns the double array with the largest *average* value. For example, if the input is:

$$\{\{1.5,2.3,5.7\},\{12.5,-50.25\}\}$$

then it should return the array $\{1.5, 2.3, 5.7\}$ (average value is 3.17).

Warm-up Question 3 (0 points)

Write a class describing a Cat object. A cat has the following `attributes`: a name (String), a breed (String), an age (int) and a mood (enum Mood). The cat `constructor` takes as input a String and sets that value to be the breed. The mood of a cat can be one of the following: `sleepy, hungry, angry, happy, crazy`. The `Cat` class also contains a method called `talk()`. This method takes no input and returns nothing. Depending on the mood of the cat, it prints something different. If the cat's mood is `sleepy`, it prints *meow*. If the mood is `hungry`, it prints *RAWR!*. If the cat is `angry`, it prints *hssssss*. If the cat is `happy` it prints *purrrr*. If the cat is `crazy`, it prints a random String of between 10 and 25 characters (letters).

The cat `attributes` are all **private**. Each one has a corresponding **public** method called `getAttributeName()` (ie: `getName(), getMood()`, etc.) which returns the value of the `attribute`. All but the `breed` also have a **public** method called `setAttributeName()` which takes as input a value of the type of the attribute and sets the attribute to that value. Be sure that only valid mood sets are permitted. (ie, a cat's mood can only be one of five things). There is no setBreed() method because the breed of a cat is set at birth and cannot change.

Test your class in another file which contains only a main method. Test all methods to make sure they work as expected.

Warm-up Question 4 (0 points)

Write a class `Vector`. A `Vector` should consist of three `private` properties of type double: x,y, and z. You should add to your class a constructor which takes as input 3 doubles. These doubles should be assigned to x,y, and z. You should then write methods `getX()`, `getY()`, `getZ()`, `setX()`, `setY()`, and `setZ()` which allow you to get and set the values of the vector.

Warm-up Question 5 (0 points)

Add to your `Vector` class a method calculateMagnitude() which returns a double representing the magnitude of the vector. The magnitude can be computed by taking

$$\sqrt{x^2 + y^2 + z^2}$$

Warm-up Question 6 (0 points)

Write a method `scalarMultiply` which takes as input a `double[]`, and a `double scale`, and returns `void`. The method should modify the input array by multiplying each value in the array by `scale`. Question to consider: Would this approach work if we had a `double` as input instead of a `double[]`?

Warm-up Question 7 (0 points)

Write a method `deleteElement` which takes as input an `int[]` and an `int target` and deletes all occurrences of `target` from the array. The method should return the new `int[]`. Question to consider: Why is it that we have to return an array and can't simply change the input parameter array?

Warm-up Question 8 (0 points)

Write the same method, except this time it should take as input a `String[]` and a `String`. What is different about this than the previous method? (Hint: Remember that `String` is a reference type.

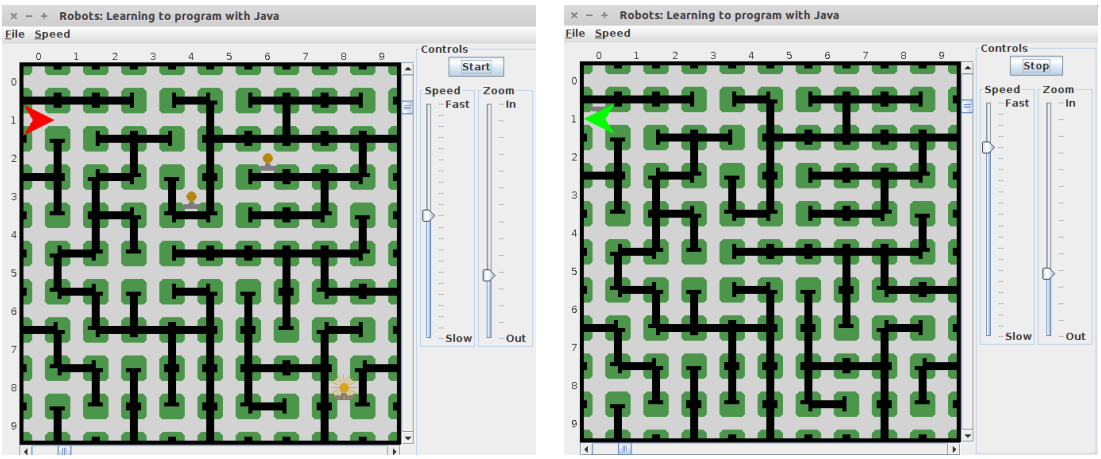
Assignment

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

Part 2:

Question 1: Intro to Objects: Robot City (40 points)

For this question you have to get a robot (represented by a coloured triangle) to navigate a maze and



(a) Initial Position

(b) Ending Positions

Figure 1: Result of the `getLoot()` method

collect flashing lights. You are provided with an object-oriented model of the world using cities, robots and flashing lights. Instead of creating your own classes you will be using software that has already been developed by Byron Becker at the University of Waterloo. Before you begin, please review the application programming interface (API) for the classes provided in the `becker.robots` package. That is, you need to read the documentation in order to figure out which methods are available to you. 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.
- `a4-becker-docs.zip` This is an archive file which contains documentation about the classes included in the provided jar file.

All the above files can also be found at: `http://www.learningwithrobots.com/software/downloads.html` Additional information on this package can be found at: `http://www.learningwithrobots.com/doc/becker/robots/package-summary.html`. In order to get the provided file `RobotsMoveLights.java` to compile, and to use the robot objects, you need to tell your computer where to find the `a4-becker.jar` file.

- Eclipse Users: To do this in Eclipse, right click on your project folder (the first folder in the Package Explorer pane on the left hand side of the screen). Go down to 'Build Path' and then click on 'Add External Archives...'. You then have to find the location of the `a4-becker.jar` file, select it, and click 'open'. If you did not move the file, it should be in your 'Downloads' folder.
- Dr Java Users: 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 RobotsMoveLights.java. When you execute the program, it should display an image similar to Figure 1(a). Note that the mazes are randomly generated, so it is normal that the layout of your maze is different from the one in the photo.
Your task is to give the robot (the coloured triangle) a series of commands such that at the end your image will look like Figure 1(b). In Figure 1(b), it doesn't matter if your robot is facing a different direction, as long as it is in the correct location. You should not have to modify the main method in this file - just the `getLoot()` method. This method takes as input a reference to a `Robot` object. Initially,

there are three flashers placed throughout the maze. The robot has to navigate the maze, collect the flashers, and bring them back to its original position. Once the robot has picked up the last flasher, it must change colour before 'going home'. In the example, it changes from red to green, although you may choose any colour change that you like.

It is important to note that no matter what specific maze is generated, every square can always be reached via a standard 'follow the right wall' method of exploration. It is therefore *required* that you program the robot to find the flashers using this technique.

For details, see `https://en.wikipedia.org/wiki/Maze_solving_algorithm#Wall_follower`

Some methods that might be useful to you are: `robot.move()`, `robot.turnLeft()`, `robot.pickThing()`, `robot.getIntersection()`. In order to get full marks, you will have to use others as well. This list should just help get you started. Your method must work regardless of the exact location of flashers and the formation of the maze. Note that it is recommended to write small 'helper' methods if you wish. For example, there is no `robot.turnRight()` method. You may want to write your own method that does this. This is completely optional.

Question 2: Making your own Objects: Hangman (60 points)

For this question, you will create your own objects in order to implement the game of hangman. Hangman is a guessing game for two players, or a player and a computer. One player (or a computer) thinks of (generates) a word, phrase or sentence and the other tries to guess it by suggesting letters. The guessing player has a limit on the maximum number of incorrect guesses she can make before losing the game. She knows how many letters are in the word, and is told which letters are correct as she make guesses. She also has access to the list of letters she has already guessed.

1. For the first part, you will create a `Letter` class. A letter in hangman has two `private` attributes:

   (a) A `char` value, `value`, indicating the character value of the letter.

   (b) A `boolean` value, `isGuessed`, indicating whether or not its value has been guessed by the player.

   The `Letter` constructor takes as input the character of the letter and sets `value` accordingly. It also sets the default value of `isGuessed` to be false.
   This class also has the following `public` methods:

   - `getValue()` This method returns the value of the `value` attribute.
   - `getRevealed()` This method returns the value of the `isGuessed` attribute.
   - `reveal()` This method changes the value of isGuessed from `false` to `true`.

   Be sure to test your `Letter` class thoroughly before moving on to the next part!

2. In the next part, you will create the `Hangman` class. This class has the following `private` attributes:

   - An array of `Letter` values called `letters`.
   - A `char` array called `guesses`. This array will store (at least) all of the *incorrect*, valid characters guessed by the player.
   - A `boolean`, `gameOver`, indicating whether or not the game is over.
   - An `int`, `maxNumGuesses`, which stores the maximum number of incorrect guesses permitted.
   - An `int`, `numGuessesMade`, which keeps track of the number of *incorrect* guesses made by the player.

The `Hangman` class has the following `public` methods:

- A constructor that takes as input a String corresponding to the mystery word. It **converts the String to uppercase** and then sets all of the values in the `letters` array to the characters of the input String. This constructor also sets the maximum numbers of incorrect guesses to a default value of eight.
- A constructor that takes as input a String corresponding to the mystery word, as well as the maximum number of incorrect guesses. It then sets the `letters` and `maxNumGuesses` attributes accordingly.
- A playGame method that takes no input, returns nothing, and plays a complete game of hangman from start to finish.

Note: in the constructors, you may assume that the input String contains only valid characters. Additionally, be sure to initialize the `guesses` array to a reasonable size, as well as initialize all of the other private attributes.
You will also need to write the following `private` methods:

- A method called `guess` that takes as input a character and returns a boolean value indicating whether or not the guess was *valid* (invalid guesses include symbols, and incorrect characters that have already been guessed). If it is valid and incorrect, this method will convert the character to uppercase, and add the character to the `guesses` array. If it is valid and correct, it will update the

boolean `isGuessed` attribute of any applicable Letter values in the `letters` array. If applicable, it will also update the value of the `numGuessesMade` attribute.

- A method called `displayBoard` that prints the following information: The valid incorrect guesses made thus far, the (partially revealed) word, and the number of remaining guesses.

You may write any additional helper methods that you deem appropriate. Ensure that any such methods you write are `private`.

When the game finishes, you should either indicate to the player that they have won, or let them know that they have lost. In both cases, you should print the full mystery word.

A sample run-through of the game is shown on the last page of this assignment. You are welcome to change the phrasing of the dialogue, as long as you include all relevant information.

What To Submit

You have to submit one zip file with all your files in it to MyCourses under Assignment 3. 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.
`RobotsMoveLights.java`
`Letter.java`
`Hangman.java`
Optional: confessions.txt

```
Welcome to DrJava.  Working directory is C:\Users\
> run Hangman
You have made the following guesses:
Word so far: _ _ _ _ _ _ _ _ _ _ _ _ _ _
You have 8 remaining.
What is your next guess?
o
You have made the following guesses:
Word so far: _ O _ _ _ _ O _ _ _ O _ _ O
You have 8 remaining.
What is your next guess?
h
You have made the following guesses: H
Word so far: _ O _ _ _ _ O _ _ _ O _ _ O
You have 7 remaining.
What is your next guess?
c
You have made the following guesses: H
Word so far: C O _ _ _ _ O _ _ _ O _ _ O
You have 7 remaining.
What is your next guess?
o
What is your next guess?
o
You have made the following guesses: H
Word so far: C O _ _ _ _ O _ _ _ O _ _ O
You have 7 remaining.
What is your next guess?
&
Invalid guess. Try again
m
You have made the following guesses: H
Word so far: C O M _ _ _ O _ _ _ O _ _ O
You have 7 remaining.
What is your next guess?
p
You have made the following guesses: H
Word so far: C O M P _ _ O _ _ _ O _ _ O
You have 7 remaining.
What is your next guess?
t
You have made the following guesses: H
Word so far: C O M P T _ O _ _ _ O T _ O
You have 7 remaining.
What is your next guess?
w
What is your next guess?
w
You have made the following guesses: H
Word so far: C O M P T W O _ _ _ O T W O
You have 7 remaining.
What is your next guess?
z
You have made the following guesses: H
Word so far: C O M P T W O Z _ _ O T W O
You have 7 remaining.
What is your next guess?
e
You have made the following guesses: H
Word so far: C O M P T W O Z E _ O T W O
You have 7 remaining.
What is your next guess?
r
You got the word!
C O M P T W O Z E R O T W O
```