UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

Basic Java 2 - Roomba Simulator

Nada Basit and Mark Floryan

January 13, 2022

1 SUMMARY

A while ago, I purchased a Roomba, a small round robot that scurries along my floor and vacuums (surely you've heard of these by now as they've become relatively common). The robot is quite cute, and I always wonder precisely how the algorithm for determining its actions work. For this homework, you will write some code that controls a virtual Roomba vacuum robot, and try to construct an algorithm that best cleans the floors of some virtual rooms. Your summary:

1. Download the starter code and import the project into Eclipse.
2. Implement the makeMove method in the Roomba class.
3. **FILES TO DOWNLOAD:** Roomba.zip
4. **FILES TO SUBMIT:** MyRoomba.java

1.1 DOWNLOAD AND REVIEW CONTENTS OF STARTER CODE

You can download the starter code for this project from the course repository here. Once you have done so, you should import the project into Eclipse.

Importing a project in Eclipse is easy. First, unzip the starter code somewhere on your machine. Then, within Eclipse, simply navigate to the **File –> Import** menu. Several options will appear next. Select **Existing Projects into Workspace** and click *Next*. Another dialog will appear asking you for the path to the folder where the project exists. Press **Browse** next to **Select Root Directory** and navigate to the project folder. Then, simply click **Finish** and you should see the project appear within the left hand bar of Eclipse. **Note: Your version of Eclipse might differ slightly, but the process should be similar / very close to what is listed above.**

There are several Classes that already exist in the project, that we will enumerate here:

- package **main**
  - *Main.java*: Contains the main method. Instantiates a room, determines the room size, creates a Roomba with initial location in room, and starts the simulation. **You DO NOT need to change anything in this file, unless you want to tweak the numbers to change parameters of the similarity. Play around with it if you want to!**
  - *MyRoomba.java*: Your primary task is to **write the makeMove() method** in this class. See more details below.
- package **world**
  - *Move.java*: An enum that lists the valid moves a Roomba can make (move forward, turn clockwise, or turn counterclockwise). The makeMove() method should return one of these.
  - *RoomTile.java*: The different types of tiles that can exist in a room. A tile is either dirty, clean, or blocked (e.g., furniture is located there). **You do not need to look at this file unless you want to.**
  - *Room.java*: Defines a full room, made up of many RoomTile objects. **You do not need to look at this file unless you want to.**
  - *Roomba.java*: Defines the overall behavior of a Roomba. You may look at this class but you cannot change any of the code.
  - *RoombaGui.java*: Code for the visual interface of the project. **You do not need to look at this file unless you want to.**
  - *RoombaSimulator.java*: Overall simulator that handles the number of cycles before Roomba must be done, handles Roomba's movement, etc. **You do not need to look at this file unless you want to.**

Note that you should only be writing the **makeMove()** method in **MyRoomba.java**. You can safely ignore everything else.

1.2 Implement Making Moves

The move making method has the following signature:

```
public Move makeMove();
```

Your only task is to write this method. We've provided a VERY simple implementation to get you started. Your method must return an instance of Move, specifically the move you'd like your Roomba to make this round. you can return **Move.FORWARD, Move.TURNCLOCKWISE, OR Move.TURNCOUNTERCLOCKWISE**

```
public Move makeMove(){
    return Move.FORWARD; //this robot will always move forward
}
```

Notice that your Roomba **cannot see the room directly**. This means, you need some other means of detecting the world around you. The actual Roomba uses several sensors to look at the environment. The simulator you are given provides the following:

- *this.frontBumper*: This boolean is set to true if the front of your Roomba has come in contact with a blocked room tile OR the edge of the room.
- *this.infraredSensor*: This integer provides the distance of the closest object. More specifically, the closest object is within this number of steps away from Roomba's center. *Note that for this calculation, a diagonal move is considered the same distance as a cardinal move.*
- *this.wallSensor*: This sensor (boolean) is set to true if and only if there is an object or wall directly to the right of Roomba's current position. This is useful for tracing along wall and around objects if desired.

You should use the variables listed above in order to make decisions about what your Roomba should do next. Roomba cannot look at the room / furniture layout directly. For example, you can use this.frontBumper to see when you've hit a wall and return this.TURNCLOCKWISE to turn the roomba).

1.3 Thinking in terms of "Frames"

This assignment works very similarly to how programming a game might work. You need to imagine / remember that your *makeMove()* method is being called repeatedly (about 60 times per second). This means your Roomba is constantly making decisions about what to do for the next 16 milliseconds or so only. Then the makeMove() method is invoked AGAIN to make a decision about what to do for the next 16 milliseconds, etc.

It might help to imagine that your makeMove() method is inside of a while loop (it is!), and write the method so that it only make a small decision right now. Remember that your method will run again and reevaluate the decision very soon so Roomba isn't committed to this action for very long.

1.4 Requirements

For submitting this homework, your makeMove() method must adhere to the following requirements:

- you MUST have at least three if statements in your method.
- your Roomba MUST use some amount of randomness in decision making (i.e., your code must invoke Math.random() and use the result in its decision)
- you MUST use each of the sensor variables at least once in your code. Each variable must actually have an effect on your robot's decisions (you can't just print out the value of a sensor).
- your Roomba should be cleaning more of the room than the simple Roomba we provided with the starter code (most of the time).

If your code has the items above, you will get a 10/10. *Do NOT overthink this. This is an open-ended assignment and this was done on purpose. Write some code, play around with it / tweak it a bit, and then turn it in!*. This is not meant to be a difficult assignment.