# Basic Java 2 - Roomba Simulator

### Dr. Mark R. Floryan

August 30, 2019

## 1 SUMMARY

A while ago, I purchased a Roomba, a small round robot that scurries along my floor and vacuums. The robot is quite cute, and I always wondered precisely how the algorithm for determining its actions worked. 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.

There are several Classes that already exist in the project, that we 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.

  - *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).

  - *Room.java*: Defines a full room, made up of many RoomTile objects.

  - *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.

  - *RoombaSimulator.java*: Overall simulator that handles the number of cycles before Roomba must be done, handles Roomba's movement, etc.

Note that you should only be writing the **makeMove()** method in **MyRoomba.java**.

## 1.2 IMPLEMENT MAKING MOVES

The move making method has the following signature:

```
1  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.

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 iff 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.

## 1.3 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.

- your Roomba must ALWAYS clean more of the room than the simple Roomba we provided with the starter code.