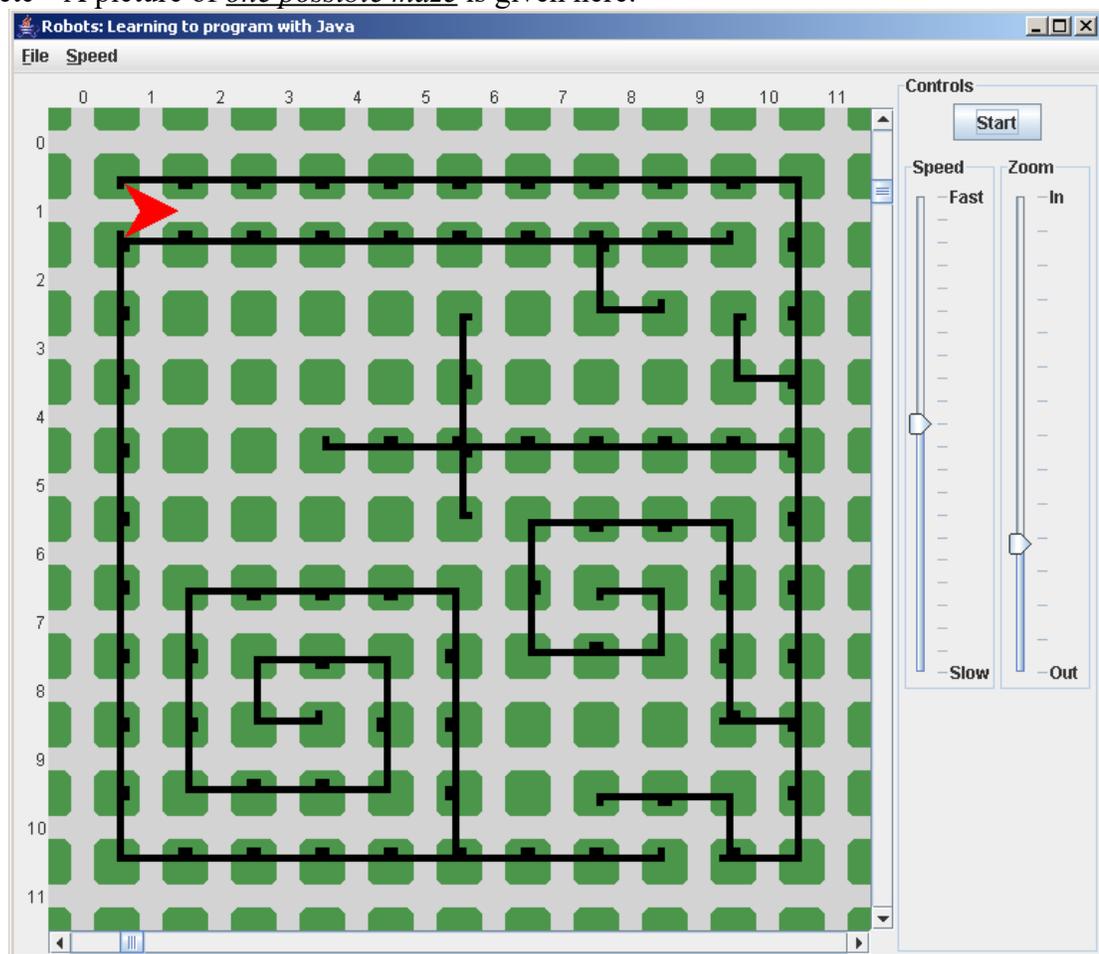


BIT 115, ASSIGNMENT: MAZE NAVIGATION

Navigating A Robot Through A Maze

Download the file `Maze.java` from website. You should add code to the `NavigateMaze` command to make the robot go from the starting point to the end point. The only things which you may assume about the maze are the following: with the exception of the entry point and exit point, the outer wall of the maze will be unbroken (so you don't have to worry about the robot wandering out of the maze by going through the 'wrong door out'). You may also assume that the maze will be configured so that the `MazeBot` will start in between 2 walls, and will be pointing into the maze. I.e., there will be a wall on it's left, and a wall on it's right. You're also guaranteed that there actually is a way to get through the maze to the exit point. You're not allowed to leave the maze (by turning 180 degrees around, and leaving via the entry point), walk around the maze to the exit point, and pronounce yourself done. **You must write a program so that the robot will guide itself through the maze; you cannot depend on the user (say) to 'steer' the robot for you.**

Thus, the `Maze.java` file gives a single maze that your robot should be able to navigate through, but this isn't the only maze – any maze that satisfies the limitations set forth in the prior paragraph should be something that your robot can handle. As such, you can't just give it a set of directions like "move forwards 9 times, turn right once, move twice, etc" A picture of *one possible maze* is given here:



Additionally, each MazeBot should also start with exactly 1,000 Things in its backpack. Each time the MazeBot moves forwards, it should leave a Thing in the current intersection, but only if there isn't a Thing there already. Obviously, the robot shouldn't break if it runs out of Things (regardless of whether there's something in the intersection or not)

The MazeBot also needs to be able to return the total number of spaces it has moved, via the command (method) named `printTotalNumberOfSpacesMoved`. This will print the total number of spaces that the MazeBot has moved since the `NavigateMaze` method was called. In other words, if someone calls `NavigateMaze` on the MazeBot, and it takes 15 moves to move through the maze, then `printTotalNumberOfSpacesMoved` should return 15. An outline of this method is provided in the `Maze.java` file.

Likewise, you will need to add methods that will tell someone how many spaces the robot has moved while facing east (name this `printTotalNumberOfSpacesMovedEast`), another for the number of spaces moved while facing west (name this `printTotalNumberOfSpacesMovedWest`), another for the number of spaces moved while facing north, and one last for the number of spaces moved while facing south. Much like the plain-vanilla `printTotalNumberOfSpacesMoved`, these methods only record the total number of spaces moved (in their respective directions) since the last time the MazeBot was told to `NavigateMaze`.

You should create a command, named something like `printAllDistances`, which will use the above method to print out all of that above information, to the human user. Make sure that whenever `NavigateMaze` finishes navigating its way through a maze, this new command is called.

Obviously, the MazeBot should never cause itself to break.

You should put your answer to this part of the homework assignment in a file named `Maze.java`.

Documentation:

Make sure that every file contains the names of all group members, as well as the class name ("BIT 115: Introduction To Programming"), year and quarter ("2006 Winter"), and section number ("Section 1"). For your Java files, put these in comments at the top of the file. Your Java code should also contain comments explaining new methods and decision points.

The group should also produce a short, 2 page (maximum!) document describing how the program works. I want to see concise, yet insightful, and helpful explanations of each of the algorithms that make up the program (in particular, you should include a description of the algorithm for finding a way through the maze, for dropping Things, for tracking the total number of spaces moved). Additionally, you'll need to provide a quick overview of how each of these algorithms interact to solve the overall problem.

Group Work:

You are allowed to work with one **or two** other person(s) on this homework assignment, but you are not required to do so. **In other words, groups may be composed of three people, or two people, or just one individual.** Working together with other people should help you overcome conceptually minor technical difficulties, and will allow you to discuss strategies and tactics about how to solve the problem. **If you work with someone(s) else, then you are required to hand in just one assignment for the entire group (It doesn't matter who hands in the assignment, as long as everyone's name is on it).** An identical grade will be given to all people, although the instructor reserves the right to modify that grade if subsequent information indicates that one person didn't contribute equally. In particular, the instructor reserves the right to assess the extent of learning via such methods as verbal questioning, quizzes (announced or otherwise), etc. The purpose of having you work in groups is to ensure that you each develop a thorough understanding of how to program; it is NOT to make one person do all the work.

Once you've found a group to work in, inform the instructor. If you decide to change groups after this point, you must first inform the instructor.

You are required to provide your partners with at least one means of contact – an email account that you check regularly would be fine. Heck, you could even create a brand new account just for this class, or even just for this partner. Likewise, you do need to be responsive, and work towards the completion of the assignment.

What to turn in:

- The java file, *maze.java*
- The documentation file, in MS Word, Open-Office doc, or PDF format.
- Make sure you have each contain the names of all group members.