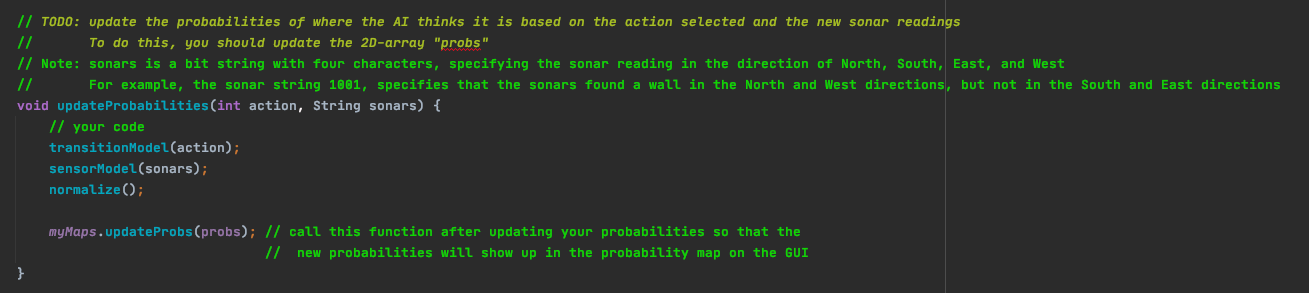
­­updateProbabilities:



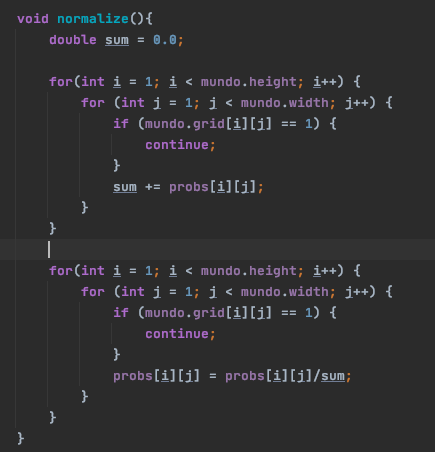
Transition Model:



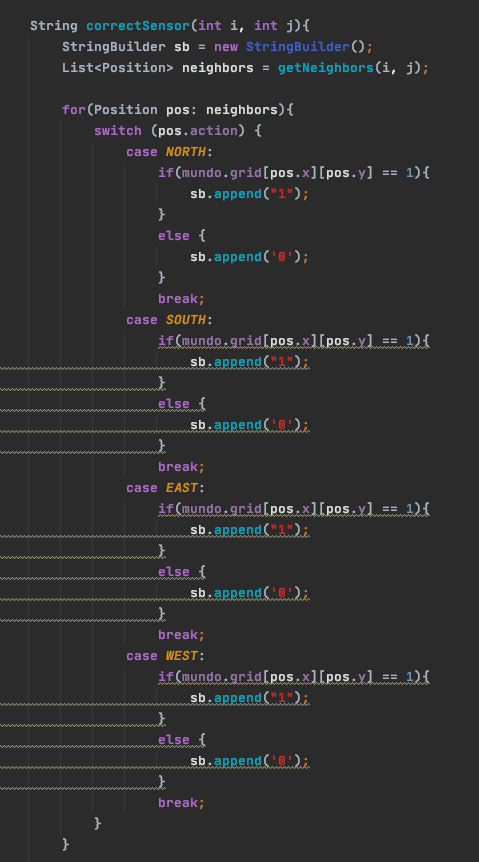
Sensor Model:

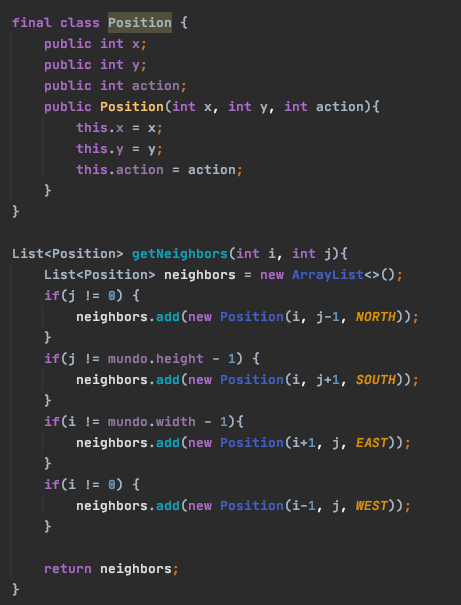


normalize:



correctSensor:



getNeighbors:

**Advice to a younger me:**

While the assignment was pretty clear, the thing that will stump you are getting the neighbors of each cell. Because your transition model relies on the squares around it, don’t try and update each cell when you reach it, but rather update it’s neighboring cells. This way you won’t get confused on directions. Walls also need to be taken into account, and if you are not careful and diagram it out on paper before coding it, you will refactor the motion model over and over again like I did. One other thing that screwed me up, was make sure your neighbors are inserted into the Positions array in the right order, or you will think that you are getting correct sensor readings when really you are not. How I started to debug this was first I just used the transition model with a 1.0 probability, and made sure those results made sense on the screen. Then I used the sensor model with a 1.0 probability. This is a good way to start testing, as you don’t have to account for incorrect readings. Also make sure that when you start, know that the sensor model just needs to update the values you get from the transition model, and that you don’t need to multiply it again back into the probability matrix.

I would say the biggest pitfalls you will run into are not handling neighbors (checking for walls and behavior if you are running into a wall, etc.) and assuming you get correct sensor readings from your function. Just stay on top of those and this assignment will be much easier.