# CS 273 Laboratory 5: Iteration

This laboratory gives you practice with iteration in Java by implementing the behavior of a robot butler and drawing concentric boxes and checkerboards.

## Preliminaries

When you unzip the starter code package, there should be three folders: boxes, checkerboard and robot. Each folder should contain several project files.

## Laboratory

### Part 1: Simple moving robot

Read the following description of robot-butler operations:

The robot butler moves along tiles in a room, under program control. It is always facing either north, south, east or west. The robot can move forward by calling the forward method like this:

robbie.forward();

To turn the robot left or right, use the methods turnRight and turnLeft:

robbie.turnRight();

robbie.turnLeft();

If the robot is instructed to move forward when there is something that is blocking it, it will crash and quit working. (Click the Replace button to replace the broken robot with a new one.) The isFrontBlocked method tells whether there is an object (e.g., a wall or a rock) that will block the robot from moving forward. It returns a boolean value and may therefore be used as the condition for an if or while statement. For example:

if (!robbie.isFrontBlocked()) {

...

}

Note, the above code is the same as the following code, but the above is more sophisticated programming style and what I would recommend you practice using.

if (robbie.isFrontBlocked() == false) {

...

}

Get your robot moving around:

Open the project in the robot folder. (The project has already been built.)

For this project you will be editing and compiling the RobotAction file. However, to run the program you will execute the main method in the RoomFrame class by right-clicking the RoomFrame class, then selecting void main(String[]).

Run the program. When it is running, click the Go button. The robot should move two squares forward, then stop. If you hit the Go button again it moves again. You can also use the Backward, Left and Right buttons.

Modify the robot's behavior:

The Go button calls the runRobot() method in the RobotAction class. Modify the RobotAction class so that the robot keeps going until it reaches an object or a wall. Don't forget you need to use the Go button to start the robot moving.

Test your code by starting the robot from different positions using the buttons at the top of the screen.

**checkpoint 1 (15 points): Run your program for your lab instructor or assistant.**

### Part 2: Move the robot forward until an object (or wall) is reached; then turn around and repeat.

Modify your runRobot() method so that it does as before, but instead of stopping when it reaches an object or wall, it turns around and goes forward until it reaches another object (or wall). Then it stops and stays stopped. (Do not keep going back and forth between walls forever.)

Test your code by starting the robot from different positions using the buttons at the top of the screen.

**checkpoint 2 (15 points): Run your program for your lab instructor or assistant.**

### Part 3: Have your robot take up to 5 steps, and do the “dizzy-robot dance”

Comment out your previous robot code. Have your robot move forward up to 5 spaces, fewer if it reaches a wall or object first.

Then use one or more loops to make 2 clockwise circles (720 degrees total), immediately followed by 2 counterclockwise circles.

**checkpoint 3 (10 points): Run your program for your lab instructor or assistant.**

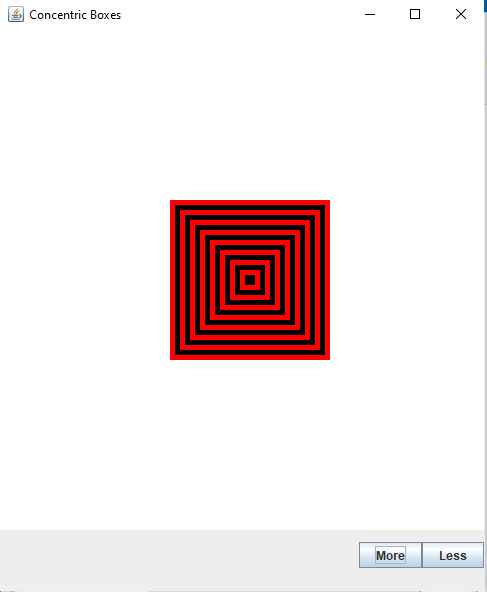
### Part 4: Draw N concentric boxes

Open the project in the boxes folder. It presently draws a black box whose size is 10x10 pixels, and whose center is at (250,250).

Modify the drawBoxes method so that it draws a series of boxes, all centered at (250,250). The colors of these boxes should alternate red and black. The inner box should be 10x10; each larger box should be 10 pixels larger in both height and width than the one that it immediately encloses. The number of boxes to draw is contained in the variable numBoxes. This variable is already set by another part of the program that you do not have to write. Specifically, the program increases numBoxes whenever the user clicks the More button and decreases numBoxes every time the user clicks Less button (but are bounded by 1 and 30).  
**Note: your code should not change numBoxes.**

**Hint:** Be sure to draw boxes in order from largest to smallest so the big boxes don't cover up the small ones!

This is what you’re going for. The smallest box in the center should be black and should remain black each time you hit the More or Less button.



**checkpoint 4 (30 points): Run your program for your lab instructor or assistant.**

If you have completed all above checkpoints, you now have a C- (70) for this lab.

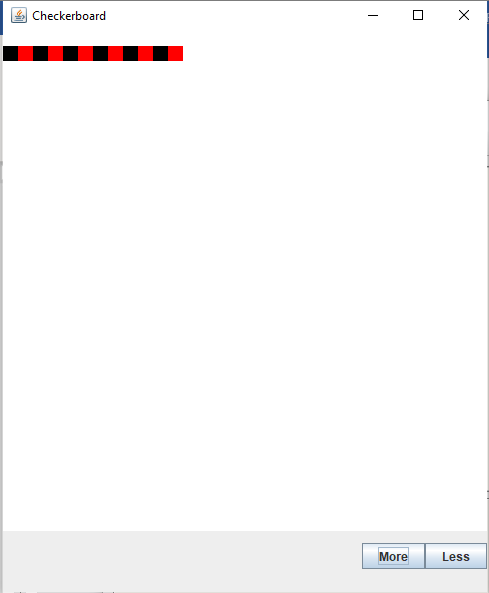
### Part 5: Draw an NxN checkerboard, part 1

Open the project in the checkerboard folder. It presently draws nothing at all.

Over the next two checkpoints, you will code the drawCheckerboard method so that it draws a square NxN checkerboard, where each square on the checkerboard is 15x15 pixels. The squares on the checkerboard should alternate red and black, with the top left square always being black. The length and width of the checkerboard you should draw is contained in the variable numSquares. As before, this variable is already set by another part of the program that you do not have to write. The program increases numSquares whenever the user clicks the More button and decreases numSquares every time the user clicks Less button (but is bounded by 1 and 30).  
**Note: your code should not change numSquares.**

Take this task in steps. First, write code to draw just the top row of numSquares squares, alternating red and black.

This is what you’re going for. The top left square should be black and should remain black as you hit the More and Less buttons. The top left square should start at (0,0).



**checkpoint 5 (15 points): Run your program for your lab instructor or assistant.**

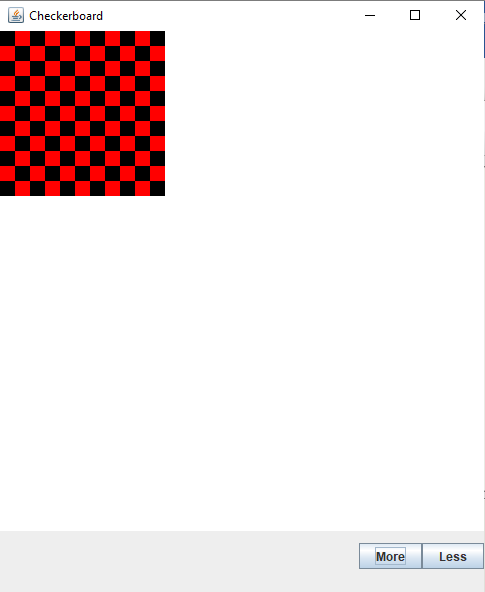
If you have completed all above checkpoints, you now have a B (85) for this lab.

### Part 6: Draw an NxN checkerboard, part 2

Next, modify your code so that you draw the row numSquares times. You will end up with stripes, not a checkerboard, but that’s okay. Hint: this will require nested loops.

Finally, modify your code so that the colors alternate correctly to make the checkerboard pattern. Hint: remember that you have two loop control variables to work with in determining which color to use in drawing each square.

This is what you’re going for. The top left square should be black and should remain black as you hit the More and Less buttons.



**checkpoint 6 (15 points): Run your program for your lab instructor or assistant.**

If you have completed all above checkpoints, you now have an A (100) for this lab.

### EXTRA CREDIT The last checkpoint below is extra credit.

### Option 1: Hide and Seek

Modify your robot once again to seek and find the obstacle in the middle of the room regardless of what position (and orientation) it starts from. The robot should stop when it is certain that it occupies a square next to the obstacle.

**checkpoint EC1 (5 points): Run your program for your lab instructor or assistant.**

### Option 2: Spiral Pattern

This time, your robot should:

* visit every grid cell in the whole room
* without visiting the same cell twice
* move outwards from its starting position and orientation in a spiral pattern.

You can still get some points for this if your spiral pattern doesn’t visit every grid cell or sometimes visits the same cell twice – see what you can do, and ask around for ideas!

**checkpoint EC2 (up to 5 points): Run your program for your lab instructor or assistant.**