# CPSC 1100 – LAB 04

Methods, Constructors, Tester Classes, and javaDocs

This lab will deal with implementing methods, constructors, and tester classes. Additionally, you will learn to comment your code with the javadocs format. The lab will be due next week. **PLEASE COMMENT YOUR CODE.** You will have points taken off if you do not comment your code. Keep your code neat.

**Some useful links:**

BlueJ tutorial [www.bluej.org/tutorial/tutorial-201.pdf](http://www.bluej.org/tutorial/tutorial-201.pdf)

Java tutorial home page: <http://docs.oracle.com/javase/tutorial/>

Start here: <http://docs.oracle.com/javase/tutorial/java/index.html>

variables <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>

data types <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

## Tasks: Follow the directions below to complete your lab assignment

Note in the steps listed below, you should stop and test EVERY method you create before you move on to the next one. Create tests in your PlayerPieceTester class and call methods from the PlayerPiece class. You should print both the expected results (calculated by hand), and the results calculated by the methods of the PlayerPiece class. For example, you can create a PlayerPiece object, named piece1, and then verify that piece1.getNumPotions() returns the value you expect. More details on the PlayerPieceTester class are given below in the instructions. Note that the more carefully you test your methods, the more likely you are to find bugs. When I grade this project I will insert my own tester class that will test the methods you create in the PlayerPiece class.

1. Create a new project. Create a new class called PlayerPiece. Replace all the code in the PlayerPiece.java file with PlayerPiece.java.txt as posted on Google Drive.
2. Create a new class called PlayerPieceTester. Replace all the code in PlayerPieceTester.java with the code in PlayerPieceTester.java as posted on Google Driver.
3. Right click on the PlayerPieceTester class, and choose the main method. When asked for arguments, just click ok. You should see a terminal window pop up that displays expected results, as well as actual results. Go and look at the main method in the PlayerPieceTester class and make sure that you understand how the results were calculated and printed. You will need to create tests for your methods below. (You should create a method, then create a test for it).
4. Fill in the no argument constructor for the PlayerPiece class. Initialize instance variables to the values indicated in the comments next to each instance variable in the starter code. So your first line of code in the no argument constructor will be pieceName = “no name”;
5. Create a test in the PlayerPieceTester class that verifies your values are set correctly in the constructor. (Hint: use the appropriate get() method for each instance variable).
6. Fill in the second constructor listed that takes in a long list of parameters. Initialize instance variables to the values of the arguments. The first one has been done for you. (pieceName).
7. Create a test in the PlayerPieceTester class that verifies your values are set correctly in this constructor. (Again, use the get() methods to access values of your PlayerPiece object).
8. Edit your mutator methods in the PlayerPiece class that move the piece by one location. moveUp(), moveDown(), moveLeft(), and moveRight()
9. moveUp() will need to add 1 to locY.
10. moveDown() will need to subtract 1 from locY.
11. moveLeft() will need to subtract 1 from locX.
12. moveRight() will need to add 1 to locX.
13. Create a test for each of these methods created in steps 9,10,11,12 (You will call the “move” method on one line of code, then print the results from the appropriate “get” method on the next line to test these methods. For example, you call moveUp(), you will need to then use getLocY() to ensure the piece has moved up).
14. Create a mutator method in the PlayerPiececlass called setLocXY(int x, int y). As you can see this method will take 2 int's as parameters. Inside the method set the instance variables locX and locY to the appropriate parameters of the method.
15. Create a test for setLocXY()
16. Create a method inside of the PlayerPiece class called deductHealth(int amt). Return type void. This method will deduct amt from the instance variable currentHealth.
17. Create a test for deductHealth()
18. Create a method inside of the PlayerPiece class called addHealth(int amt). Return type void. This method will add amt to the instance variable currentHealth.
19. Create a test for addHealth()
20. Once you have created all methods, run your tests, and take a screen-shot of your output to include with your project. Text output is OK, does not have to be a screen-shot.
21. Create javadoc comments for the PlayerPiece class. You should comment the class itself, every method, every parameter, and every return value in this class. You can select Tools → Project Documentation or just press <ctrl> j to view your current documentation. The GameBoard class has already been commented for you. If a box pops up asking to view or regenerate, make sure you click “regenerate”. This will recreate the javadocs based on the javadoc comments that you place within your code. (You can access a commented version of GameBoard.java.txt under the Lab04 heading. We do not need this class to complete this lab however, so it is not included in your BlueJ project).

## To Turn In via Google Driver

You should turn in all of your java files and a text document with your output.