

CMIS 141 Hands-on Lab

Week 4

Overview

The week we continue our study of Java by taking a closer look at classes and objects. The topics covered in this lab include creating classes, creating objects and creating methods and controlling access to class members.

It is assumed the JDK 8 or higher programming environment is properly installed and the associated readings for this week have been completed.

Submission requirements

Hands-on labs are designed for you to complete each week as a form of self-assessment. You do not need to submit your lab work for grading. However; you should post questions in the weekly questions area if you have any trouble or questions related to completing the lab. These labs will help prepare you for the graded assignments, therefore; it is highly recommended you successfully complete these exercises.

Objectives

The following objectives will be covered by successfully completing each exercise:

1. Create Java classes and construct objects from the classes
2. Assign member modifiers to control access within Java classes

Exercise 1 – Create Java classes and construct objects from the classes

Java classes can be thought of a blueprints that we can use to construct objects. We have already used constructors when we created an instance of the Scanner, String and other classes.

In this exercise we create a class representing a Point. The Point class will have a constructor, fields and methods. For example, a Point (in 2-dimensions) will have an x and y value. We won't be plotting points yet. We will just be creating a Point class and calculating values that might be used in a Point class with methods.

- a. Open your favorite text editor and Type (or copy and paste) the following Java

```
/*
 * File: TestPoint.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program constructs instances
 * of the Point class and uses
 * its methods
 */
```

```

public class Point {
    private double x;
    private double y;

    // Constructor
    public Point (double px, double py) {
        x = px;
        y = py;
    }

    // Default constructor
    public Point () {
        x = 0.0;
        y = 0.0;
    }

    // Setter methods
    // setX
    public void setX(double px) {
        x = px;
    }
    // setY()
    public void setY(double py) {
        y = py;
    }

    // Getter methods
    // getX
    public double getX() {
        return x;
    }
    // getY
    public double getY() {
        return y;
    }

    // Use Math method to get the distance
    // between 2 points
    public double getDistance(Point p1, Point p2) {
        // delta x
        double deltaX = Math.abs(p1.getX() - p2.getX());
        // delta y
        double deltaY = Math.abs(p1.getY() - p2.getY());

        // Distance formula
        double distance = Math.sqrt(deltaX*deltaX + deltaY*deltaY);
        return distance;
    }

    // toString method
    public String toString() {
        String str = "(" + x + ", " + y + ")";
        return str;
    }
}

```

```
}
```

- b. Save the file as “Point.java” in a location of your choice.
- c. To compile the file, type `javac Point.java` at the command prompt.
- d. To test the Point class, you will need an additional class. These classes are often called Driver or Test classes and are used to construct instances of the class and call the methods. You create a class containing a main method. In the main method you construct an existing class and then call the methods. To create a test class for the Point class, open your favorite text editor and copy and paste the following code:

```
/*
 * File: TestPoint.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program constructs instances
 * of the Point class and uses
 * its methods
 */

public class TestPoint {
    public static void main(String[] args) {

        // Construct a point with 1.0, 1.0
        Point p1 = new Point(1.0,1.0);

        // Construct a default point
        Point p2 = new Point();

        // Call the getter methods
        double p1X = p1.getX();
        double p1Y = p1.getY();
        // Print results
        System.out.println("p1 values from getX() getY() " +
                           p1X + "," + p1Y);

        double p2X = p2.getX();
        double p2Y = p2.getY();
        // Print results
        System.out.println("p2 values from getX() getY() " +
                           p2X + "," + p2Y);

        // Call the Distance Method
        double distance = p1.getDistance(p1,p2);
        // Print results
        System.out.println("Distance between p1 and p2 is: " +
                           distance);
        // Change the XY value of P1
        // Using the setter method
        double newX = 2.0;
        double newY = 2.0;
```

```

    p1.setX(newX);
    p1.setY(newY);

    // Recalculate the Distance
    distance = p1.getDistance(p1,p2);
    // Print results
    System.out.println("New Distance between p1 and p2 is: " +
                        distance);

    // Display the values using toString
    System.out.println(p1.toString());
    System.out.println(p2.toString());

}
}

```

- e. Save the file a TestPoint.java in the same directory as the Point.java file. Compile and then execute java TestPoint. Note, that you have previously compiled Point.java. If you hadn't compiled Point.java, the TestPoint.java file would not find the class. The output will look similar to this:

```

Command Prompt

C:\Users\jim\Documents\UMUC\Courses\CHIS141\Spring2015\Code\Week4>java TestPoint
p1 values from getX() getY() 1.0,1.0
p2 values from getX() getY() 0.0,0.0
Distance between p1 and p2 is: 1.4142135623730951
New Distance between p1 and p2 is: 2.8284271247461903
(2.0, 2.0)
(0.0, 0.0)

C:\Users\jim\Documents\UMUC\Courses\CHIS141\Spring2015\Code\Week4>

```

As you analyze and experiment with the code, note the following:

1. The Point.java file contains 2 different constructors. The default constructor will assign x and y to 0.0. Classes can have multiple constructors. Designing constructors are often based on the data available or the specific application. You use constructors to create an instance (or object) of a class.

2. The x,y fields are Private. This means the values are encapsulated requiring the call to getX() or getY() to reveal the value. If you try to call p1.x or p1.y you will receive compile error. For example, if I added this code to the TestPoint.java:

```
System.out.println("p1.x is " + p1.x);
```

And then tried to recompile, an error message similar to the following would be received:

```
TestPoint.java:51: error: x has private access in Point
    System.out.println("p1.x is " + p1.x);
```

Defining fields as private, is a good security practice demonstrates good object-oriented design. In most cases, you should declare fields as private in the classes you design.

3. Because the fields are private, we define getter methods called getX() and getY(). These methods allow for access to values of x and y as demonstrated in the following code:

```
double p2X = p2.getX();
double p2Y = p2.getY();
```

4. Setter methods are also used allowing the ability of the user to change the x, and y values of the Point. The following code demonstrates their use:

```
double newX = 2.0;
double newY = 2.0;
p1.setX(newX);
p1.setY(newY);
```

5. Notice, we took advantage of the static Math.abs() and Math.sqrt() methods in the Math class in the getDistance() method of the Point class. In most cases, You should use existing classes as opposed to reinventing the wheel.
6. Also, notice the toString() method returns the current x,y values of the point when called.

Now it is your turn. Try the following exercise:

Create a Java class named Point3D using your favorite text editor. Be sure you name the file "Point3D.java". Add code to the file in the main() method that will construct a 3D Point (x, y and z). You should have a default constructor located at (1.0,-1.0, -1.0) and another constructor that uses three double values for x, y and z. Provide a getDistance() method that returns the distance between two 3D points. Also, provide getter and setter methods for x, y and z. Create a toString() method that prints the values of x, y and z in the format (x,y,z). Finally create a TestPoint3D.java file to construct several points and demonstrate the use of all of the methods.

Exercise 2 - Assign member modifiers to control access within Java classes

Member modifiers are used to denote the access level for fields, methods and classes. Modifiers include public, private, protected and no modifier. Understanding the differences will take some practice and delving deeper into packages and sub-classes. Since we just have introduced objects, we will focus on private versus public for fields and methods at this time. Private methods and fields are only accessible by the class itself without encapsulation (e.g. getX() to return the x component in the Point class). Public methods and fields are accessible by all other classes.

To demonstrate this we will use the Point class from the previous class and modify the members to show the differences. We will also use the TestPoint class to demonstrate the impact of changing these modifiers.

- a. Open up the Point.java file and copy it to PointAccess.java. Change the Y variable to public access and change the getDistance() method to private. When you change the name of the class, several other changes are required so the code will compile. The code changes are highlighted in yellow below.

```
/*
 * File: PointAccess.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program constructs instances
 * of the Point class and uses
 * its methods
 */

public class PointAccess {
    private double x;
    public double y;

    // Constructor
    public PointAccess (double px, double py) {
        x = px;
        y = py;
    }

    // Default constructor
    public PointAccess () {
        x = 0.0;
        y = 0.0;
    }

    // Setter methods
    // setX
    public void setX(double px) {
        x = px;
    }
    // setY()
    public void setY(double py) {
        y = py;
    }
}
```

```

// Getter methods
// getX
public double getX() {
    return x;
}
// getY
public double getY() {
    return y;
}

// Use Math method to get the distance
// between 2 points
private double getDistance(PointAccess p1, PointAccess p2) {
    // delta x
    double deltaX = Math.abs(p1.getX() - p2.getX());
    // delta y
    double deltaY = Math.abs(p1.getY() - p2.getY());

    // Distance formula
    double distance = Math.sqrt(deltaX*deltaX + deltaY*deltaY);
    return distance;
}

// toString method
public String toString() {
    String str = "(" + x + ", " + y + ")";
    return str;
}
}

```

- b. Modify the TestPoint.java and copy it to TestPointAccess.java. The code changes required are shown below in yellow:

```

/*
* File: TestPointAccess.java
* Author: Dr. Robertson
* Date: January 1, 2015
* Purpose: This program constructs instances
* of the Point class and uses
* its methods
*/

public class TestPointAccess {
    public static void main(String[] args) {

        // Construct a point with 1.0, 1.0
        PointAccess p1 = new PointAccess(1.0,1.0);

        // Construct a default point
        PointAccess p2 = new PointAccess();
    }
}

```

```

// Call the getter methods
double p1X = p1.getX();
double p1Y = p1.getY();
// Print results
System.out.println("p1 values from getX() getY() " +
    p1X + "," + p1Y);

double p2X = p2.getX();
double p2Y = p2.getY();
// Print results
System.out.println("p2 values from getX() getY() " +
    p2X + "," + p2Y);

// Call the Distance Method
double distance = p1.getDistance(p1,p2);
// Print results
System.out.println("Distance between p1 and p2 is: " +
    distance);
// Change the XY value of P1
// Using the setter method
double newX = 2.0;
double newY = 2.0;
p1.setX(newX);
p1.setY(newY);

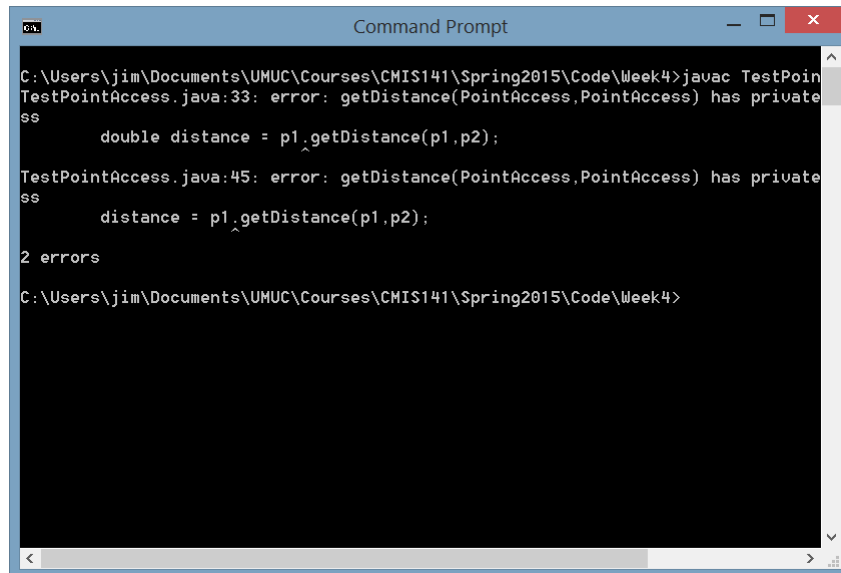
// Recalculate the Distance
distance = p1.getDistance(p1,p2);
// Print results
System.out.println("New Distance between p1 and p2 is: " +
    distance);
// Display the values using toString
System.out.println(p1.toString());
System.out.println(p2.toString());

}

}

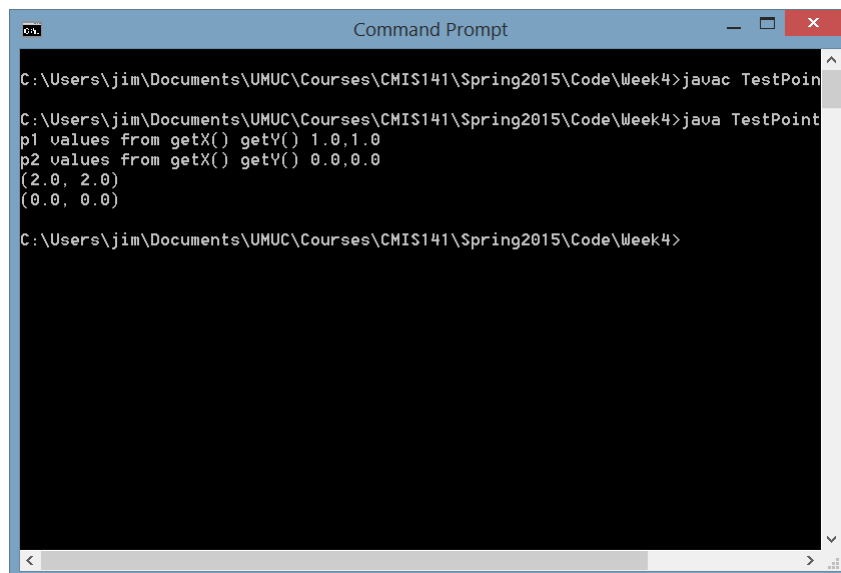
```

- c. Attempting to run this code, will result in an error as we are trying to call a private method of the PointAccess class from the TestPointAccess class.



```
Command Prompt
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>javac TestPoint
TestPointAccess.java:33: error: getDistance(PointAccess,PointAccess) has private
ss
    double distance = p1.getDistance(p1,p2);
TestPointAccess.java:45: error: getDistance(PointAccess,PointAccess) has private
ss
    distance = p1.getDistance(p1,p2);
2 errors
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>
```

- d. Comment out those lines of code and recompile and run.



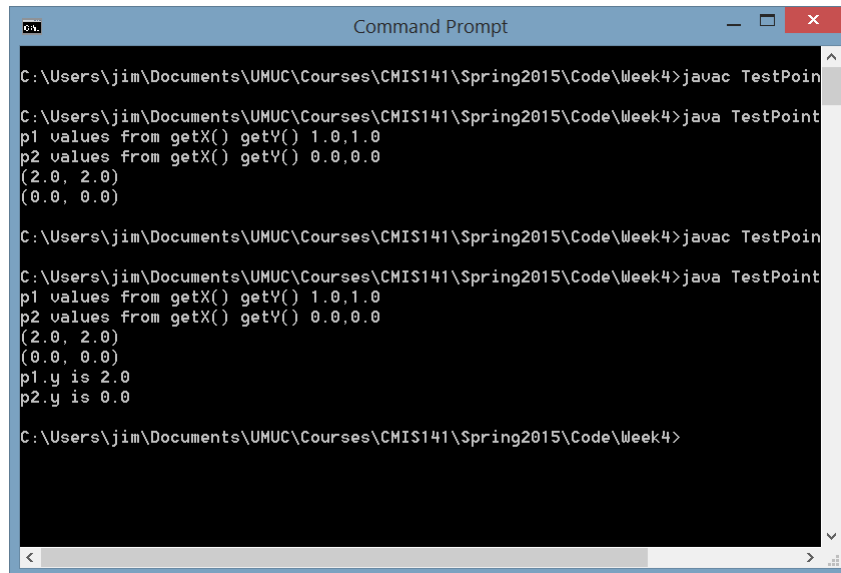
```
Command Prompt
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>javac TestPoint
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>java TestPoint
p1 values from getX() getY() 1.0,1.0
p2 values from getX() getY() 0.0,0.0
(2.0, 2.0)
(0.0, 0.0)
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>
```

With the lines related to calling the private `getDistance()` method and associated print methods are commented out, the code compiles and runs. If you envision, methods will be used by other classes, you should not declare them private as they cannot be used by the other classes.

- e. By changing the `y` field to public, we can now use `p1.y` to access the results directly. As mentioned in the reading this is not recommended. To demonstrate this functionality add the following code to the bottom of the `TestPointAccess.java` code:

```
// Display the y values directly
System.out.println("p1.y is " + p1.y);
System.out.println("p2.y is " + p2.y);
```

After compiling and executing the the p1.y and p2.y field variables are directly accessible.



```
Command Prompt
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>javac TestPoint
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>java TestPoint
p1 values from getX() getY() 1.0,1.0
p2 values from getX() getY() 0.0,0.0
(2.0, 2.0)
(0.0, 0.0)
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>javac TestPoint
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>java TestPoint
p1 values from getX() getY() 1.0,1.0
p2 values from getX() getY() 0.0,0.0
(2.0, 2.0)
(0.0, 0.0)
p1.y is 2.0
p2.y is 0.0
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week4>
```

Now it is your turn. Try the following exercise:

Modify the Java class named Point3D to change the access to make the setter methods private. Try to call the setX(), setY() and setZ() methods from the TestPoint3D class and note the errors. Change the x, y and z fields to public access. Then attempt to print the p1.x, p1.y, p1.z, p2.x, p2.y and p2.z values directly as opposed to using the getter methods. You will need to comment out some code, to get past the compile errors related to the private methods.