Lab 2: Designing Classes

Part A: Getting Started

Download **Lab2Point.exe,** extract it, open the project.

In this part you will add test cases to an existing class and its test suite.
The handout has the code for the `Class ManhattanPoint` and its test suite `ManhattanPointTest`.

1. Define `aManhattanPoint` with coordinates `x = 29` and `y = 17.`
2. Add test for the method `distanceTo0()`
3. Add test for the method `isWithin(...)`
4. Add test for the method `isCloserThan(...)`
5. Add test for the method `getCloserPoint(...)`
6. Save and print the results of your tests, and the source code for your extended test suite.

Part B: Designing-Building a Class
Download **Lab2Book.exe,** extract it, open the project.

1. Define the class `Book` which contains as member data the name of the author, the title of the book, the book id, the year published.
2. Build the skeleton of the test suite for this class.
3. Define the constructor for this class.
4. Define the `toString` method for this class and test the constructor.
5. Develop the method `howOld()` which determines how old is the book.

Part C: Class Composition

1. Design-Build the class `LibRecord` which contains lending record for a book in the library. The record contains a `boolean` value `available`, the `day` it is due (make it just an integer), and the reference to a `Book` object. Add it to your project.
2. Develop a method `takeOut`, which records that the book has been taken out for three weeks.
3. Develop the method `returnBook`, which records the return of the book.
4. Draw UML diagram of this collection of classes.

Appendix A: Rules for writing the `toString` *method for a class (thanks aLJaFP):*

- if the class doe not contain any member data fields, use
```
public String toString(){
       return "new" + getClass().getName() + "()"; }
```
- if the class has one member data field, say x, use
```
public String toString(){
       return "new" + getClass().getName() + "(" + x + ")"; }
```
- if the class has two member data fields, say x and y, use
```
public String toString(){
       return "new" + getClass().getName() + "(" + x + ", " +  y + ")"; }
```

To define an object to experiment with:

```
SomeClass y = new SomeClass(.....);
println(y);

expected(...);
actual(y.methodName());
```

Appendix B: Building Your Own Project to Design a Class `MyClass`

In **File** menu select **New**

1. Select **Java2SE Stationery**
2. Type in the name for your project and hit return
3. Expand the **Generic** item and select **Java Application**
4. When the project opens, right-click next to its name and select MDI child
   This gives you more space to work with - the project floats in the window.
5. In the **Project** menu select **Create Group** and name it **CodeBase**
6. In **Project** menu select **Add Files** and select the file `jpt.jar` and `jpfalt.jar`
7. The window asks where to add files - select **Java Application Release** only
8. Expand the **Sources** tab in the project file list.
9. Delete `TrivialApplication`.
10. In **File** menu select **New**
11. Select **File** tab, then **Text File** - make sure you see where is it stored.
12. Give the file the name of the class you will write: **MyClass.java** for `Class MyClass` .
13. Now you can develop your class.
14. Copy the skeleton of the test class (e.g. **BookTest.java**) and re-name it **MyClassTest.java**.
15. In three places change the `BookTest` to `MyClassTest`.
16. In **Edit** menu select **Java Application Release Settings** near the bottom
17. On the left select **Java Target** and replace `TrivialApplication` with the name of your test class `MyClassTest`.
18. On the left select **Java Output** and replace `TrivialApplication` with the name of your test class `MyClassTest`.
19. Make sure the output type is again **Java Application**
20. Develop the test suite and run your program