Lab 4 : INHERITANCE & POLYMORPHISM

1. OBJECTIVE

The objective of Lab 4 is to practise on class inheritance and polymorphism.

Notes:
It is intended to be a short lab so that teams can get together to discuss/work on the assignment project which would have been published.

2. INTRODUCTION

Inheritance and Polymorphism are 2 important concepts in Object-Oriented Design and Programming. In this lab, you will get to learn more about the concepts in action. In one the tasks, you will need to decide whether ***concrete class, abstract class or interface*** is appropriate for the task required. In addition, you will also need to decide on the appropriate ***'is a'*** **or** ***'has a'*** relationship (inheritance/generalization VS delegation/object composition).

3. *Your Tasks for this LAB*

3.1 Polymorphic Sorting

Hints :
This is lab does not need a lot of coding, it is just a matter of where needs to be changed or added additional codes.

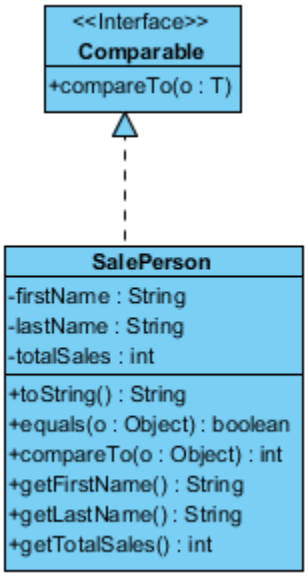
The file *Sorting.java* contains the Sorting class (in attachment). This class implements both the selection sort and the insertion sort algorithms for sorting any array of *Comparable* objects in ascending order. In this exercise, you will use the Sorting class to sort several different types of objects. (For details on the *Compareble* interface, refer to Java API doc http://docs.oracle.com/javase/7/docs/api/)

1. The file Numbers.java (in attachment) reads in an array of integers, invokes the selection sort algorithm to sort them, and then prints the sorted array. Save Sorting.java and Numbers.java to your directory. Numbers.java won't compile in its current form. Study it to see if you can figure out why.

2. Try to compile Numbers.java and see what the error message is. The problem involves the difference between primitive data and objects. Change the program so it will work correctly (note: you don't need to make many changes - the *autoboxing* feature of Java 1.5 (or higher) will take care of most conversions from int to Integer). You are to do research in the internet and understand better *autoboxing*.

3. Write a program Strings.java, similar to Numbers.java, that reads in an array of String objects and sorts them. You may just copy and edit Numbers.java.

4. Modify the insertionSort algorithm so that it sorts in descending order rather than ascending order. Change Numbers.java and Strings.java to call insertionSort rather than selectionSort. Run both to make sure the sorting is correct.



5. The class diagram on the right defines the SalePerson class that represents a sale person. The sale person has a first name, last name, and a total number of sales (an int).
   - The *toString* method will return the name of the sale person and total sales in the formal :
     <lastName> , <firstName> : <totalSales>
   - The *equals* method will check whether the first and last names of Object are the same as the current sale person.
   - The *compareTo* method make the comparison based on total sales; that is, return a negative number if the executing object has total sales less than the other object and return a positive number if the sales are greater. ***Use the name of the sales person's last name to break a tie (in ascending alphabetical order).***
   - ***Create and Write the SalePerson class***

6. The file WeeklySales.java (in attachment) contains a driver for testing the compareTo method and the sorting . Compile and run it. Make sure your compareTo method is correct. The sales staff should be listed in **the order of sales from most to least**. If the sale staffs have the same number of sales, they are listed in ascending alphabetical order of their last names.

3.2 Calculate Surface Area of a Figure.

By using the concepts of inheritance and **polymorphism**, you are required to design a program that calculate the total surface area of a figure. The following are the requirements an constraints :

- You should have a Class/Interface called *Shape and decide its appropriate attributes and behaviours*
- You should have basic shapes like *Square*, *Rectangle*, *Circle* and *Triangle*.
- The program will request the user to :
  - enter the total number of shapes
  - choose the shape and enter the required dimension/s for the selected shape
  - choose the type of calculation (for now, we will just calculate *Area*, with future plan to calculate *Volume* as well).
- The calculation/s should be done upon user's request and *NOT when dimensions are entered*.

1. For a start, use the 2-D figure on the right to verify your program. The figure consists of a Circle (radius=10), a Triangle (height=25, base =20) and a Rectangle (length=50, breadth = 20) . ***Calculate the total area of the 2- D figure***.
   (You will create an Application class **Shape2DApp.java** for this purpose)

2. We will now expand and extend your design to cater to 3-D figures. Imagine the figure on the right is turn into a 3-D figure – Circle becomes Sphere, Triangle becomes a square-based Pyramid and the Rectangle is a cubiod. ***Calculate the total surface area of the 3- D figure***. [Note : You need to think whether 'is a' or 'has a' relationship is more appropriate and relevant for between 2D and 3D shapes.
   (You will create an Application class **Shape3DApp.java** for this purpose)

3. We will include more Shapes. The square-based Pyramid will be replaced with a Cone and the Cubiod is replaced with a Cylinder. ***Calculate the total surface area of the new 3- D figure***.
   (You will **reuse** the Application class **Shape3DApp.java** with appropriate selection)

Continue on next page…..

IMPORTANT :

For this task, you are encouraged to work in pair. At the end of this task, you or the pair will demonstrate to your lab sup the working of your program.

Note :

The solutions for 3D Figures implementation can be many, simple or complex (like considering the overlapping regions). But the main objectives is to considered the idea and concept of polymorphism in the design and implementation. To build an application like AutoCAD, it can be just a discussion with your lab supervisor.