Homework-1

CSCI 5448-Object Oriented Analysis and Design

Submitted by: Dheeraj Chinni Ranga

1) **a) Abstraction**: Sets of concepts (Methods) provided in a form of a class or module by hiding the implementation details and show only the functionality for solving a problem or task is called Abstraction.

   Good Example: Suppose consider an abstract class of employee which will have the methods like Name, Designation, Worked hours, payment calculation for a permanent employee, payment calculation for a contract employee, and payment type (cash or check or direct deposit).

   Bad Example: Employee class with methods that are difficult to implement and are not necessary to be abstracted.

b) Encapsulation: This is the process by which one can hide the details of implementation of one class methods to the other classes or modules in that software system.

   Good Example: Consider a class which has the personal details of a person suppose Name, Age and SSN it is the best practice to make the variable private so that no one can manipulate them.

   Bad Example: Consider a class with the same personal details but the methods or variable are made public which is a bad practice.

c) Cohesion: Cohesion refers to class or methods which will focus on only one process.

   Good Example: Consider a payroll system where there are separate methods for calculating the number of hours and then calculating the amount for these number of hours.

   Bad Example: Suppose in the pay roll system if a single method does the work of calculating the number of hours and generating the pay checks which can be direct deposit or check or cash will be a bad example for cohesion.

d) Coupling: coupling represents the strength between the classes or methods or modules.

   Good Example: Consider a system where just very less information from other class or method to do its own job.

Bad Example: Consider a system where each class or method are highly depends on other class or method to get its job done.

2) **Functional Decomposition:** This is the process by which we can break a single problem in to number of small problems which can be easily solved. Once the problem has been divided in to number of sub problems they need to be arranged in to an order to achieve the goal.

   Assumptions: I am considering there is only permanent employees who are billed hourly or monthly and there are two types of payment methods checks and direct deposit.

   Design:
   - **a)** Connect to the database.
   - **b)** Retrieve the employee details including number of hours worked, pay rate and type of payment.
   - **c)** Sort the employee based on the type of payment method (Hourly or Fulltime).
   - **d)** Calculate the payment based on their pay rate.
   - **e)** Calculate the taxes and other benefits.
   - **f)** Sort the employee based on the type of payment(Direct Deposit or Check)
   - **g)** Generate the pay slips.

3) I will create an Abstract Employee class which holds the methods like Employee ID, Pay Rate. This abstract class will be extended by the two sub classes namely Hourly Employee and Monthly Payment employee which extends the method of abstract classes and have a private methods like number of hours worked in Hourly employee and number of days worked in Monthly employee which returns number of hours and days worked by the specific employee and there is an employee Id verification method where the payment class refers to the employee class if the employee Id is valid and which type of employee it belongs to (Hourly or monthly).

   The second main class is Payment which takes Employee Id as an input, this employee ID will be validated and categorized in the constructor of the payment if this is invalid then it will be returned. If it is valid then the control will be given to the methods up on the result from the employee classes. There are totally 4 methods two methods are used to calculate the salary according to the pay rate of particular employee and it will transfer the control to the other two methods namely direct deposit and Check according to the preference of the employee which will create direct deposit or check.

The third class is the controller class which will create objects to the employee class and instantiate objects to the payment class, in the payment class the object is given with the input of the employee Id and in the constructor of the payment class the Employee Id is validated and then transferred to the Hourly payment method or monthly payment method. These methods will use the methods of the pay rate (employee class) to get the details of pay rate for particular employee Id. After calculating the payment the direct deposit method or check method will be called and the payment will be posted.

4) I have created an abstract class of shapes from which all the other classes are inherited.

Shapes.java

```
import java.io.*;
import java.util.List;
import java.util.LinkedList;

public abstract class Shape{
	private static int counter = 0;
	private int idnumber;
	public Shape()
	{
		idnumber = ++counter;
	}

	public int idNumber(){ return idnumber; }
	public abstract void shapename();
	public String toString()
	{
		return String.format("%14s : %s", getClass().getName(), idnumber);
	}
	public abstract String getname();


}
```

Square.java

```
public class Square extends Shape{
	public Square()
	{
		super();
```

```
	}
	public void shapename()
	{

		System.out.println("Square");
	}

	public String getname()
	{
		return "Square";
	} }
```

Rectangle.java

```
public class Rectangle extends Shape{
	public Rectangle()
	{
		super();
	}
	public void shapename()
	{
		System.out.println("Rectangle");
	}

	public String getname()
	{
		return "Rectangle";
	}
}
```

Triangle.java

```
public class Triangle extends Shape{
	public Triangle()
	{
		super();
	}
	public void shapename()
	{
		System.out.println("Triangle");
	}
```

```
        public String getname()
        {
                return "Triangle"; } }
```

Circle.java

```
public class Circle extends Shape{
        public Circle()
        {
                super();
        }
        public void shapename()
        {
                System.out.println("Circle");
        }

        public String getname()
        {
                return "Circle";
        }
}
```

The test class will create the shapes and sort them according to the alphabetical order and display them.

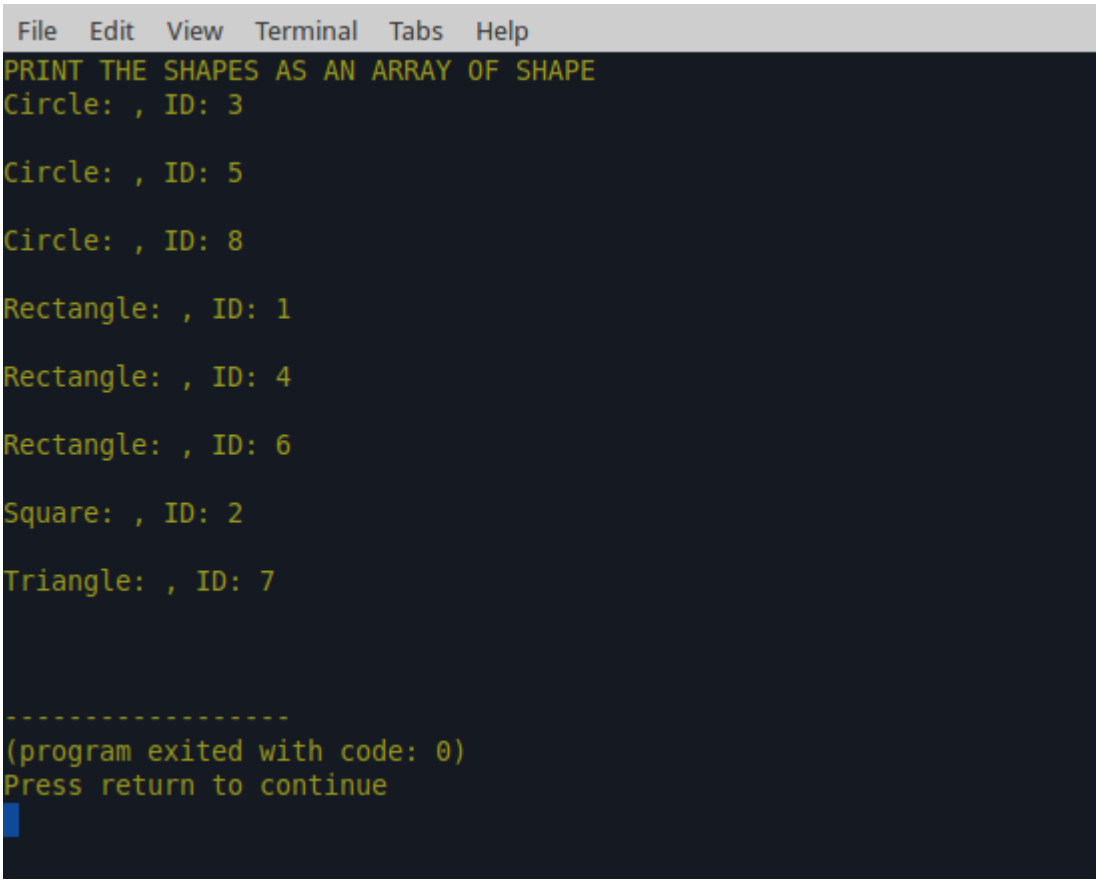
ShapeTest.java

```
public class ShapeTest{

private Shape shapes[];

  public void createShapes() {

    shapes = new Shape[8];

    shapes[0]  = new Rectangle();
    shapes[1]  = new Square();
    shapes[2]  = new Circle();
    shapes[3]  = new Rectangle();
    shapes[4]  = new Circle();
    shapes[5]  = new Rectangle();
```

```
    shapes[6] = new Triangle();
    shapes[7] = new Circle();

  }

  public void printShapes() {

    System.out.println("PRINT THE SHAPES AS AN ARRAY OF SHAPE");
    for ( int i = 0; i < shapes.length; i++ ) {
      System.out.println(shapes[i].getname() + ": , ID: " +
                shapes[i].idNumber());
      System.out.println();
    }
  }

  public void sorting () {
    for (int i = 0; i < shapes.length; i++) {
      Shape k = shapes[i];
      int position = i;
      while (position > 0 && compareShapes(shapes[position-1], k) > 0) {
        shapes[position] = shapes[position-1];
        position--;
      }
      shapes[position] = k;
    }
  }

  private int compareShapes (Shape s1, Shape s2) {
    if ((s1.getname()).equals(s2.getname()))
      return s1.idNumber() - s2.idNumber();
    else return (s1.getname()).compareTo(s2.getname());
  }

  public static void main (String[] args) {
    ShapeTest shapeTest = new ShapeTest();
    shapeTest.createShapes();
    shapeTest.sorting();
    shapeTest.printShapes();
  }
}
```

The screen shot of the output is give in the next page.

Output is:


File Edit View Terminal Tabs Help
PRINT THE SHAPES AS AN ARRAY OF SHAPE
Circle: , ID: 3
Circle: , ID: 5
Circle: , ID: 8
Rectangle: , ID: 1
Rectangle: , ID: 4
Rectangle: , ID: 6
Square: , ID: 2
Triangle: , ID: 7
(program exited with code: 0)
Press return to continue