Java programming I

Abstract – Interface – Polymorphism
Lab Guide 5

Session Objectives

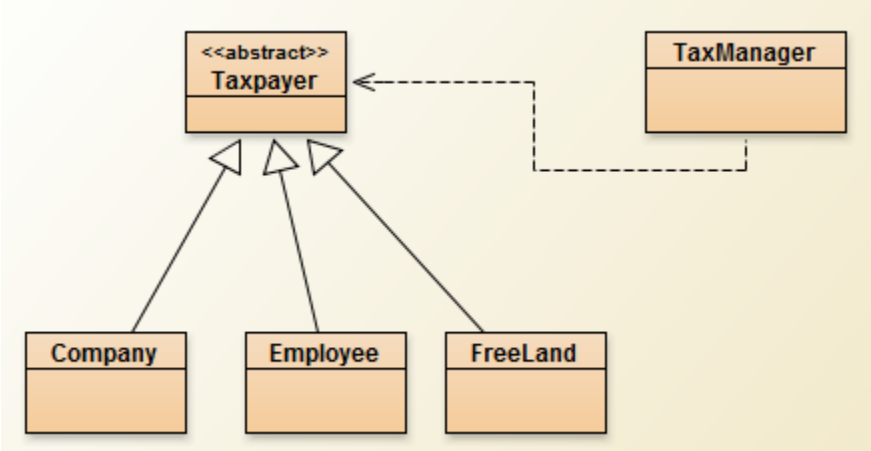
In this session, you will be practicing with:

- Creation and using interface
- Creation and using abstract class
- Polymorphism

Exercise 3 Applying polymorphism (30 minutes).

The following code and video shows how to to use polymorphism

Scan and retype the code below and in video. Discuss with your class mate or instructor.




```
ublic abstract class Taxpayer{

    private String id;

    public String getId(){
        return id;
    }
    public abstract double pay();
}
```

```
public class Company extends Taxpayer
{
```

```
    public double pay(){
        return 1000;
     }
}
```

```
public class Company extends Taxpayer
{
    public double pay(){
        return 1000;
     }
}
```

```
public class FreeLand extends Taxpayer
{
    public double pay(){
        return 10;
    }
}
```

```
public class Employee extends Taxpayer
{
    public double pay(){
        return 100;
     }
}
```

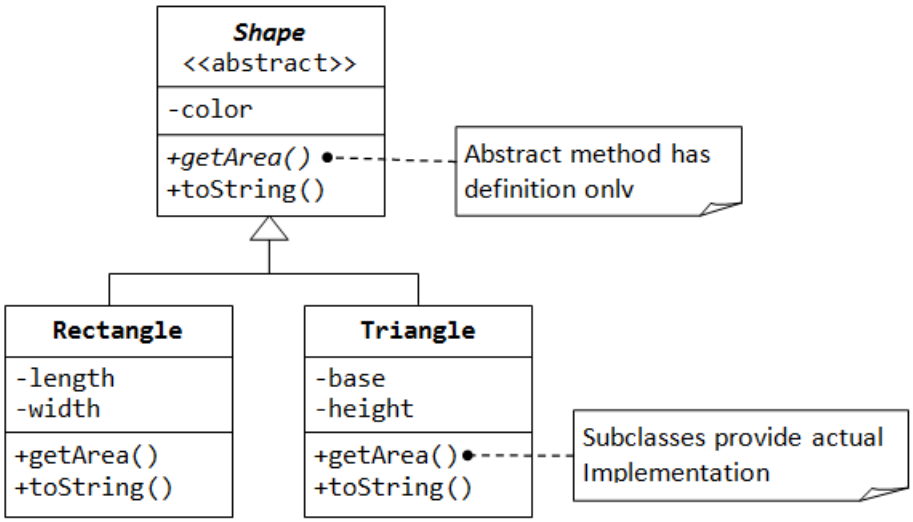
```
public class TaxManager {

     private static final int MAX = 100;
     private Taxpayer[] list = new Taxpayer[MAX];
     private int count = 0;

     public boolean addTaxpayer(Taxpayer taxpayer) {
         if (count >= MAX) {
             return false;
         }
         list[count++] = taxpayer;
         return true;
     }
```

```
    public double getTax() {
        double sum = 0;
        for (int i = 0; i < count; i++) {
            //method pay is polymorphic because we do not
            // know exact object is instance of which class
            sum += list[i].pay();
        }
        return sum;
    }
}
```

Part 2 – Lab Assignment (60 minutes)

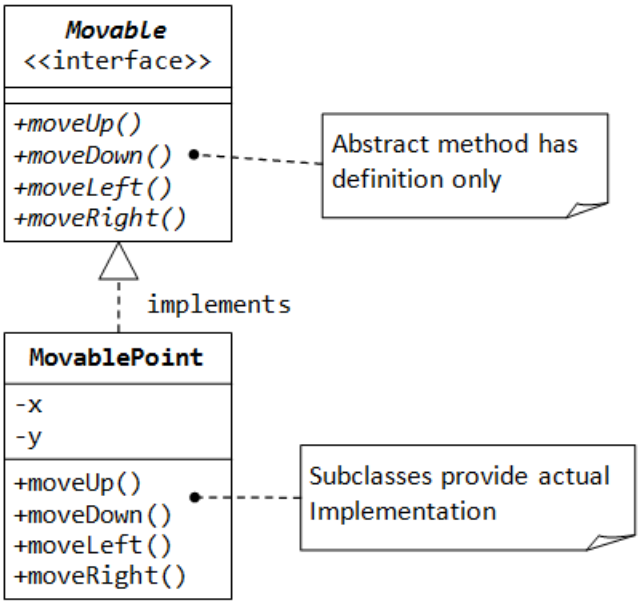
Do the following assignments. Discuss with your class-mates and your instructor if needed.

Exercise 1: Write a program with has design as shown bellow



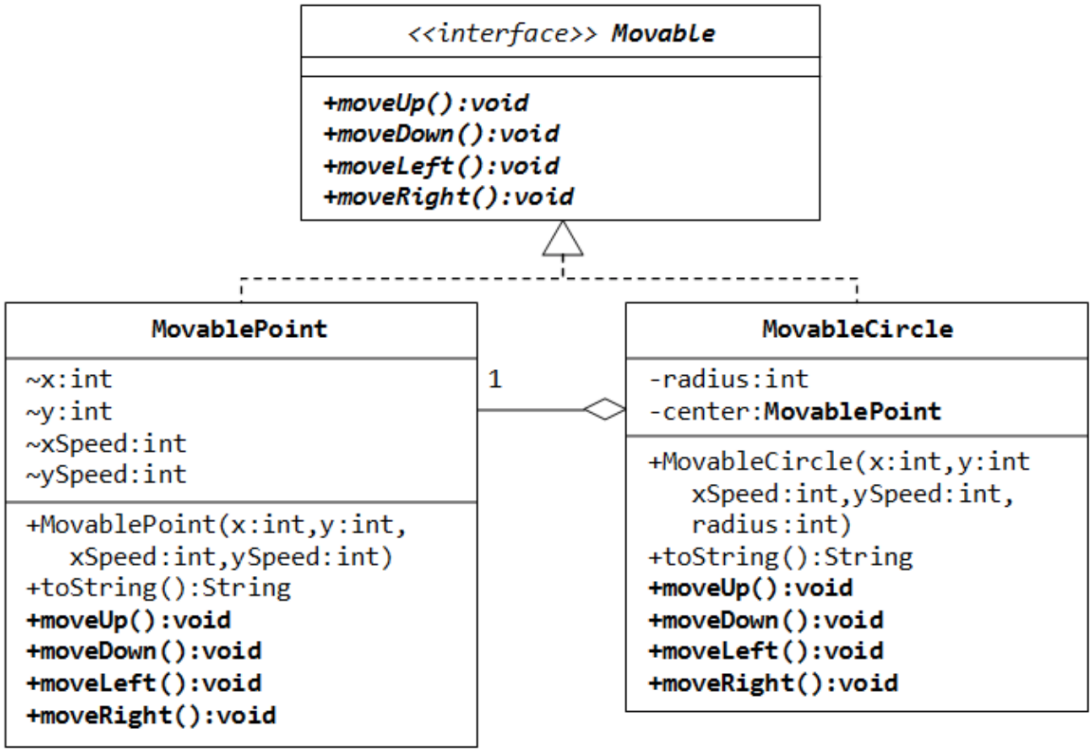
Derive two subclasses named `Rectangle` and `Triangle` from the `abstract` class Shape. Two subclasses must implement abstract methods getArea() declare in Shape class. Write another class called `TestShape` with `main()` method for testing `Rectangle` and `Triangle` class.

Exercise 2: **:** Write a program with has design as shown bellow



The class `MovablePoint` is implement the `Movable` interface and and provide their own implementation to the `abstract` methods defined in the interface `Movable`. In the `Movable` class, when `moveUp()` is called, that is y++, `moveDown()` is called, that is y--. And similarly for `moveLeft()` and `moveRight()`

Exercise 3: Write a program that has design bellow


<<interface>> Movable
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
MovablePoint
~x:int
~y:int
~xSpeed:int
~ySpeed:int
+MovablePoint(x:int,y:int,
xSpeed:int,ySpeed:int)
+toString():String
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
1
MovableCircle
-radius:int
-center:MovablePoint
+MovableCircle(x:int,y:int
xSpeed:int,ySpeed:int,
radius:int)
+toString():String
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void