LAB 6 – Polymorphism

Objectives:

At the end of this lab, the students are able to

i. Use *instanceof* operator and downcasting concept in polymorphism.
ii. Apply abstract class and method and to implement method overriding and overloading.
iii. Using final class and method.
iv. Implementing interface.

6.1 Activity 1

6.2.1 Objective

Implement polymorphism concept and use *instanceof* and downcasting to select different form of objects at runtime.

6.2.2 Problem Description

The UML diagram in Figure 1 shows the relationship for Shape.

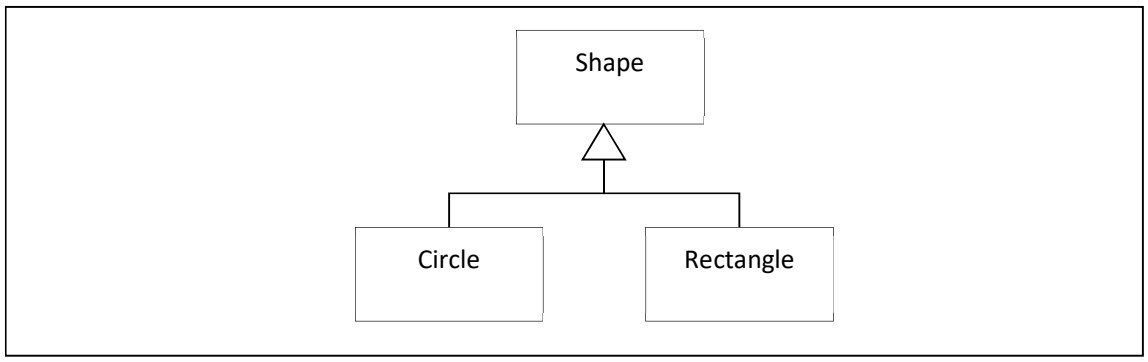

Figure 1

Shape has the following requirements:
- Name of the shape
- A method to calculate area that returns a default as 0.0f

Circle has the following requirements:
- Has a radius
- Has a method to calculate area of circle as 3.14 x radius x radius

Rectangle has the following requirements:
- Has width and length
- A method to calculate area of rectangle as width x length

At the main program, create the object by passing the following values:
- For circle, define object name as “Circle” and radius = 4
- For rectangle, define object name as “Rectangle”, width = 2 and length = 4.

If the current object is rectangle, update the length = 6. Finally, display the area of circle and rectangle.

[Estimated Time: 45 minutes]

6.2.3 Solution Activity 1

Algorithms

Step 1 – Declare a class Shape, attribute that represents Shape Name.
Step 2 – Define getter and setter for each instance variable in Shape’s class
Step 3 – Define method to calculate area of shape and return as 0.0f.
Step 4 – Define *toString()* method to display the Shape Name.
Step 5 – Define the class Circle with radius as an instance variables.
Step 6 – Define getter and setter for instance variable.
Step 8 – Define method to calculate area of circle.
Step 9 – Define *toString()* method to display the Shape Name and area of circle.
Step 10 – Define the class Rectangle with width and length as an instance variable.
Step 11 – Define getter and setter for instance variable.
Step 12 – Define method to calculate area of rectangle.
Step 13 – Define *toString()* method to display the Shape Name and area of rectangle.
Step 14 – Create class ShapeTest to test each of the subclass.
Step 15 – Instantiate class Circle by passing (“Circle”, 4).
Step 16 – Instantiate class Rectangle by passing (“Rectangle”, 2, 4).
Step 17 – Define an array that represent the Circle and Rectangle object.
Step 18 – Perform looping for each of elements of array.
Step 19 – If current object is Circle, update width = 6.
Step 20 – Display the area of respective object via *toString()* method.

Programming

Step 1: Go to CSF3043 folder and create sub-folder called as Lab – Chapter 6.

Step 2: Go to CSF3043 –> Lab – Chapter 6 folder and create Sub-working folder called Activity 1.

Step 3: Open IDE NetBeans or JCreator.

Step 4: Create new file/class called Shape.java

Step 5: Define the class Shape and instance variables as below:

```
/**
 * ------------------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name   : Shape
  * Description    : To implement parent class for Shape.
  * Creation Date  : 14 Aug 2011
  * Modified Date  : None
  * Version        : Version 1.00
  * ------------------------------------------------------------------------
 */
public class Shape {

     //Define instance variables.........
     private String ShapeName;
```

Step 6: Define the constructor for Shape class as below:

```
    //Define constructor 1.....
    public Shape(String ShapeName) {
      //Initialize the value.....
      setShapeName(ShapeName);
    }
```

Step 7: Define getter and setter for each of instance variable.

```
    //Define getter and setter for each instance variables....
    public void setShapeName(String ShapeName) {
        this.ShapeName = ShapeName;
    }

    public String getShapeName() {
        return (this.ShapeName);
    }
```

Step 8: Define the additional method to calculate area and to display the result.

```
    //Calculate area....
    public double calculateArea()
    {
         return (0.0f);
    }

    // Return string representation of a Shape object
    public String toString()
    {
       return String.format( "%s %s\n",
          "Shape Name         = ", getShapeName()   );
    }
}
```

Step 9: Compile the Shape.java.

Step 10: Open new file/class called Circle.java

Step 11: Define the instance variables and constructor for class Circle as below:

```
/**
 * ---------------------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name   : Cicrle
  * Description    : To implement the business requirements for class Circle.
  * Creation Date  : 14 Aug 2011
  * Modified Date  : None
  * Version        : Version 1.00
  * ---------------------------------------------------------------------------
  */
public class Circle extends Shape {

     //Define instance  variables...
     private int radius;

     public Circle(String ShapeName, int radius) {
          super(ShapeName);
          setRadius(radius);
     }
```

Step 12: Define the getter and setter for each of the instance variables as below:

```
     //Define method to calculate area...
     public double calculateArea()
     {
         return ( (float) (3.14 * getRadius() * getRadius())) ;
     }

     public String toString(){
         return String.format( "%s\n%s %.2f\n",
                               super.toString(), "Area of Circle    = ", calculateArea());
     }
}
```

Step 13: Define the additional method to calculate area and display the result.

```
    //Define method to calculate area...
    public double calculateArea()
    {
        return ( (float) (3.14 * getRadius() * getRadius())) ;
    }

    public String toString(){
        return String.format( "%s\n%s %.2f\n",
                              super.toString(), "Area of Circle    = ", calculateArea());
    }
}
```

Step 14: Compile the Circle.java.

Step 15: Open new file/class called Rectangle.java.

Step 16: Define the instance variables and constructor for class Rectangle as below:

```
/**
 * ---------------------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name  : Rectangle
  * Description   : To implement the business requirements for class Rectangle.
  * Creation Date : 14 Aug 2011
  * Modified Date : None
  * Version       : Version 1.00
  * ---------------------------------------------------------------------------
 */
public class Rectangle extends Shape {

    //Define instance  variables...
    private int width;
    private int length;

    public Rectangle(String ShapeName, int width, int length) {
        //Called constructor from super class......
          super( ShapeName);
          setLength(length);
          setWidth(width);
    }
```

Step 17: Define the getter and setter for each of the instance variables as below:

```
//Define getter & setter...
 public int getWidth() {
     return (this.width);
 }

 public void setWidth(int width) {
     this.width = width;
 }

 public int getLength() {
     return (this.length);
 }

 public void setLength(int length) {
     this.length = length;
 }
```

Step 18: Define the additional method to calculate area and display the result.

```
    //Define method to calculate area...
    public double calculateArea()
    {
        return ( (float) (getWidth() * getLength())) ;
    }

    public String toString(){
        return String.format( "%s\n%s %.2f\n",
                              super.toString(), "Area of Rectangle = ", calculateArea());
    }
}
```

Step 19: Compile the Rectangle.java

Step 20: Open new file/class called ShapeTest.java.

Step 21: Instantiate, define an array to store these objects and initialize the array.

```
/**
 * ---------------------------------------------------------------------------------------
 * @author       : Mohamad Nor Hassan
 * Program Name  : ShapeTest
 * Description   : This is the main program to invoke the implementation of class Shape(),
 *                 Circle() and Rectangle().
 * Creation Date : 14 Aug 2011
 * Modified Date : None
 * Version       : Version 1.00
 * ---------------------------------------------------------------------------------------
 */
public class ShapeTest {
    public static void main(String[] args) {
        // TODO code application logic here

        //Instantiate the Circle object....
        Circle objCircle = new Circle("Circle", 4);

        //Instantiate the Rectangle object....
        Rectangle objRectangle = new Rectangle("Rectangle", 2, 4);

        //Create two (2) elements of array Shape....
        Shape myShape[] = new Shape[2];

        //Initialize the array
        myShape[0] = objCircle;
        myShape[1] = objRectangle;

        System.out.println("Shape process polymorphically...!");
        System.out.println("");
```

Step 22: Read an array and verify if current object is Circle, then set width to 6. Finally, print the result.

```
        //Processing an array...
        for (Shape currentShape : myShape )
        {
            if ( currentShape instanceof Rectangle )
            {
                Rectangle myRectangle = (Rectangle) currentShape;
                myRectangle.setLength(6);

                //Display result...
                System.out.println(myRectangle);
            }
            else
            {
                System.out.println(currentShape);
            }
        }
    }
}
```

Step 23: Compile your program and evaluate the output.

6.2 Activity 2

6.2.1 Objective

Demonstrate how to use abstract class and method, and finally implementing method overriding and overloading.

6.2.2 Problem Description

The UML diagram in Figure 2 shows the relationship between Consultant and Java Consultant. The Java Consultant class will inherit the attributes and behavior of the Consultant class.

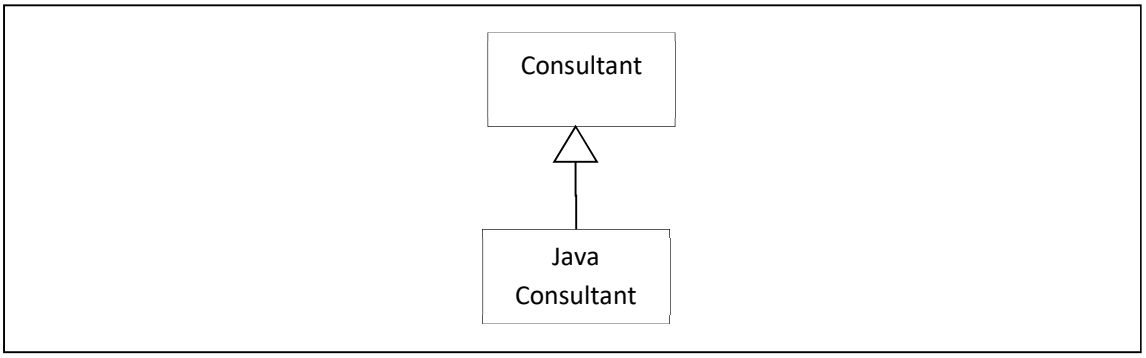

Figure 2

The Consultant class has the following requirements:

- Consultant name, company and age.
- A method to display details of consultant. This method consists of two (2) different implementation: First implementation – will display consultant name and company.
- Second implementation – will display consultant name, company and age. The value of age is obtained through parameter passing.
- Has an abstract method for calculate and displaying salary. The salary is based on MYR800.00 per working day.

The Java Consultant class has the following requirements:

- Has a number of working days and a salary rate as MYR800.00 per day.
- Has a method to calculate and display salary. The salary is based on number of working days x salary rate per day.

Write a program to demonstrate the use of abstract class and method. In addition, show when the method overloading and overriding take place in your program.

[Estimated Time: 35 minutes]

6.3 Activity 3

6.3.1 Objective

Using *final* class and final method.

6.3.2 Problem Description

Write a simple program by using final class and method to calculate the commission of selling the magazine for company XYZ Sdn. Bhd. The commission rate is MYR2.50 per selling magazine. Your output should include person name, number of magazine sold and the commission amount.

[Estimated Time: 20 minutes]

6.4 Activity 4

6.4.1 Objective

Implement an interface in your program.

6.4.2 Problem Description

Write a program to demonstrate the use of interface via class *CreditCard().* Credit card has credit card number, transaction date, transaction description and transaction amount. The interface *Payment()* has a method to calculate the current payment amount of credit card that has to be paid for a month. Your output should display a credit card statement for a month in which consists of credit card number, several transaction details, total of transaction amounts and the current payment amount.

[Estimated Time: 40 minutes]

Lab Exercise

Objective:

Propose a solution using inheritance, composition and polymorphism, concept.

Problem Description:

Using classes, design an online address book to keep track of the names, addresses, phone numbers, and birthdays of family members, close friends, and certain business associates. Your program should be able to handle a maximum of 500 entries.

- Define the class Address that can store a street address, city, state, and zip code. Use the appropriate methods to print and store the address. Also, use constructors to automatically initialize the data members.
- Define the class ExtPerson using the class Person, the class Date (as designed in Lab-Chapter3 Part 1-Activity 1. You may modified it accordingly), and the class Address.
- Add extension classes to categorize the person as a family member, friend or business associate. Differentiate each class by adding its own possible data members. For example, business associate might has company information.
- Use constructors to automatically initialize the data members for each defined classes where applicable.
- Add (or override) methods to print and store the appropriate information based for each defined classes previously.

The program should perform the following operations:
- Store the data according to the categories.
- Sort the address book by last name.
- Search for a person by last name.
- Search for a person by category
- Print the address, phone number, and date of birth (if available) of a given person.

[Estimated Time: 60 minutes]