LAB 5 – Inheritance

Objectives:

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

i. Understand *"is-a"* relationship and demonstrate the basic inheritance concept.
ii. Use super to call parent constructor and qualifier (.) to access a member of parent class from subclass, and how to use *"has-a"* relationship.
iii. Applying the concept of overriding via inheritance.

5.1 Activity 1

5.1.1 Objective

Using a *super()* to access constructor of superclass and qualifier (.) to access methods from parent class. Finally, how to use "has-a" relationship.

5.1.2 Problem Description

The implementation for business requirements of the class *Shape* contains has a relationship with three (3) other classes; class *Rectangle, Square,* and *Coordinate* as shown in Figure 1.

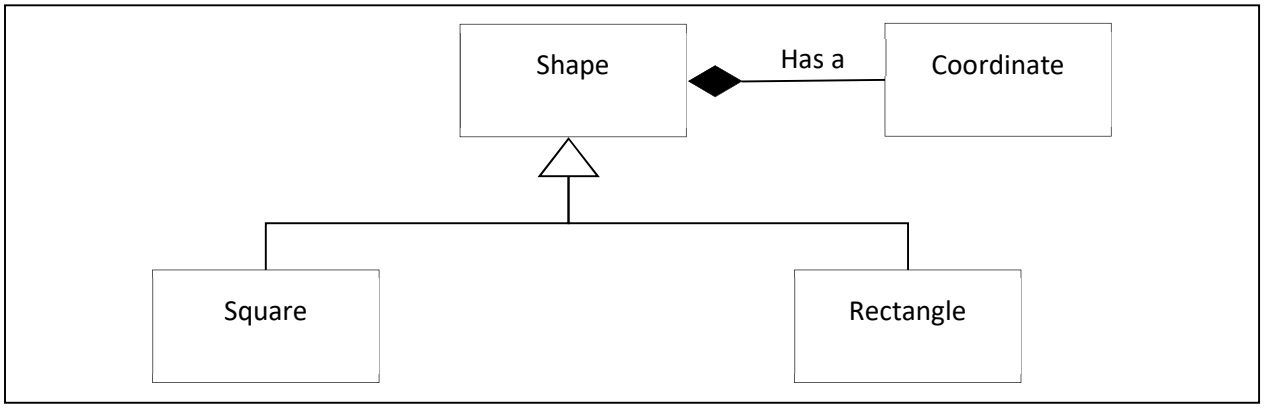

Figure 1

The class *Shape* provides a generic operation to calculate the length for width, height and area for *Square and* Rectangle, and display purposes of each of the coordinate (x, y) respectively. Class *Square* and *Rectangle* contain only the specific method to display the area for the class.

Write a program to demonstrate the implementation of the above class. To test the class, create a main class known as *ShapeTest*. Instantiate the object for *mySquare* and *myRectangle* by passing the parameters (0,0, 0,2, 2,2, 2,0) and (0,0, 0,2, 5,2, 5,0) respectively.

[Estimated Time: 60 minutes]

5.1.3 Solution Activity 2

Algorithms

Step 1 – Declare a class Coordinate, attributes that represent x-coordinate and y-coordinate and respective methods.
Step 2 – Define the class Shape with four (4) instance variables represent the point(x, y).
Step 3 – Define getter and setter for each instance variable in Shape's class
Step 4 – Define additional method to find the height, width of and to calculate the area
Step 5 – Define the method *toString()* to display the result.
Step 6 – Define class Square.
Step 7 – Create constructor by calling constructor from super class by passing four (4) that represent coordinate (x, y) respectively
Step 8 – Create method to calculate and display the area of square.
Step 9 – Define class Rectangle.
Step 10 – Create constructor by calling constructor from super class by passing four (4) that represent coordinate (x, y) respectively
Step 11 – Create method to calculate and display the area of rectangle.
Step 12 – Create class ShapeTest to test each of the subclass.
Step 13 – Instantiate class Square by passing (0,0, 0,2, 2,2, 2,0).
Step 14 – Invoke method to calculate the area of square.
Step 15 – Instantiate class Rectangle by passing (0,0, 0,2, 5,2, 5,0).
Step 16 – Invoke method to calculate the area of rectangle.

Programming

Step 1: Go to CSF3043 folder and create sub-folder called as Lab – Chapter 5.

Step 2: Go to CSF3043 –> Lab – Chapter 5 folder and create Sub-working folder called Activity 2.

Step 3: Open IDE NetBeans or JCreator.

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

Step 5: Define the class Coordinate instance variables as below:

```
/**
 * ---------------------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name   : Coordinate
  * Description    : To implement the business requirements for class Cooridnate.
  * Creation Date  : 04 Feb 2011
  * Modified Date  : None
  * Version        : Version 1.00
  * ---------------------------------------------------------------------------
 */
public class Coordinate {
    //Define instance variables....
     private double x; // x coordinate
     private double y; // y coordinate
```

Step 6: Define pre-defined constructor for Coordinate class

```
    //Define constructor....
    public Coordinate(double xCoordinate, double yCoordinate) {
      x = xCoordinate; // set x
      y = yCoordinate; // set y
    }
```

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

```
    //Define getter and setter...
    public double getX() {
        return (this.x);
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return (this.y);
    }

    public void setY(double y) {
        this.y = y;
    }
```

Step 8: Define the additional method to display the coordinate (x,y) as below

```
   // return string representation of cooridnate object
   public String toString()
   {
      return String.format( "( %.1f, %.1f )", getX(), getY() );
   }
}
```

Step 9: Compile the Coordinate.java.

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

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

```
/**
 * ------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name   : Shape
  * Description    : To implement parent class for Shape.
  * Creation Date  : 04 Feb 2011
  * Modified Date  : None
  * Version        : Version 1.00
  * ------------------------------------------------------------
  */
public class Shape {

     //Define instance variables.........
     private Coordinate point1; // first endpoint
     private Coordinate point2; // second endpoint
     private Coordinate point3; // third endpoint
     private Coordinate point4; // fourth endpoint

     //Define constructor 1.....
     public Shape(double x1, double y1,
                  double x2, double y2,
                  double x3, double y3,
                  double x4, double y4) {

       //Initialize the points.....
       point1 = new Coordinate( x1, y1 );
       point2 = new Coordinate( x2, y2 );
       point3 = new Coordinate( x3, y3 );
       point4 = new Coordinate( x4, y4 );
     }
```

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

```
    //Define getter and setter for each instance variables....
    public Coordinate getPoint1() {
        return (this.point1);
    }

    public void setPoint1(Coordinate point1) {
        this.point1 = point1;
    }

    public Coordinate getPoint2() {
        return (this.point2);
    }

    public void setPoint2(Coordinate point2) {
        this.point2 = point2;
    }

    public Coordinate getPoint3() {
        return (this.point3);
    }

    public void setPoint3(Coordinate point3) {
        this.point3 = point3;
    }

    public Coordinate getPoint4() {
        return (this.point4);
    }

    public void setPoint4(Coordinate point4) {
        this.point4 = point4;
    }
```

Step 13: Define the additional method to retrieve height and width the area as below:

```
    //Define additional method....
    //get height...
  public double getHeight(){
          return Math.abs( getPoint2().getY() - getPoint1().getY());
    }

    public double getWidth(){
       return Math.abs( getPoint4().getX() - getPoint2().getX() );
    }
```

Step 14: Define the additional method to calculate area and display the coordinate for each point.

```
    //Calculate area....
    public double calculateArea()
    {
         return (getWidth() * getHeight());
    }

    // return string representation of a Shape object
    public String toString()
    {
       return String.format( "%s:\n%s",
          "Coordinates of Shape", getCoordinatesAsString() );
    }

    // return string containing coordinates as strings
    public String getCoordinatesAsString()
    {
       return String.format(
          "%s, %s, %s, %s\n", point1, point2, point3, point4 );
    }
}
```

Step 15: Compile the Shape.java.

Step 16: Open new file/class called Square.java.

Step 17: Write the implementation for class Square as below:

```
 * ------------------------------------------------------------------------
  * @author        : Mohamad Nor Hassan
  * Program Name   : Square
  * Description    : To implement the business requirements for class Square.
  * Creation Date  : 04 Feb 2011
  * Modified Date  : None
  * Version        : Version 1.00
  * ------------------------------------------------------------------------
 */
public class Square extends Shape {

    public Square( double x1, double y1,
                   double x2, double y2,
                   double x3, double y3,
                   double x4, double y4) {

        //Called constructor from super class......
          super( x1, y1, x2, y2, x3, y3, x4, y4 );
    }

    public void squareArea(){
        System.out.println(super.toString());
        System.out.printf("Area of Square = %.1f\n", super.calculateArea());
    }
}
```

Step 18: Compile the Square.java.

Step 19: Open new file/class called Rectangle.java.

Step 20: Write the implementation for class Rectangle as below:

```
/* ------------------------------------------------------------------------------------
 * @author        : Mohamad Nor Hassan
 * Program Name   : Rectangle
 * Description    : To implement the business requirements for class Rectangle.
 * Creation Date  : 04 Feb 2011
 * Modified Date  : None
 * Version        : Version 1.00
 * ------------------------------------------------------------------------------------
 */
public class Rectangle extends Shape {

    public Rectangle(double x1, double y1,
                     double x2, double y2,
                     double x3, double y3,
                     double x4, double y4) {

        //Called constructor from super class......
          super( x1, y1, x2, y2, x3, y3, x4, y4 );
    }

    public void rectangleArea(){
         System.out.println(super.toString());
         System.out.printf("Area of Rectangle = %.1f\n", super.calculateArea());
     }
}
```

Step 21: Compile the Rectangle.java.

Step 22: Open new file/class called ShapeTest.java.

Step 23: Define the class ShapeTest() as below:

```
/* ------------------------------------------------------------------------------------
 * @author        : Mohamad Nor Hassan
 * Program Name   : ShapeTest
 * Description    : This is the main program to invoke the implementation of class Shape(),
 *                  Square() and Rectangle().
 * Creation Date  : 04 Feb 2011
 * Modified Date  : None
 * Version        : Version 1.00
 * ------------------------------------------------------------------------------------
 */
public class ShapeTest {
    public static void main(String[] args) {
        // TODO code application logic here
```

Step 24: Instantiate Square class, calculate and display the result

```
        //Instantiate the Square object....
        Square mysquare = new Square(0,0, 0,2, 2,2, 2,0);
        mysquare.squareArea();

        System.out.println(" ");
```

Step 25: Instantiate Rectangle's class and calculate and display the result.

```
        //Instantiate the Rectangle object....
        Rectangle myrectangle = new Rectangle(0,0, 0,2, 5,2, 5,0);
        myrectangle.rectangleArea();
    }
}
```

Step 26: Compile the ShapeTest.java.

Step 27: Run the program and evaluate the output.

5.2 Activity 2

5.2.1 Objective

Implement inheritance relationship.

5.2.2 Problem Description

The ABC company consists of two (2) of employees; permanent and contract employee. Both employees consist of common attributes such as employee number, employee name and type of employee. The calculation of salary is different and based on the following formula:

- Permanent employee – Basic salary + allowance (RM300.00)
- Contract employee – no of hours x RM30.00 per hour

Draw an UML diagram to show the relationship between the classes. Based on your UML diagram, write a program to display employee number, name, type of employee and the salary.

[Estimated Time: 25 minutes]

5.3 Activity 3

5.3.1 Objective

Applying the concept of overriding via inheritance.

5.3.2 Problem Description

Copy your solution in Activity 2 and paste into CSF3043 -> Lab – Chapter 5 -> Activity 3. Based on the solution you wrote in Activity 2, modify the common method double *calculateEmpSalary()* to calculate the salary as below:

Employee salary = basic salary

In your Permanent Employee and Contract Employee's class, rename the method to calculate salary as double *calculateEmpSalary()*. Create a main program as *EmployeeTest* and override the implementation of calculating salary at permanent and contract employee.

[Estimated Time: 20 minutes]

Lab Exercises

Objective:

Propose a solution using inheritance concept.

Problem Description:

Using inheritance concept, design an online profile record to keep track the names, address, phone numbers, date of birth and certain specific related information for members at a university. Use the appropriate methods to print and store the profile records.

Draw an UML diagram to show inheritance hierarchy members at a university. Use class Person as the superclass of the hierarchy. The class Employee, Student and Alumnus are derived from the class Person. Then extend Employee with classes Academician and Administration. Meanwhile, extend Student with classes UndergraduateStudent and PostGraduateStudent. Continue to extend the hierarchy as deep (i.e., as many levels) as possible. For example, DoctoralStudent and MastersStudent might be subclasses of PostGraduateStudent.

Based on your UML diagram, write a program by considering the following information:

- Identify generic variables and operations for representing all superclasses. For example, class Person provides a generic operation to display first name and last name as the person's full name, address and phone numbers.
- Identify specific variables and operations for representing all subclasses. For example, class Employee might contain instance variables such as employee id, faculty/school name, position, salary and hire date.
- Use your own creativity to key-in and display the output by manipulating the code.

[Estimated Time: 60 minutes]