COMP2026 Problem Solving Using Object Oriented Programming

Laboratory 11

Part A Discovery Exercises

Task 1: Upcasting and Downcasting

a) Create the following classes in an IntelliJ project.

```
public class S {

    private String name;

    public S(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return name;
    }

    public void doSomething(){
        System.out.println("I am a shape.");
    }
}
```

```
public class C extends S{

    private double radius;

    public C(String name, double radius)
    {
        super(name);
        this.radius = radius;
    }

    public double getRadius()
    {
        return radius;
    }

    public void doSomething(){
        System.out.println("I am a circle.");
    }
}
```

```
public class STester {
    public static void main(String[] args) {
        new STester().runApp();
    }

    private void runApp(){
        S a = new S("ShapeA");
        S b = new C("Circle1", 5);
        //upcast the C object to parent type S
    }
}
```

b) Add the following **`doSomething()`** method calls in the **`STester`** class.

```
public class STester {
    public static void main(String[] args) {
        new STester().runApp();
    }

    private void runApp() {
        S a = new S("ShapeA");
        S b = new C("Circle1", 5);
        a.doSomething();
        b.doSomething();
    }
}
```

c) Run the **`STester`** program again and paste the output below.

```

```

d) Add the following **`getName()`** method calls in the **`ShapeTester`** class.

```
public class STester {
    public static void main(String[] args) {
        new STester().runApp();
    }

    private void runApp() {
        S a = new S("ShapeA");
        S b = new C("Circle1", 5);
        a.doSomething();
        b.doSomething();
        System.out.println(a.getName());
        System.out.println(b.getName());
    }
}
```

e) Run the **`STester`** program again and paste the output below.

f) Add the following **`getRadius()`** method call in the **`ShapeTester`** class.

```
public class STester {
    public static void main(String[] args) {
        new STester().runApp();
    }

    private void runApp() {
        S a = new S("ShapeA");
        S b = new C("Circle1", 5);
        a.doSomething();
        b.doSomething();
        System.out.println(a.getName());
        System.out.println(b.getName());
        System.out.println(b.getRadius());
    }
}
```

g) Run the **`STester`** program again and paste the output below. The program should produce errors.

h) Downcast **`b`** to **`C`** type and call **`getRadius()`** method again.

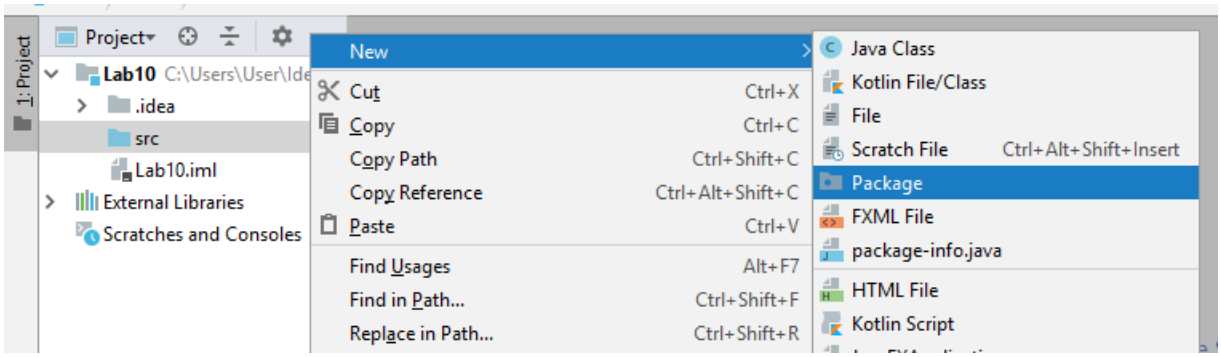
```
public class STester {
    public static void main(String[] args) {
        new STester().runApp();
    }

    private void runApp() {
        S a = new S("ShapeA");
        S b = new C("Circle1", 5);
        a.doSomething();
        b.doSomething();
        System.out.println(a.getName());
        System.out.println(b.getName());
        System.out.println(((C) b).getRadius());
    }
}
```

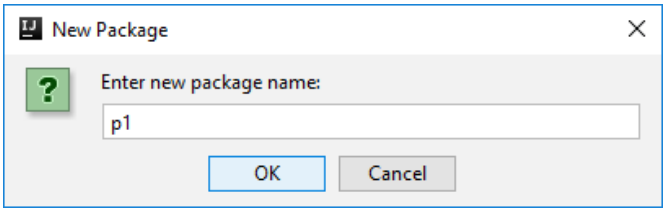
i) Run the **`STester`** program again and paste the output below.

Task 2: Understanding Access Modifier

a) Create an IntelliJ project.

b) Create a package called p1.
Right click the src folder → New → Package and then input p1 and click OK.





c) Right click the package p1 and create a class called Alpha with the following content.

```
package p1;
public class Alpha {
  public int pub;
  protected int prot;
  int pack;
  private int pri;

  public void methodA() {
    System.out.println(this.pub);
    System.out.println(this.prot);
    System.out.println(this.pack);
    System.out.println(this.pri);
  }
}
```

The methodA() tries to print each instance variable. There is no error in the

program because the method is in the class Alpha, so it has access to all members of its own class.

Access Modifier	Public	Protected	No modifier	Private
Can access in Alpha?	Y	Y	Y	Y

d) Create another class called Beta in package p1.

```
package p1;
public class Beta {
  public void methodB() {
    Alpha alphaObj = new Alpha();
    System.out.println(alphaObj.pub);
    System.out.println(alphaObj.prot);
    System.out.println(alphaObj.pack);
    System.out.println(alphaObj.pri);
  }
}
```

Class Alpha and Beta are inside the same package. Fill in the following table with 'Y' for yes and 'N' for no.

Access Modifier	Public	Protected	No modifier	Private
Can access in Beta?				

e) Create another class called Gamma in package p1.

```
package p1;
public class Gamma extends Alpha{
  public void methodG() {
    System.out.println(this.pub);
    System.out.println(this.prot);
    System.out.println(this.pack);
    System.out.println(this.pri);
  }
}
```

Gamma is a subclass of class Alpha inside the same package. Fill in the following table with 'Y' for yes and 'N' for no.

Access Modifier	Public	Protected	No modifier	Private
Can access in Gamma?				

f) Create another package called p2. Create a class called Delta in package p2.

```
package p2;
import p1.*;
public class Delta extends Alpha{
  public void methodD() {
    System.out.println(this.pub);
    System.out.println(this.prot);
    System.out.println(this.pack);
    System.out.println(this.pri);
  }
}
```

Delta is a subclass of class Alpha in a different package. Fill in the following table with 'Y' for yes and 'N' for no.

Access Modifier	Public	Protected	No modifier	Private
Can access in Delta?				

g) Create another class called Epsilon in package p2.

```
package p2;
import p1.*;
public class Epsilon{
  public void methodE() {
    Alpha alphaObj = new Alpha();
    System.out.println(alphaObj.pub);
    System.out.println(alphaObj.prot);
    System.out.println(alphaObj.pack);
    System.out.println(alphaObj.pri);
  }
}
```

Class Alpha and Epsilon are inside different packages. Fill in the following table with 'Y' for yes and 'N' for no.

Access Modifier	Public	Protected	No modifier	Private
Can access in Epsilon?				

Part B Programming Exercises

Task 1: Creating ABCEmployee Objects

Complete the **`runApp()`** method in the given **`ABCEmployeeTester.java`** to test the given **`ABCEmployee`** class.

- Create a default ABCEmployee object and use set methods to update the default ABCEmployee with
  - Name: "Susan Meyers"
  - Id: 47988
  - Department: "Accounting"
  - Position: "Vice President"
- Create an ABCEmployee object with "Mark Jones" as the name and 39119 as the id and then use set methods to update the ABCEmployee with
  - Department: "IT"
  - Position: "Programmer"
- Create an ABCEmployee object with "Joy Rogers" as the name, 81774 as id, "Manufacturing" as the department and "Engineer" as the position.
- Print all the information of Susan Meyers by using get methods.
- Print the 3 objects.

Sample output:

```
Name: Susan Meyers
ID No.: 47899
Department: Accounting
Position: Vice President

ABCEmployee{name='Susan Meyers', id=47899, dept='Accounting', position='Vice President'}
ABCEmployee{name='Mark Jones', id=39119, dept='IT', position='Programmer'}
ABCEmployee{name='Joy Rogers', id=81774, dept='Manufacturing', position='Engineer'}
```

Task 2: Writing Subclasses

a) Write a class named **`Circle`** that is a subclass of the given **`Shape`** class. The class should have the following members:

- A private double field **`radius`** represents the radius of the circle.
- A default constructor that sets the name to empty string and radius to 1.
- A constructor that accepts the **`name`** and **`radius`** as arguments.
- Override the **`area`** method to return the area of the circle. Hint: Use Math.PI
- Override the **`perimeter`** method to return the perimeter of the circle.
- Override the **`toString`** method to return the string representation of the circle in the following format:

```
Circle: name
Radius: radius
```

b) Write a class named **`Triangle`** that is a subclass of the given **`Shape`** class. The class should have the following members:
- Three private double fields for the lengths of the 3 sides of the triangle.
- A default constructor that sets the **`name`** to empty string and all side lengths to 1.
- A constructor for equilateral triangle that only accepts the **`name`** and one side length as arguments.
- A constructor for triangle that accepts the **`name`** and the 3 side lengths as arguments.
- Override the **`area`** method to return the area of the triangle (by Heron's formula).
- Override the **`perimeter`** method to return the perimeter of the triangle.
- Override the **`toString`** method to return the string representation of the triangle in the following format:

```
Triangle: name
Side lengths: side1 side2 side3
```

c) Complete the **`MyShapeList.java`** to work with a list of Shape objects.
- Create the following objects in the **`addShapes()`** method and add the objects into the **`shapeList`**.
  - A default Circle object.
  - A Circle object with "MyCircle" as the name and 10 as radius.
  - A default Triangle object.
  - An equilateral Triangle object with "EquilateralTriangle" as the name and 6 as the side lengths.
  - A Triangle object with "MyTriangle" as the name and 6, 7, 8 as the side lengths.
- Print all the objects in the **`shapeList`** in the **`printList()`** method.
- Return the sum of the areas of all the shapes in the **`totalArea()`** method.
- Return the sum of the perimeters of all the shapes in the **`totalPerimeter()`** method.
- Return the number of Circle objects in the **`numOfCircle()`** method.
- Return the number of Triangle objects in the **`numOfTrianlge()`** method.

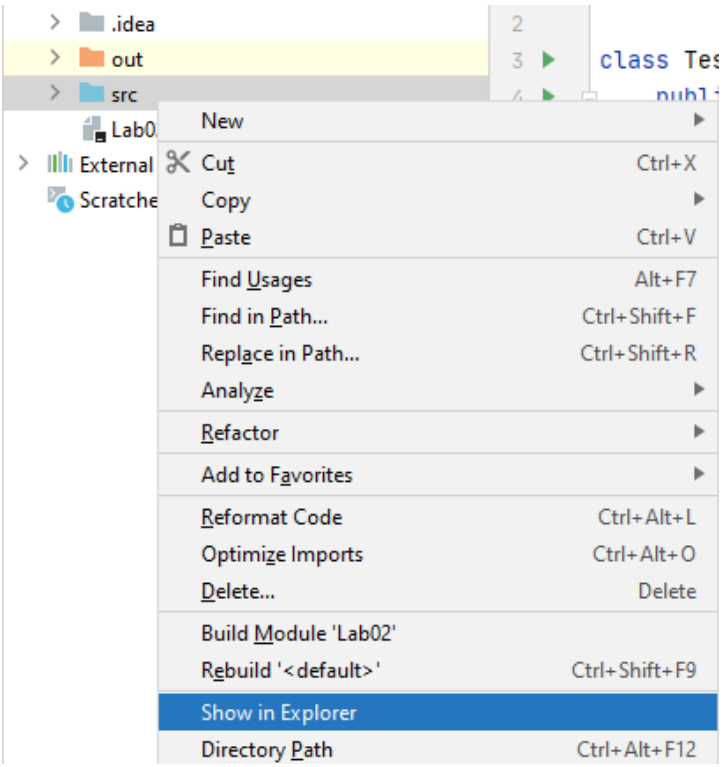
Hint: Use instanceof operator to check the object type.

Sample output:

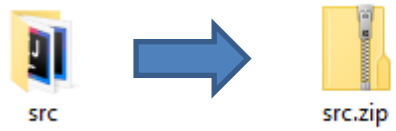
```
==================================
Print List:
Circle:
Radius: 1.0
Circle: MyCircle
Radius: 10.0
Triangle:
Side lengths: 1.0 1.0 1.0
Triangle: EquilateralTriangle
Side lengths: 6.0 6.0 6.0
Triangle: MyTriangle
Side lengths: 6.0 7.0 8.0
==================================
Total Area: 353.66
Total Perimeter: 111.12
No. of Circles: 2
No. of Triangles: 3
```

Part C Submitting Exercises

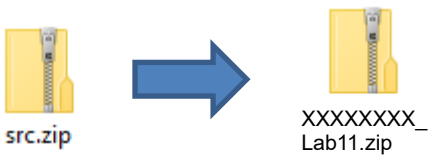
Step 1: Right-click the **src** folder and select **Show in Explorer**



Step 2: Zip the **src** folder into **src.zip**



Step 3: Rename the src.zip file to **XXXXXXXX_lab11.zip** where **XXXXXXXX** is your **student id**



Step 4: Submit **XXXXXXXX_lab11.zip** and **XXXXXXXX_lab11.docx** to Moodle.

References

[1] Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. *Dubuque*, IA: McGraw-Hill.

[2] Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.

[3] Farrell, J. (2012). ***Java programming. Boston, MA: Course Technology Cengage Learning***

[4] Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.

[5] Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.

[6] Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.

[7] Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.

[8] Schildt, H., & Skrien, D. J. (2013). Java programming: A comprehensive introduction. New York: McGraw-Hill.

[9] Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education

[10] Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.

[11] yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming

[12] Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.