



## CSE 215L: Programming Language II Lab

Faculty: Silvia Ahmed, Sec – 12, 13

Lab 08 – Summer 2020

### Objective:

After today's lab, the students should be able:

- To define a subclass from a superclass through inheritance.
- To invoke the superclass's constructors and methods using the **super** keyword.
- To override instance methods in the subclass.
- To distinguish differences between overriding and overloading.
- To explore the **toString()** method in the **Object** class.

### Superclasses and Subclasses

Syntax for Circle class extends the GeometricObject

Subclass                      Superclass  
  
`public class Circle extends GeometricObject`

Syntax for Rectangle class extends the GeometricObject

Subclass                      Superclass  
  
`public class Rectangle extends GeometricObject`

#### GeometricObject

-color: String  
 -filled: boolean  
 -dateCreated: java.util.Date

+GeometricObject()  
 +GeometricObject(color: String, filled: boolean)  
 +getColor(): String  
 +setColor(color: String): void  
 +isFilled(): boolean  
 +setFilled(filled: boolean): void  
 +getDateCreated(): java.util.Date  
 +toString(): String

The color of the object (default: white).  
 Indicates whether the object is filled with a color (default: false).  
 The date when the object was created.

Creates a GeometricObject.  
 Creates a GeometricObject with the specified color and filled values.  
 Returns the color.  
 Sets a new color.  
 Returns the filled property.  
 Sets a new filled property.  
 Returns the dateCreated.  
 Returns a string representation of this object.

#### Circle

-radius: double

+Circle()  
 +Circle(radius: double)  
 +Circle(radius: double, color: String, filled: boolean)  
 +getRadius(): double  
 +setRadius(radius: double): void  
 +getArea(): double  
 +getPerimeter(): double  
 +getDiameter(): double  
 +printCircle(): void

#### Rectangle

-width: double  
 -height: double

+Rectangle()  
 +Rectangle(width: double, height: double)  
 +Rectangle(width: double, height: double, color: String, filled: boolean)  
 +getWidth(): double  
 +setWidth(width: double): void  
 +getHeight(): double  
 +setHeight(height: double): void  
 +getArea(): double  
 +getPerimeter(): double

### Task – 1

(The *Triangle* class) Design a class named **Triangle** that extends **GeometricObject**. The class contains:

- Three **double** data fields named **side1**, **side2**, and **side3** with default values **1.0** to denote three sides of the triangle.
- A no-arg constructor that creates a default triangle.
- A constructor that creates a triangle with the specified **side1**, **side2**, and **side3**.
- The accessor methods for all three data fields.
- A method named **getArea()** that returns the area of this triangle.
- A method named **getPerimeter()** that returns the perimeter of this triangle.
- A method named **toString()** that returns a string description for the triangle.

The formula to compute the area of a triangle:

$$s = (\text{side1} + \text{side2} + \text{side3})/2;$$

$$\text{area} = \sqrt{s(s - \text{side1})(s - \text{side2})(s - \text{side3})}$$

The formula to compute the perimeter of a triangle:

$$\text{perimeter} = \text{side1} + \text{side2} + \text{side3};$$

The **toString()** method is implemented as follows:

```
return "Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3;
```

Draw the UML diagrams for the classes **Triangle** and **GeometricObject** and implement the classes. Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled. The program should create a **Triangle** object with these sides and set the **color** and **filled** properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

### Task – 2

(The *Person*, *Student*, *Employee*, *Faculty*, and *Staff* classes) Design a class named **Person** and its two subclasses named **Student** and **Employee**.

Make **Faculty** and **Staff** subclasses of **Employee**. A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. Use **MyDate** class to create an object for date hired. Java API has the **GregorianCalendar** class in the **java.util** package, which you can use to obtain the year, month, and day of a date. You may use the **GregorianCalendar** class to simplify coding.) A faculty member has office hours and a rank. A staff member has a title. Override the **toString** method in each class to display the class name and the person's name.

Draw the UML diagram for the classes and implement them. Write a test program that creates a **Person**, **Student**, **Employee**, **Faculty**, and **Staff**, and invokes their **toString()** methods.