CSE 215L: Programming language II Lab
Faculty: Dr. Ziaul Hossain (ZHO)
Sec: 05
Lab - 08 [Introduction to OOP], Summer-2020
Lab Instructor: Md. Mustafizur Rahman

Objective:

- OOP in Java
- Class Definition in Java
- Creating Instances of a Class
- UML class and Instance Diagrams

Class & Instances

In Java, a class is a definition of objects of the same kind. In other words, a class is a blueprint, template, or prototype that defines and describes the static attributes and dynamic behaviors common to all objects of the same kind.
An instance is the realization of a particular item of a class. In other words, an instance is an instantiation of a class. All the instances of a class have similar properties, as described in the class definition.

A class can be visualized as a three-compartment box, as illustrated:

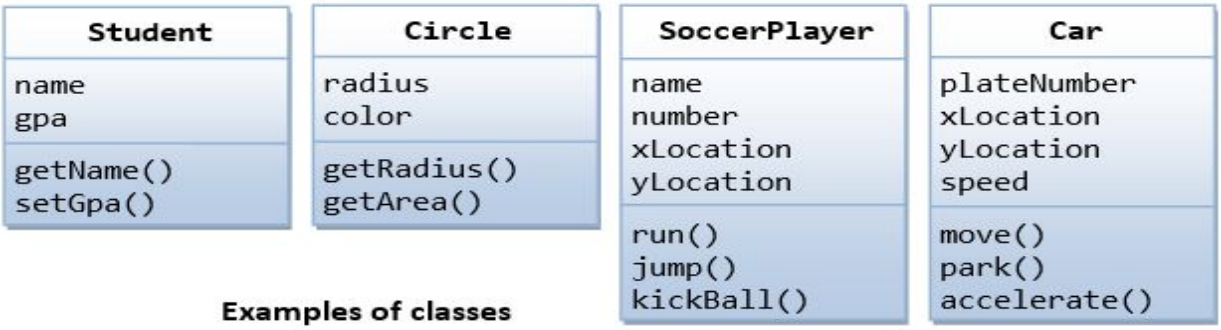
1. *Name* (or identity): identifies the class.
2. *Variables* (or attribute, state, field): contains the *attributes* of the class.
3. *Methods* (or behaviors, function, operation): contains the *dynamic behaviors* of the class.

The followings figure shows a few examples of classes:

Name
Attributes
Dynamic Behaviours

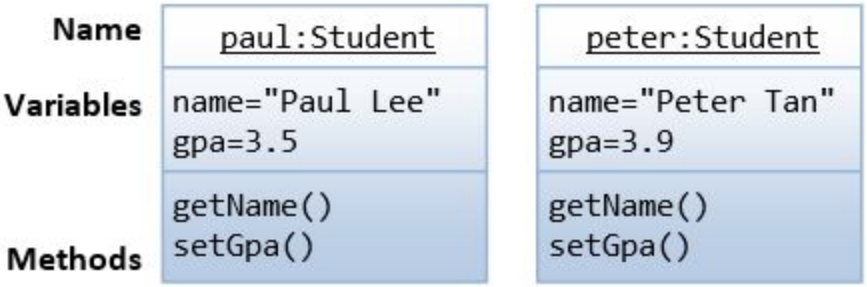
A class is a 3 compartment box

The followings figure shows a few examples of classes:



Examples of classes

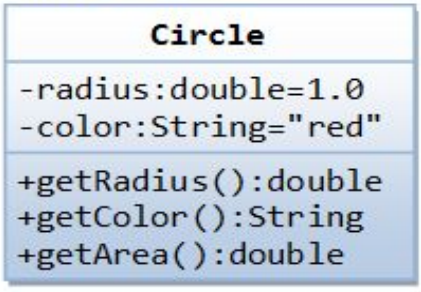
The following figure shows two instances of the class Student, identified as "paul" and "peter".



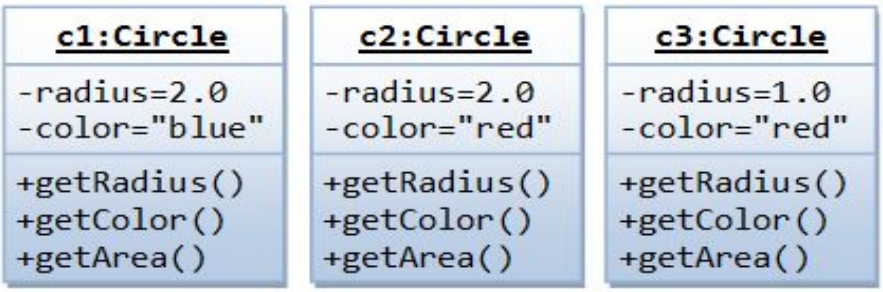
Two instances - paul and peter - of the class Student

An OOP Example:

Class Definition



Instances



```
/*
 * The Circle class models a circle with a radius and color.
 */
public class Circle {    // Save as "Circle.java"
  // Private instance variables
  private double radius;
  private String color;

  // Constructors (overloaded)
  public Circle() {                  // 1st Constructor
    radius = 1.0;
    color = "red";
  }
  public Circle(double r) {          // 2nd Constructor
    radius = r;
    color = "red";
  }
  public Circle(double r, String c) { // 3rd Constructor
    radius = r;
    color = c;
  }

  // Public methods
  public double getRadius() {
    return radius;
  }
  public String getColor() {
    return color;
  }
  public double getArea() {
    return radius * radius * Math.PI;
  }
}
```

TestCircle.java

We shall now write another class called TestCircle, which uses the Circle class. The TestCircle class has a main() method and can be executed.

```
1  /*
2   * A Test Driver for the "Circle" class
3   */
4  public class TestCircle {    // Save as "TestCircle.java"
5     public static void main(String[] args) {   // Program entry point
6        // Declare and Construct an instance of the Circle class called c1
7        Circle c1 = new Circle(2.0, "blue");  // Use 3rd constructor
8        System.out.println("The radius is: " + c1.getRadius());  // use dot operator to invoke
9  member methods
10       System.out.println("The color is: " + c1.getColor());
11       System.out.printf("The area is: %.2f%n", c1.getArea());
12
13       // Declare and Construct another instance of the Circle class called c2
14       Circle c2 = new Circle(2.0);  // Use 2nd constructor
15       System.out.println("The radius is: " + c2.getRadius());
16       System.out.println("The color is: " + c2.getColor());
17       System.out.printf("The area is: %.2f%n", c2.getArea());
18
19       // Declare and Construct yet another instance of the Circle class called c3
20       Circle c3 = new Circle();  // Use 1st constructor
21       System.out.println("The radius is: " + c3.getRadius());
22       System.out.println("The color is: " + c3.getColor());
23       System.out.printf("The area is: %.2f%n", c3.getArea());
 24   }
    }
```

Compile TestCircle.java into TestCircle.class.

<u>Task:</u> Implement the following class and test its methods