Faculty : Mr. Tarek Mizan
Lab Instructor: Nazmul Alam Diptu
Email : nazmul.diptu@northsouth.edu
Class Timing: ST 1:00 PM – 2:30 PM (LIB-611)
Topic: Introduction to OOP

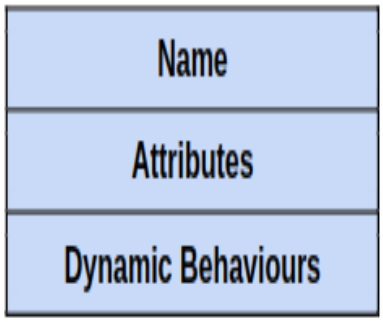
Objective

1. OOP in Java
2. Class Definition in Java
3. Method Overloading
4. Creating Instances of a Class
5. 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:

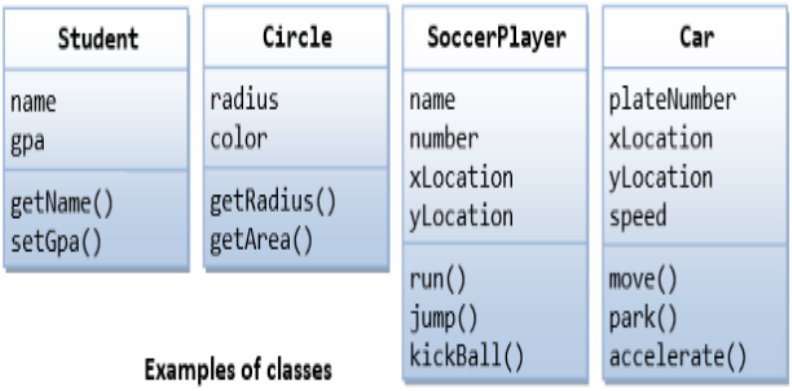
1. Name (or identity): identifies the class.
2. Variables (or attribute, state, field): contains the attributes of the class.
3. Methods (or behaviors, function, operation): contains the dynamic behaviors of the class. The followings figure shows a few examples of classes:

The followings figure shows a few examples of classes:



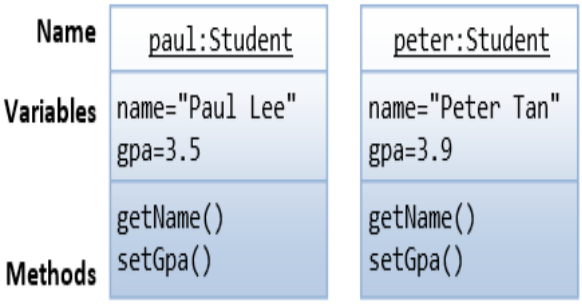
A class is a 3 compartment box

The followings figure shows a few examples of classes:



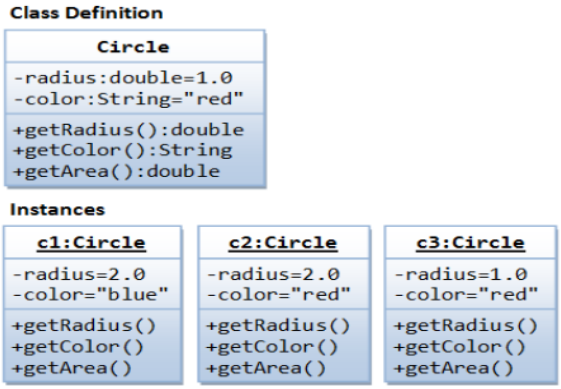
Examples of classes

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:



Method Overloading Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different.

In order to overload a method, the argument lists of the methods must differ in either of these:

1. Number of parameters. For example: This is a valid case of overloading

```
add(int, int)
add(int, int, int)
```

2. Data type of parameters. For example:

```
add(int, int)
add(int, float)
```

3. Sequence of Data type of parameters. For example:

```
add(int, float)
add(float, int)
```

Invalid case of method overloading: When I say argument list, I am not talking about return type of the method, for example if two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw compilation error.

```
int add(int, int)
float add(int, int)
```

Circle.java

```
/* * 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 = Math.abs(r);
        color = "red";
    }

    public Circle(double r, String c) {
        // 3rd Constructor
        radius = Math.abs(r);
        color = c;
    }

    // Public methods
    public double getRadius() {
        return radius;
```

```
    }

    public String getColor() {
        return color;
    }

    public double getArea() {
        return radius * radius * Math.PI;
    }
}
```

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

TestCircle.java

```
/* * A Test Driver for the "Circle" class */

public class TestCircle { // Save as "TestCircle.java"
    public static void main(String[] args) {
        // Declare and Construct an instance of the Circle class called c1
        Circle c1 = new Circle(2.0, "blue"); // Use 3rd constructor
        System.out.println("The radius is: " + c1.getRadius()); // use dot
operator to invoke member methods
        System.out.println("The color is: " + c1.getColor());
        System.out.printf("The area is: %.2f\n", c1.getArea());
        // Declare and Construct another instance of the Circle class
called c2
        Circle c2 = new Circle(2.0); // Use 2nd constructor
        System.out.println("The radius is: " + c2.getRadius());
        System.out.println("The color is: " + c2.getColor());
        System.out.printf("The area is: %.2f\n", c2.getArea());
        // Declare and Construct yet another instance of the Circle class
called c3
        Circle c3 = new Circle(); // Use 1st constructor
        System.out.println("The radius is: " + c3.getRadius());
        System.out.println("The color is: " + c3.getColor());
        System.out.printf("The area is: %.2f\n", c3.getArea());

    }

}
```

Faculty : Mr. Tarek Mizan
Lab Instructor: Nazmul Alam Diptu
Email : nazmul.diptu@northsouth.edu
Class Timing: ST 1:00 PM – 2:30 PM (LIB-611)
Topic: Loops, Jump

Tasks:

1. Implement the following class and test its methods. Also, write a JUnit test class to test it's method.