Object Oriented Programming Concepts in Java

Prof. Siddharth Shah
Department of Computer Engineering
Dharmsinh Desai University

Outline

- OOP Features
- Class
- Object
- Method
- Constructor
- Keyword ***this***
- Garbage Collection

OOP Features

- **Object:** An entity with states(properties, variables) and behavior (methods). It is an instance of a class
- **Class:** A template / blueprint from which object can be created.
- **Abstraction:** It is a way to hide the complexity (how part) of an object and allows the user to use it. It can be achieved through hierarchical classification.
- **Encapsulation:** It is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.

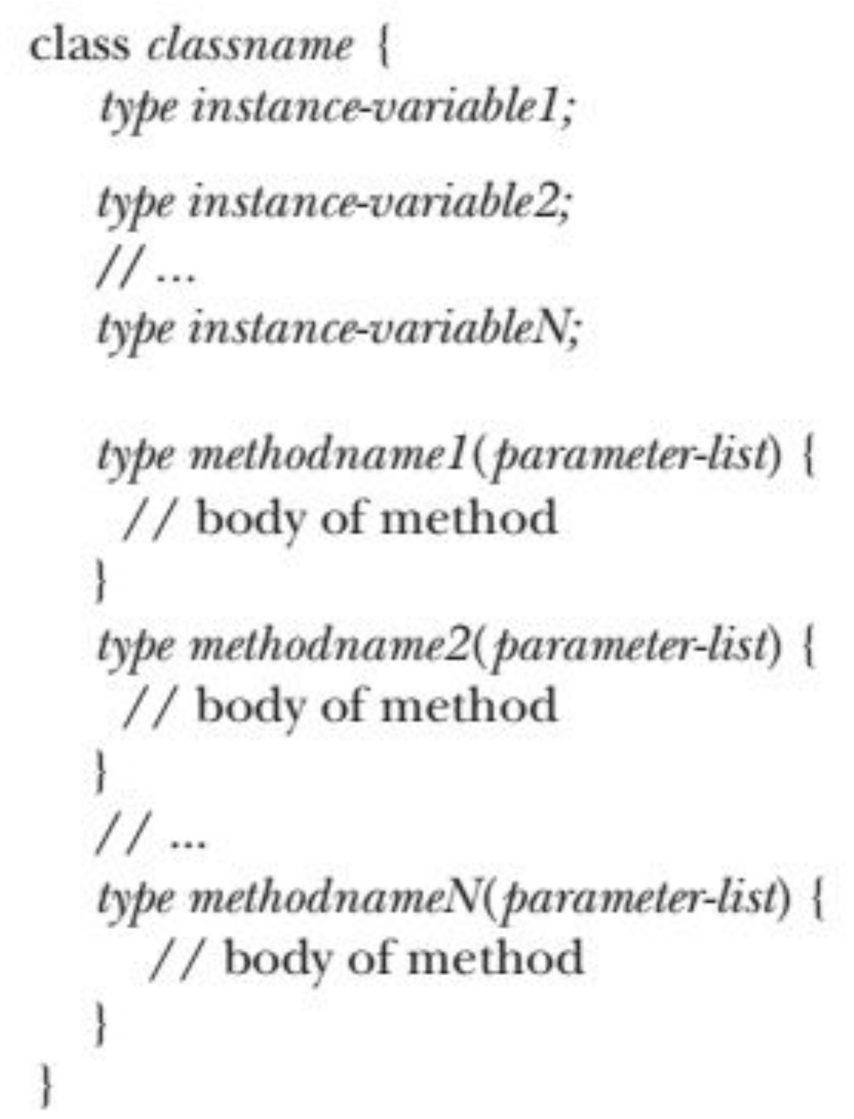
OOP Features (Cont...)

- **Inheritance:** It is the process by which one object acquires the properties of another object. This is important because it supports the concept of hierarchical classification.

- **Polymorphism:** In Greek it means “many forms”. It is a feature that allows one interface to be used for a general class of actions. It is often expressed by the phrase “one interface, multiple methods.”

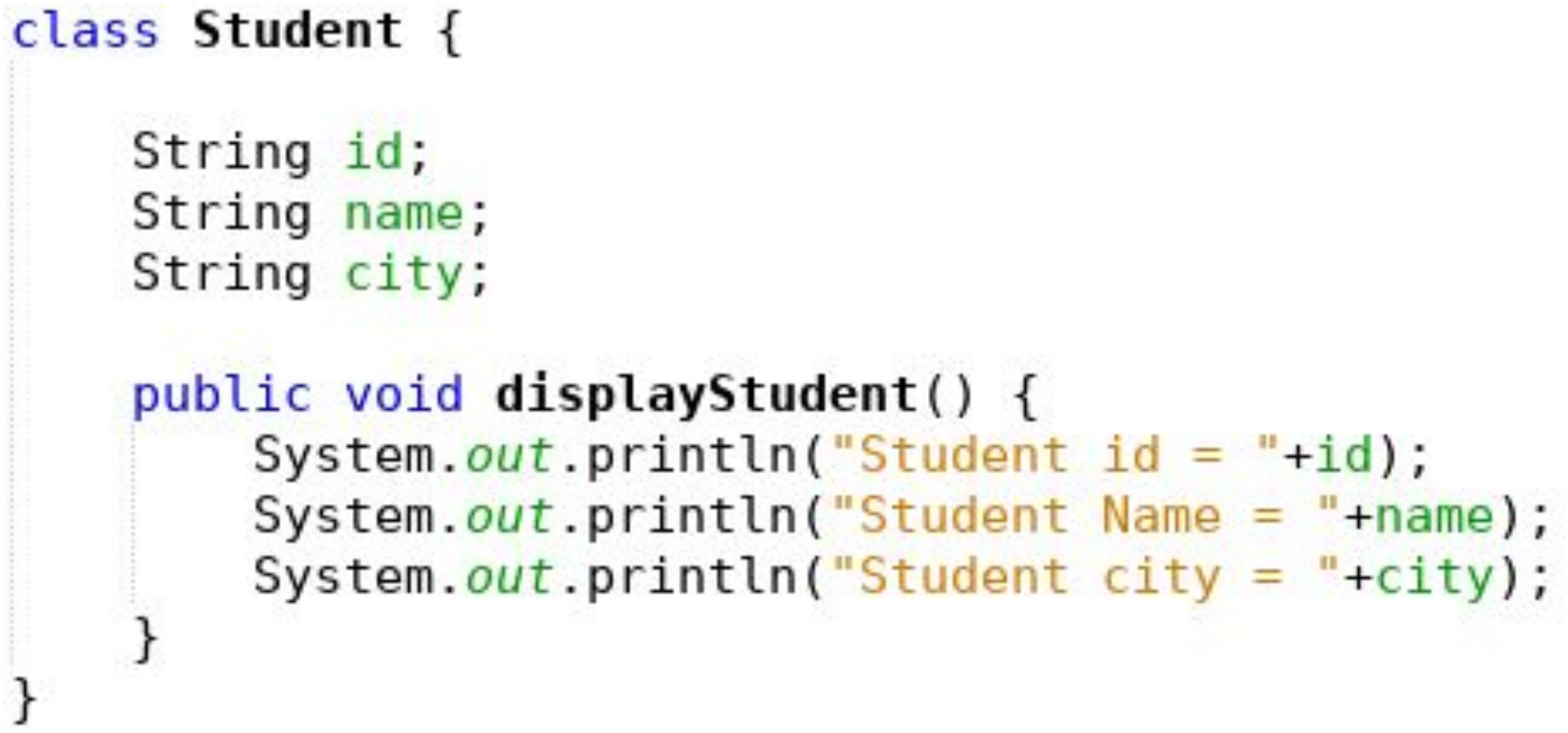
Class

- Class contains two things:
- Data (variables)
- Code (methods)
- Collectively, the methods and variables defined within a class are called members of the class.



```
class classname {
    type instance-variable1;
    type instance-variable2;
    // ...
    type instance-variableN;

    type methodname1(parameter-list) {
      // body of method
    }
    type methodname2(parameter-list) {
      // body of method
    }
    // ...
    type methodnameN(parameter-list) {
        // body of method
    }
}
```

Example: Creating a Class



```java
class Student {

    String id;
    String name;
    String city;

    public void displayStudent() {
        System.out.println("Student id = "+id);
        System.out.println("Student Name = "+name);
        System.out.println("Student city = "+city);
    }
}
```

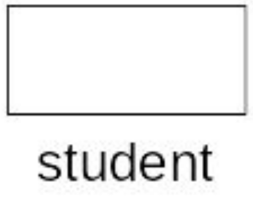
Creating Object

classname class-var;

Statement

Student student;

Effect

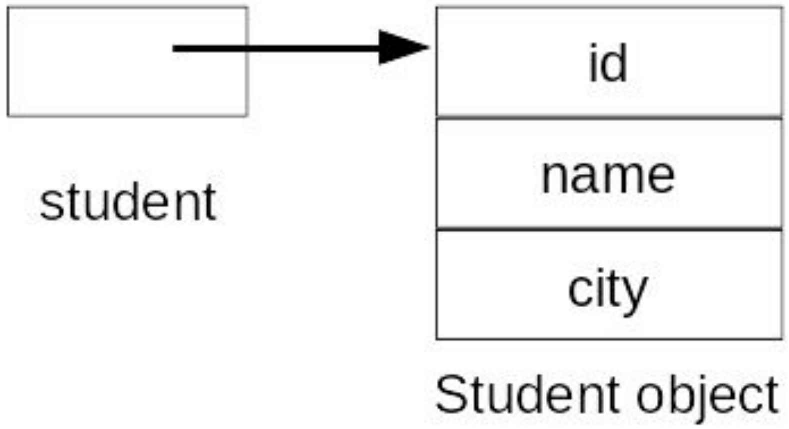


class-var = new classname ( );

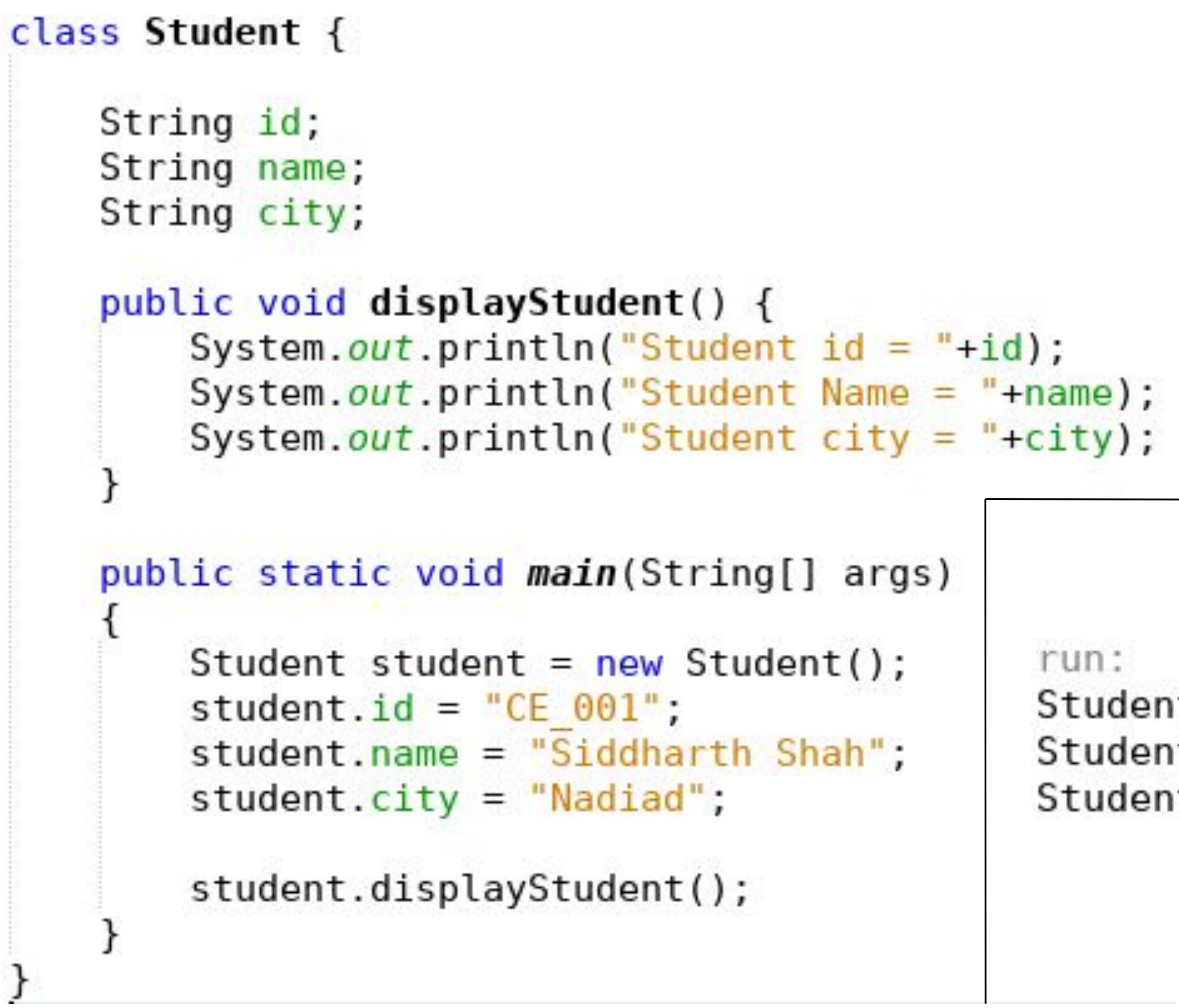
Statement

student = new Student()

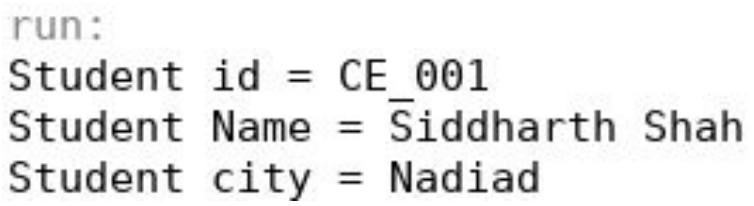
Effect



Example: Creating an Object



```java
class Student {

    String id;
    String name;
    String city;

    public void displayStudent() {
        System.out.println("Student id = "+id);
        System.out.println("Student Name = "+name);
        System.out.println("Student city = "+city);
    }

    public static void main(String[] args)
    {
        Student student = new Student();
        student.id = "CE_001";
        student.name = "Siddharth Shah";
        student.city = "Nadiad";

        student.displayStudent();
    }
}
```



```
run:
Student id = CE_001
Student Name = Siddharth Shah
Student city = Nadiad
```

Example: Creating an Object in Other Class

```
class Student {

    String id;
    String name;
    String city;

    public void displayStudent() {
        System.out.println("Student id = "+id);
        System.out.println("Student Name = "+name);
        System.out.println("Student city = "+city);
    }
}

public class StudentDemo{
    public static void main(String[] args)
    {
        Student student = new Student();
        student.id = "CE_001";
        student.name = "Siddharth Shah";
        student.city = "Nadiad";

        student.displayStudent();
    }
}
```

```
run:
Student id = CE_001
Student Name = Siddharth Shah
Student city = Nadiad
```

Method

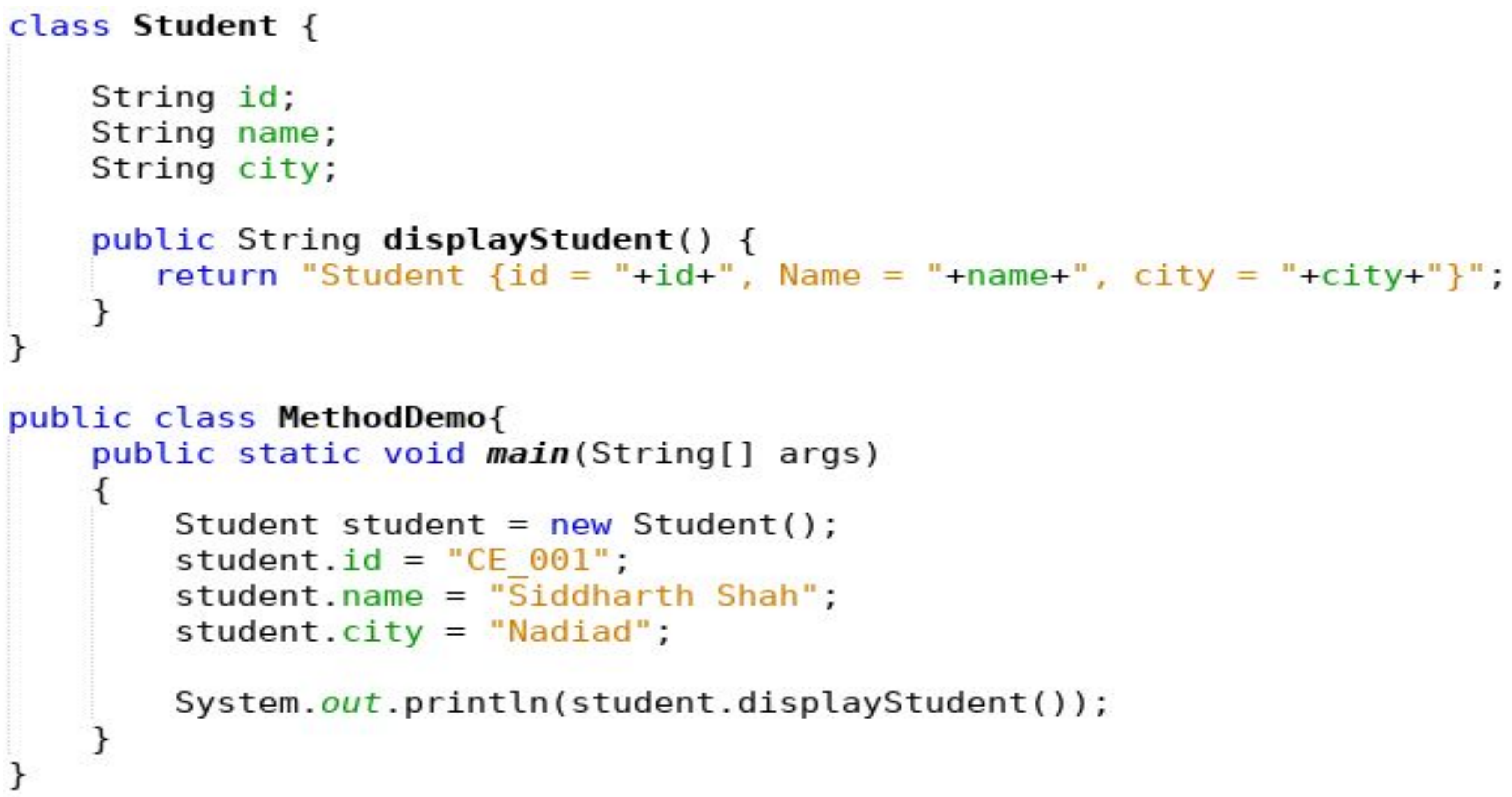
- General form of a method

```
type name(parameter-list) {
    // body of method
}
```

- Methods that have a return type other than **void,** return a value to the calling routine using the following form of the return statement:

```
return value;
```

Example: Method with return value



```
class Student {

    String id;
    String name;
    String city;

    public String displayStudent() {
        return "Student {id = "+id+", Name = "+name+", city = "+city+"}";
    }
}

public class MethodDemo{
    public static void main(String[] args)
    {
        Student student = new Student();
        student.id = "CE_001";
        student.name = "Siddharth Shah";
        student.city = "Nadiad";

        System.out.println(student.displayStudent());
    }
}
```

Output:

```
Student {id = CE_001, Name = Siddharth Shah, city = Nadiad}
```

Example: Method with Parameters

```java
class Box {

    double width;
    double height;
    double depth;

    // compute and return volume
    double volume() {
        return width * height * depth;
    }

    // sets dimensions of box
    void setDim(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }
}
```

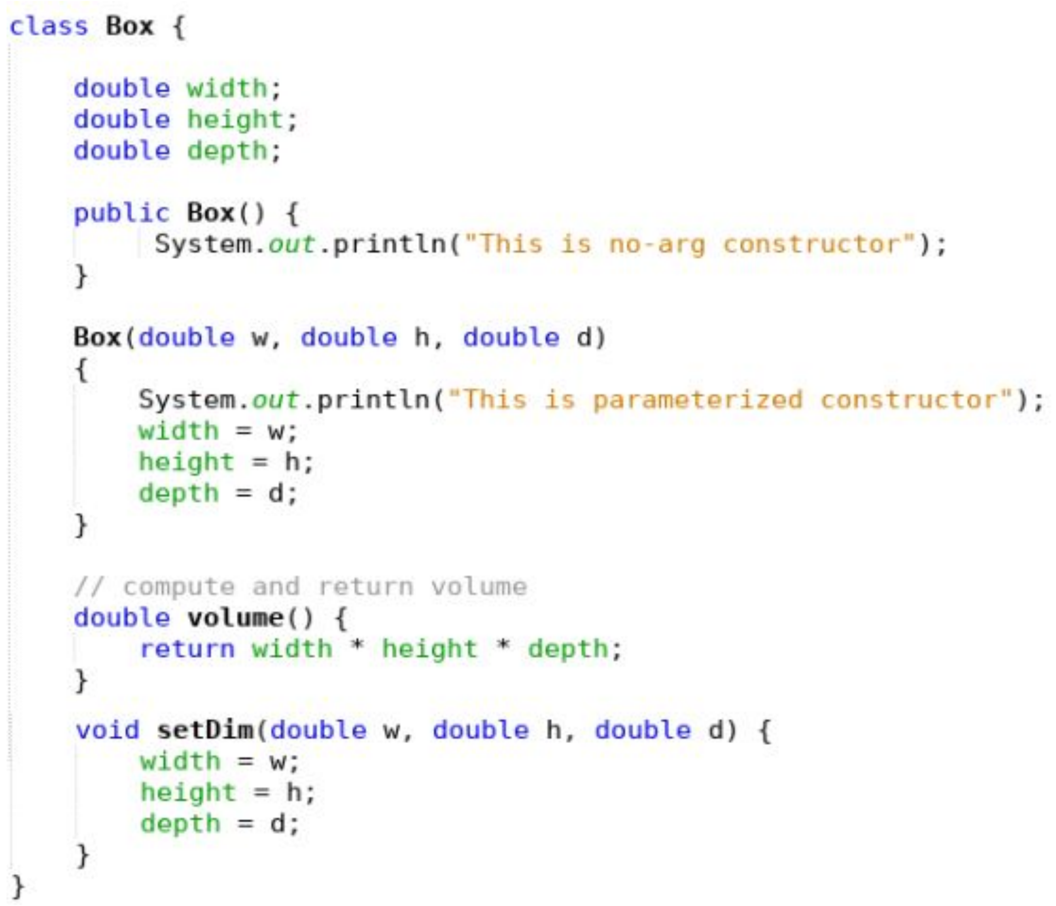
```java
public class ParameterizedMethodDemo {

    public static void main(String args[]) {
        Box mybox = new Box();
        double vol;

        // initialize mybox
        mybox.setDim(10, 20, 15);

        // get volume of mybox
        vol = mybox.volume();
        System.out.println("Volume is " + vol);
    }
}
```

```
run:
Volume is 3000.0
```

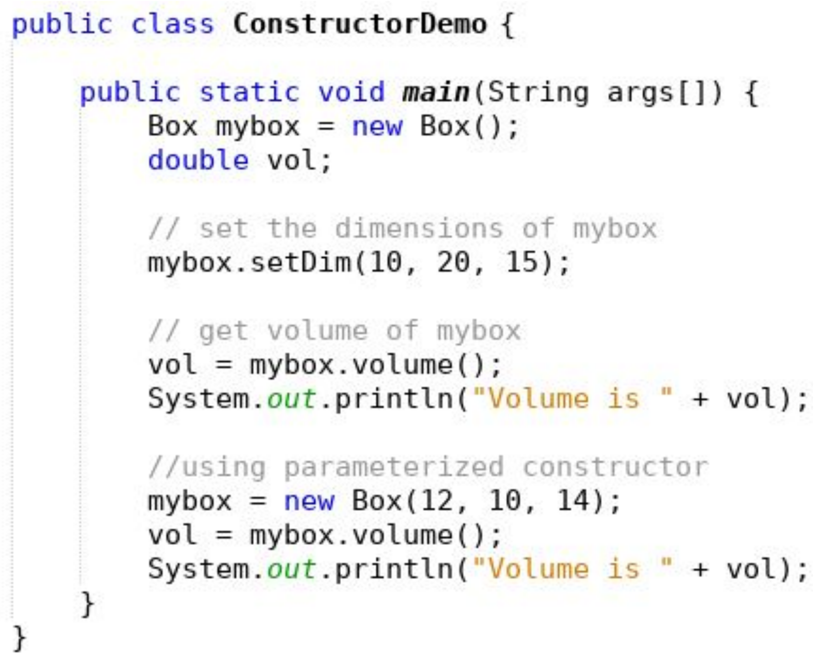
Constructor

- In Java, a constructor is a block of codes similar to the method.
- It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated.
- If there is no constructor available in the class, Java compiler provides a default no-arg constructor.
- There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

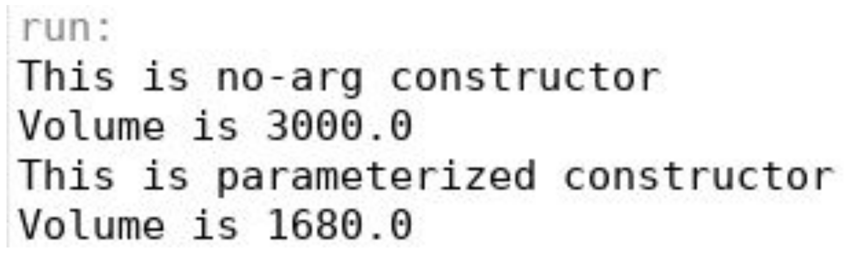
Constructor



```java
class Box {

    double width;
    double height;
    double depth;

    public Box() {
        System.out.println("This is no-arg constructor");
    }

    Box(double w, double h, double d)
    {
        System.out.println("This is parameterized constructor");
        width = w;
        height = h;
        depth = d;
    }

    // compute and return volume
    double volume() {
        return width * height * depth;
    }

    void setDim(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }
}
```



```java
public class ConstructorDemo {

    public static void main(String args[]) {
        Box mybox = new Box();
        double vol;

        // set the dimensions of mybox
        mybox.setDim(10, 20, 15);

        // get volume of mybox
        vol = mybox.volume();
        System.out.println("Volume is " + vol);

        //using parameterized constructor
        mybox = new Box(12, 10, 14);
        vol = mybox.volume();
        System.out.println("Volume is " + vol);
    }
}
```



```
run:
This is no-arg constructor
Volume is 3000.0
This is parameterized constructor
Volume is 1680.0
```