Constructor

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

Constructor?

- **Constructor in java** is a *special type of method* that is used to initialize the object.
- Java constructor is *invoked at the time of object creation*. It constructs the values i.e. provides data for the object that is why it is known as constructor.

Rules for creating java constructor

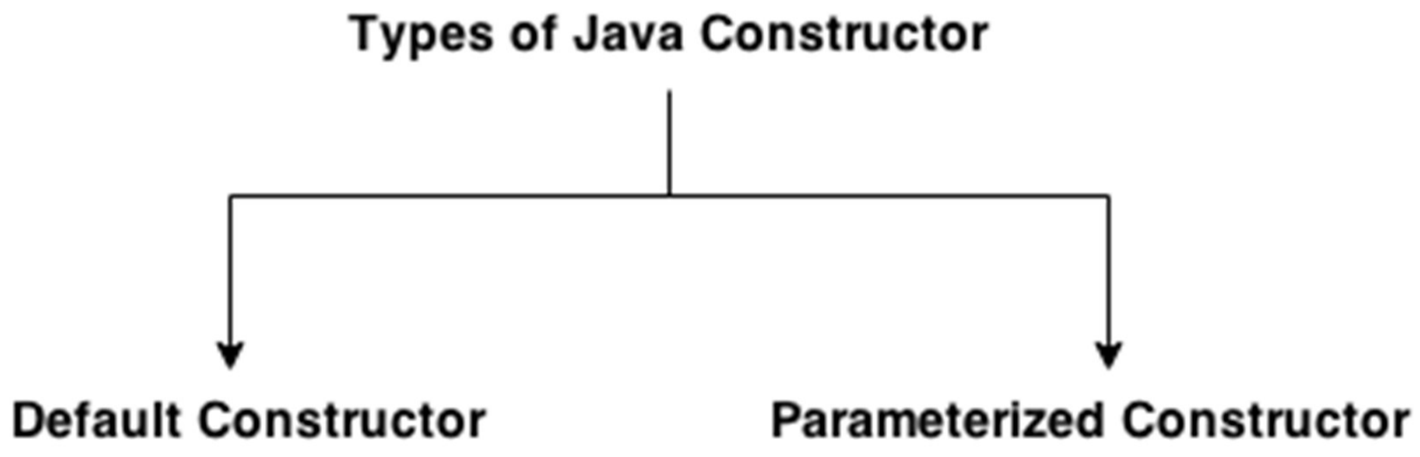
There are basically two rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type

Types of java constructors

There are two types of constructors:

- Default constructor (no-arg constructor)
- Parameterized constructor



Java Default Constructor

- A constructor that have no parameter is known as default constructor.

Syntax of default constructor:

```
<class_name>(){}
```

```
class Box {
  double width;
  double height;
  double depth;

  // This is the constructor for Box.
  Box() {
    System.out.println("Constructing Box");
    width = 10;
    height = 10;
    depth = 10;
  }

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

```java
class BoxDemo6 {
  public static void main(String args[]) {
    // declare, allocate, and initialize Box objects
    Box mybox1 = new Box();
    Box mybox2 = new Box();

    double vol;

    // get volume of first box
    vol = mybox1.volume();
    System.out.println("Volume is " + vol);

    // get volume of second box
    vol = mybox2.volume();
    System.out.println("Volume is " + vol);
  }
}
```

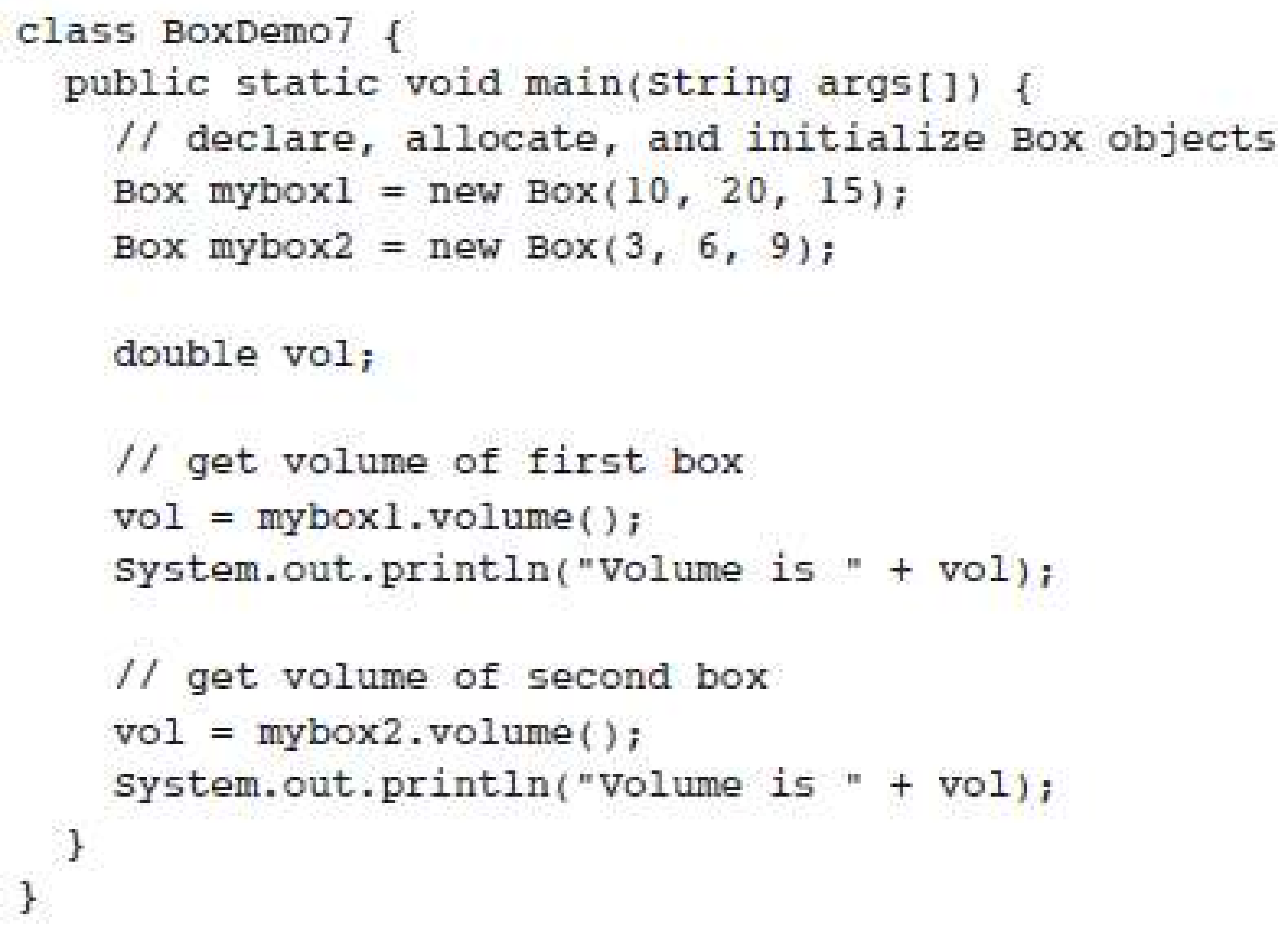
Java parameterized constructor

- A constructor that have parameters is known as parameterized constructor.
- Parameterized constructor is used to provide different values to the distinct objects.

Syntax of default constructor:

```
<class_name>(int a){}
```

```
/* Here, Box uses a parameterized constructor to
   initialize the dimensions of a box.
*/
class Box {
  double width;
  double height;
  double depth;

  // This is the constructor for Box.
  Box(double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
  }

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



```
class BoxDemo7 {
  public static void main(String args[]) {
    // declare, allocate, and initialize Box objects
    Box mybox1 = new Box(10, 20, 15);
    Box mybox2 = new Box(3, 6, 9);

    double vol;

    // get volume of first box
    vol = mybox1.volume();
    System.out.println("Volume is " + vol);

    // get volume of second box
    vol = mybox2.volume();
    System.out.println("Volume is " + vol);
  }
}
```

Java Copy Constructor

- There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.
- There are many ways to copy the values of one object into another in java. They are:
  - By constructor
  - By assigning the values of one object into another
  - By clone() method of Object class

Example

```
class Student6{
   int id;
   String name;
   Student6(int i,String n){
   id = i;
   name = n;
   }

   Student6(Student6 s){
   id = s.id;
   name =s.name;
   }
 void display(){System.out.println
(id+" "+name);}

public static void main(String ar
gs[]){
   Student6 s1 = new Student6(11
1,"Karan");
   Student6 s2 = new Student6(s1)
;
   s1.display();
   s2.display();
  }
}
```

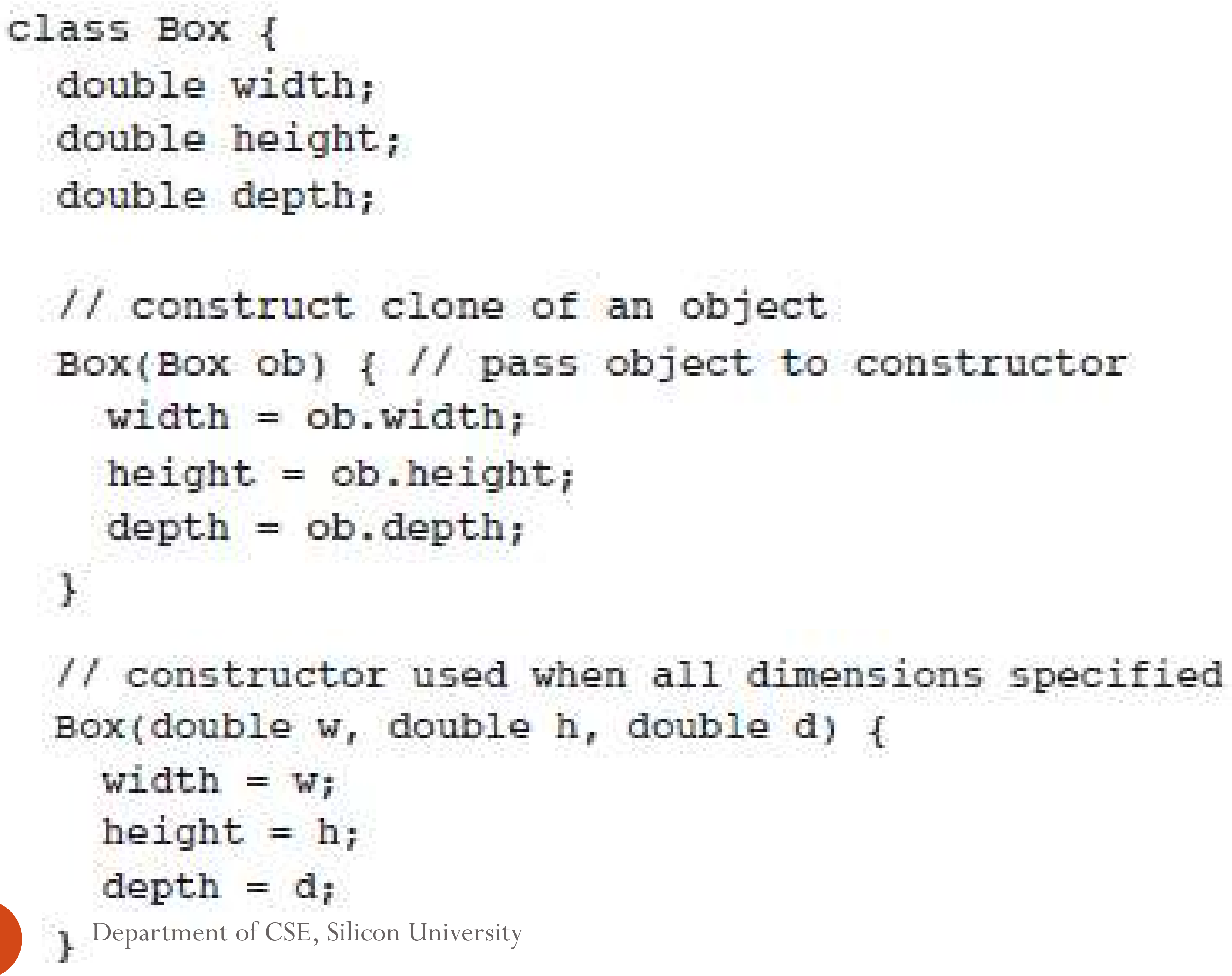
Constructor Overloading

- Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.
- The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

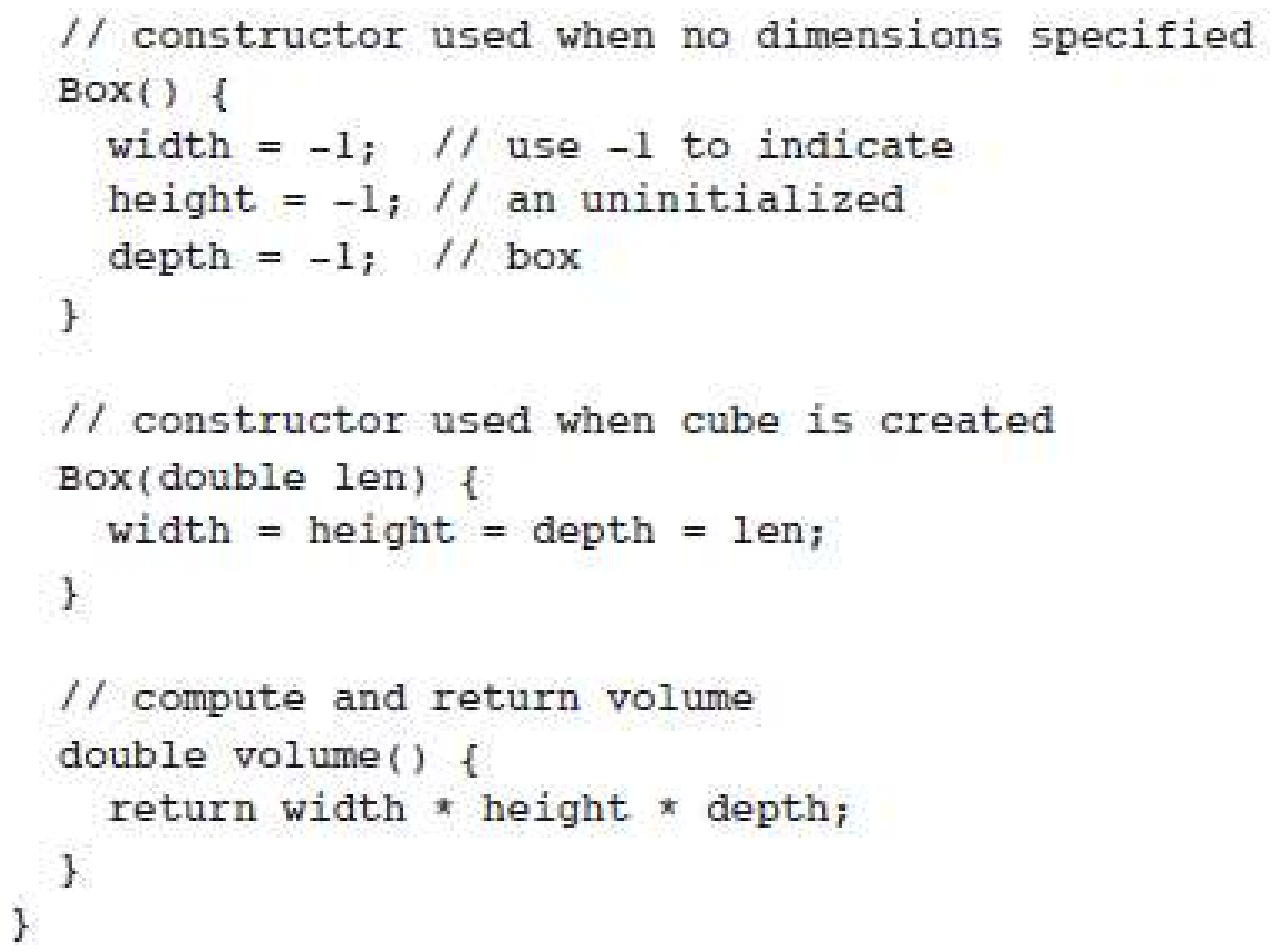
Example

```
class Student5{
  int id;
  String name;
  int age;
  Student5(int i,String n){
  id = i;
  name = n;
  }
  Student5(int i,String n,int a){
  id = i;
  name = n;
  age=a;
  }
void display(){System.out.println
(id+" "+name+" "+age);}

public static void main(String ar
gs[]){
   Student5 s1 = new Student5(11
1,"Karan");
   Student5 s2 = new Student5(22
2,"Aryan",25);
   s1.display();
   s2.display();
  }
}
```

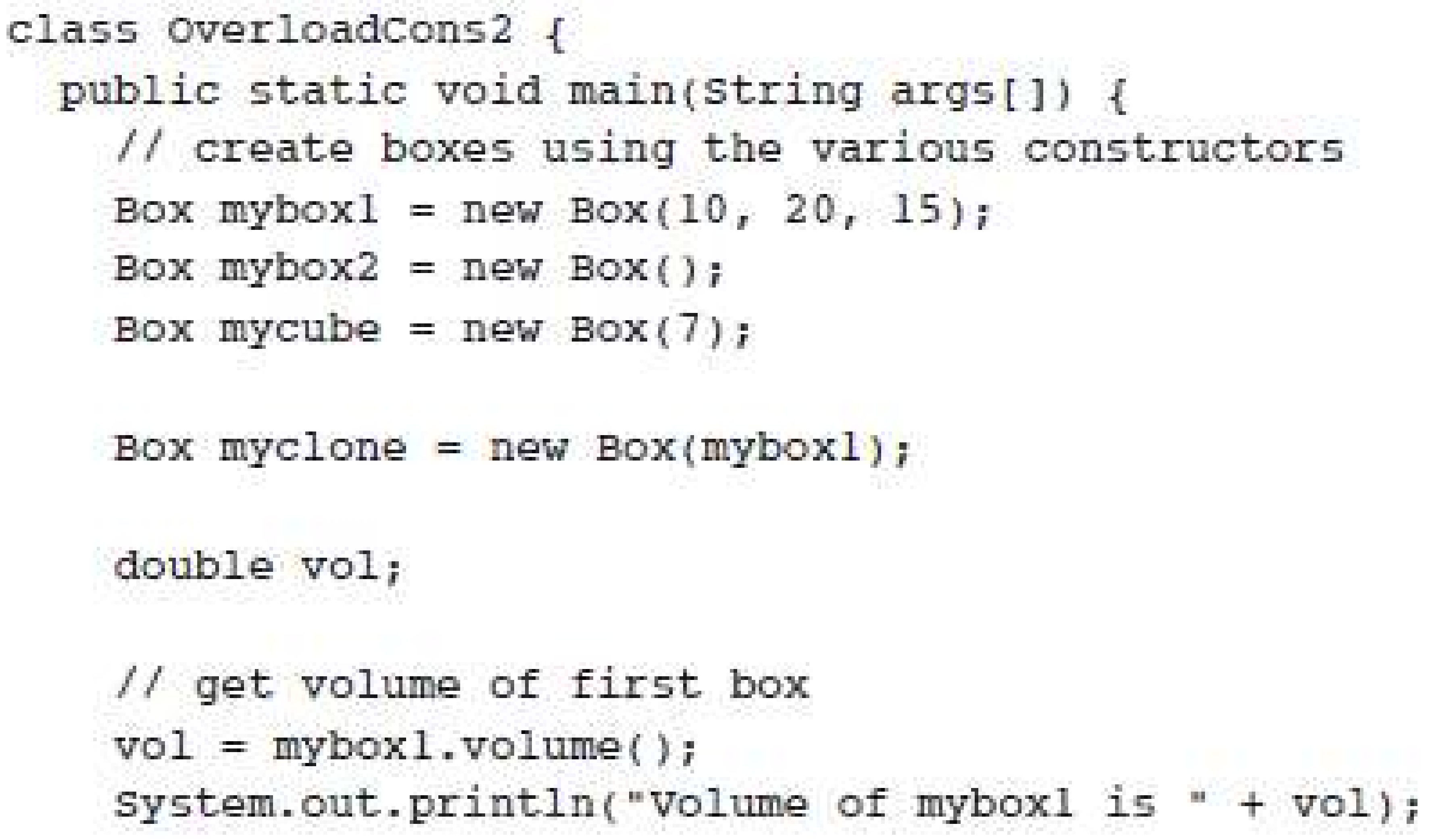
Example



```
class Box {
  double width;
  double height;
  double depth;

  // construct clone of an object
  Box(Box ob) { // pass object to constructor
    width = ob.width;
    height = ob.height;
    depth = ob.depth;
  }

  // constructor used when all dimensions specified
  Box(double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
  }
```



```
  // constructor used when no dimensions specified
  Box() {
    width = -1;  // use -1 to indicate
    height = -1; // an uninitialized
    depth = -1;  // box
  }

  // constructor used when cube is created
  Box(double len) {
    width = height = depth = len;
  }

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



```
class OverloadCons2 {
  public static void main(String args[]) {
    // create boxes using the various constructors
    Box mybox1 = new Box(10, 20, 15);
    Box mybox2 = new Box();
    Box mycube = new Box(7);

    Box myclone = new Box(mybox1);

    double vol;

    // get volume of first box
    vol = mybox1.volume();
    System.out.println("Volume of mybox1 is " + vol);
```

```
        // get volume of second box
        vol = mybox2.volume();
        System.out.println("Volume of mybox2 is " + vol);

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

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

Java Constructor vs Java Method

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behaviour of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name.

Thank you