Object Oriented Programming

Instructor: Asad Ullah Khan

Introduction to Classes

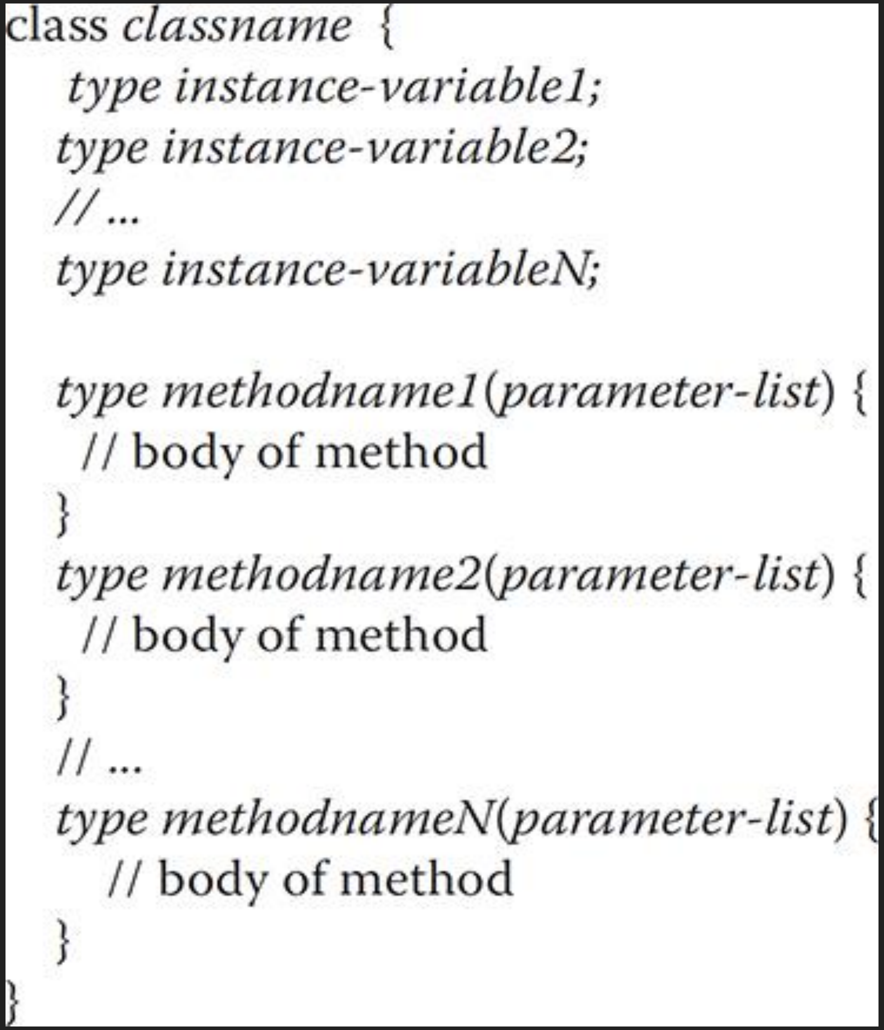
- The class is at the core of Java.
- It is the logical construct upon which the entire Java language is built because it defines the shape and nature of an object.
- As such, the class forms the basis for object-oriented programming in Java.
- Any concept you wish to implement in a Java program must be encapsulated within a class.

Introduction to Classes

- Most important thing to understand about a class is that it defines a new data type.
- This new type can be used to create objects of that type.
- Thus, a class is a template for an object, and an object is an instance of a class.
- Two words object and instance will be used interchangeably.
- When we define a class we are specifying the data that it contains and the code (Methods) that operates on that data.

Introduction to Classes

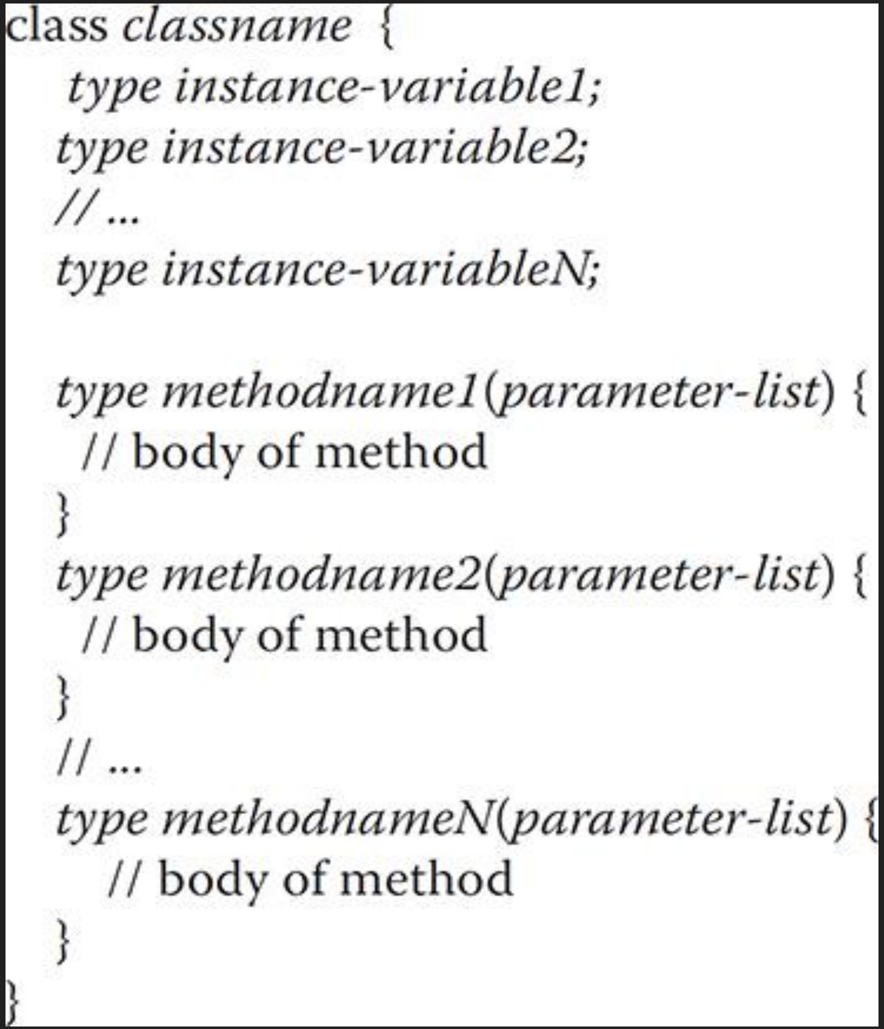
- A class is declared by use of the ***class*** keyword.
- The data, or variables, defined within a class are called ***instance variables.***
- The code is contained within ***methods***.
- The methods and variables defined within a class are called ***members*** of the class.
- A simplified general form of a class definition is shown



```
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
   }
}
```

Introduction to Classes

- Variables defined within a class are called instance variables because each instance of the class (that is, each object of the class) contains its own copy of these variables.
- Thus, the data for one object is separate and unique from the data for another.

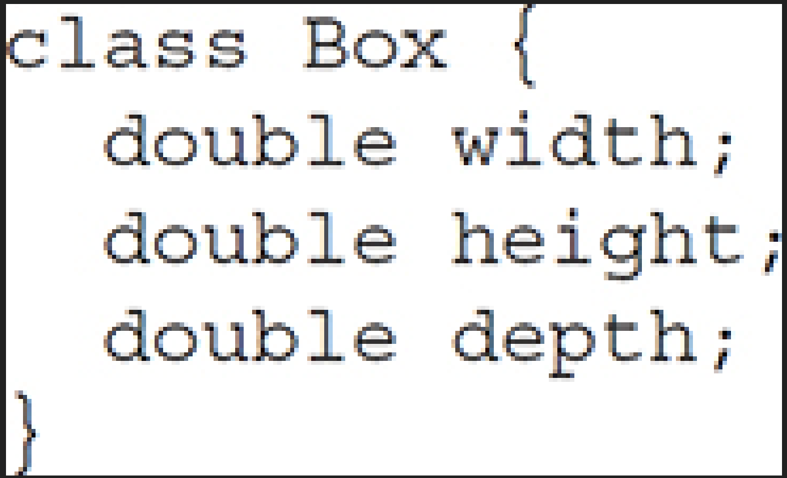


```
class classname {
    type instance-variable1;
   type instance-variable2;
   // ...
   type instance-variableN;

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

A Simple Class

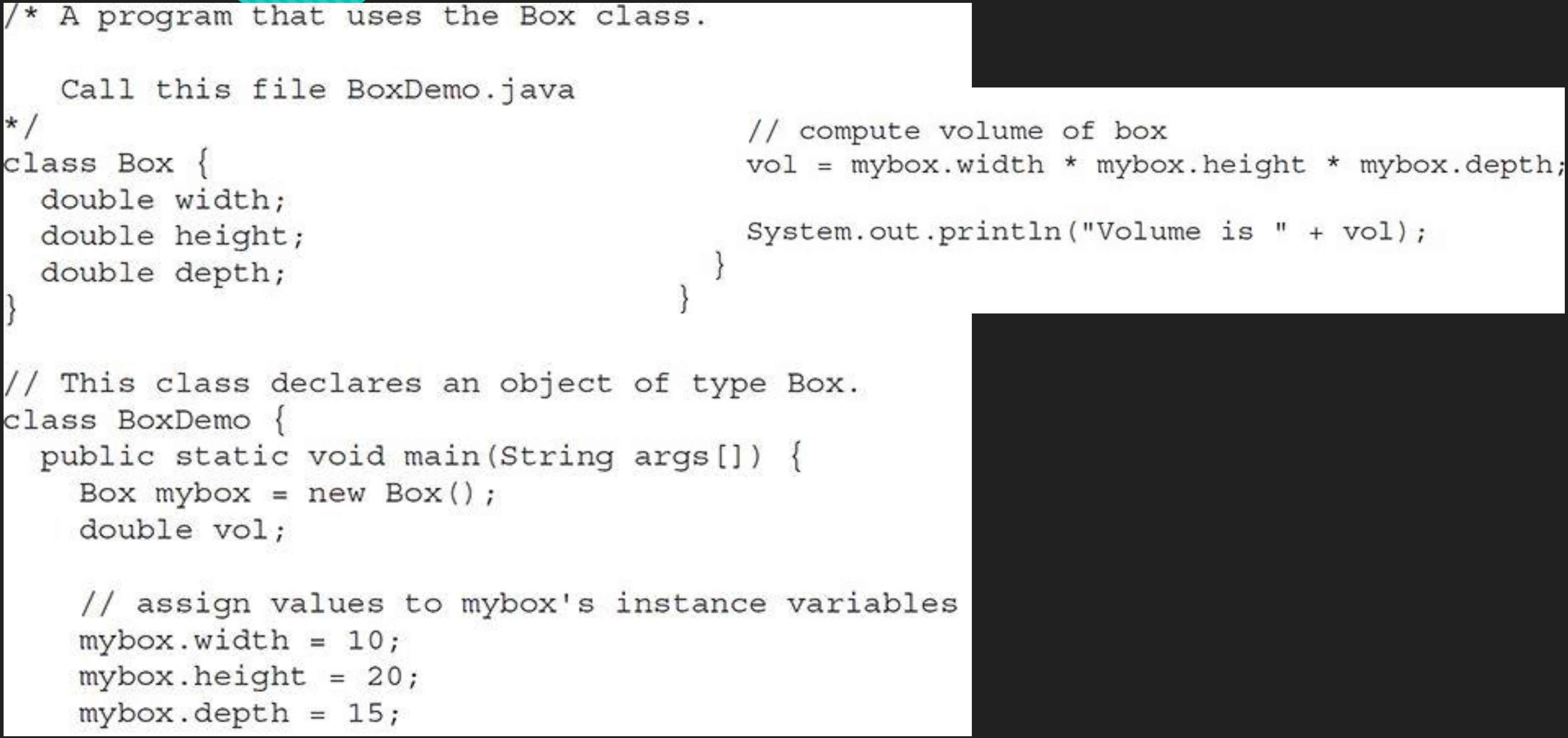
- Lets begin with a simple class, called **Box**.
- As stated, a class defines a new type of data.
- In this case, the new data type is called **Box**.
- You will use this name to declare objects of type **Box**.
- To create a Box object, you will use a statement like the following:

```
Box mybox = new Box(); //Box object called mybox
```



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

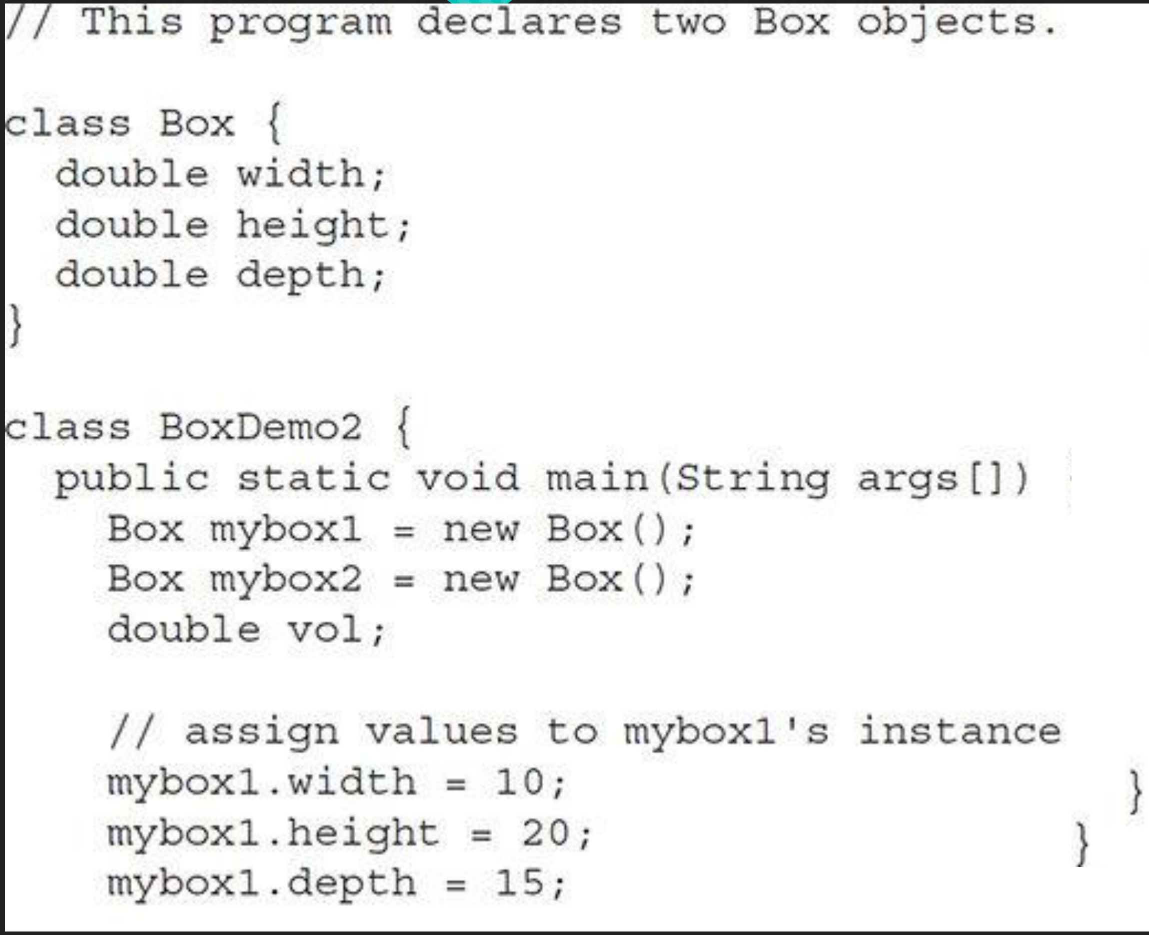
A Simple Class



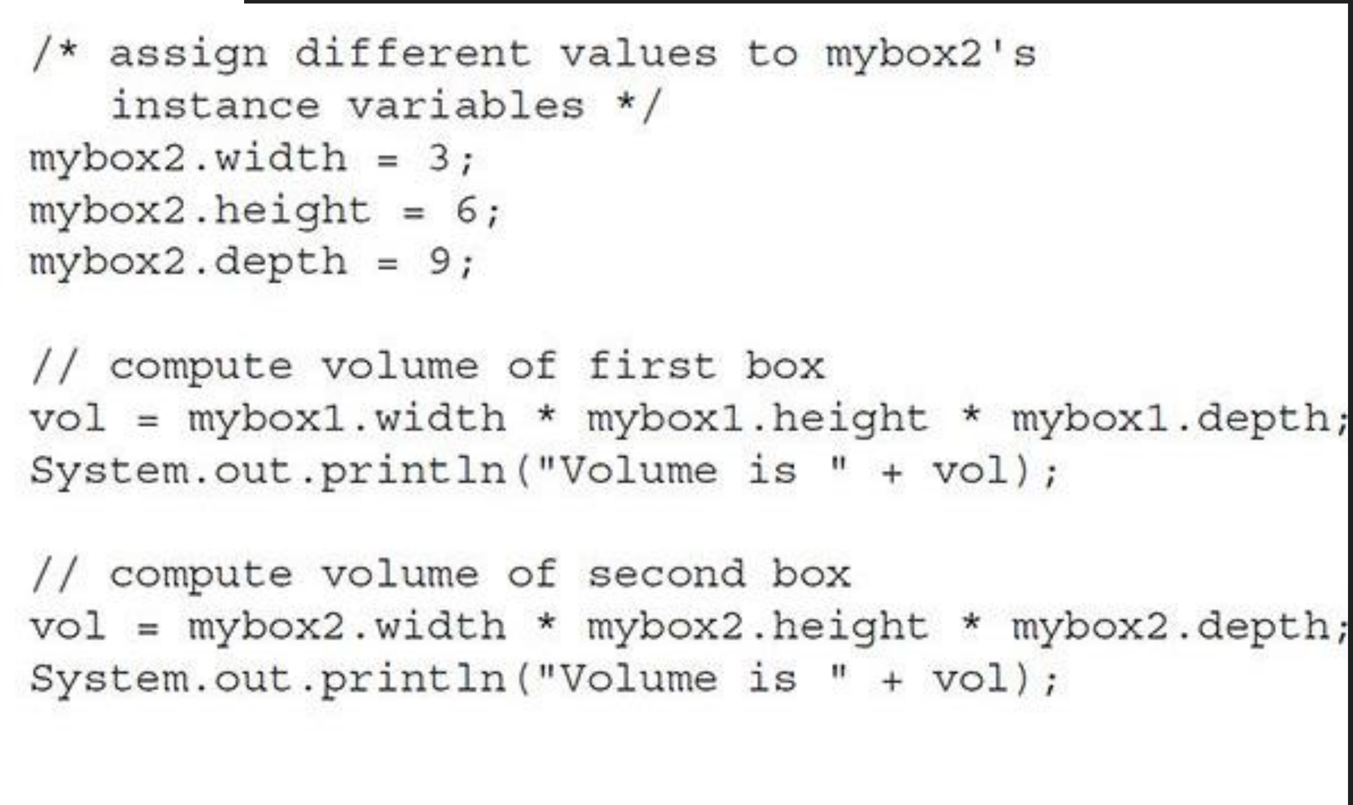
```
/* A program that uses the Box class.

   Call this file BoxDemo.java
*/
class Box {
  double width;
  double height;
  double depth;
}

// This class declares an object of type Box.
class BoxDemo {
  public static void main(String args[]) {
    Box mybox = new Box();
    double vol;

    // assign values to mybox's instance variables
    mybox.width = 10;
    mybox.height = 20;
    mybox.depth = 15;
```

```
    // compute volume of box
    vol = mybox.width * mybox.height * mybox.depth;

    System.out.println("Volume is " + vol);
  }
}
```

Another Simple Class



```
// This program declares two Box objects.

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

class BoxDemo2 {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
    double vol;

    // assign values to mybox1's instance
    mybox1.width = 10;
    mybox1.height = 20;
    mybox1.depth = 15;
```



```
    /* assign different values to mybox2's
       instance variables */
    mybox2.width = 3;
    mybox2.height = 6;
    mybox2.depth = 9;

    // compute volume of first box
    vol = mybox1.width * mybox1.height * mybox1.depth;
    System.out.println("Volume is " + vol);

    // compute volume of second box
    vol = mybox2.width * mybox2.height * mybox2.depth;
    System.out.println("Volume is " + vol);
  }
}
```

Declaring Objects

- Obtaining **objects** of a class is a two-step process.
  1. You must declare a variable of the class type.
  2. You must acquire an actual, physical copy of the object and assign it to that variable.
- You can do this using the **new** operator.
- The **new** operator dynamically allocates memory for an object and returns a reference to it.
- This reference is, essentially, the address in memory of the object allocated by **new**.
- This reference is then stored in the variable.

Declaring Objects

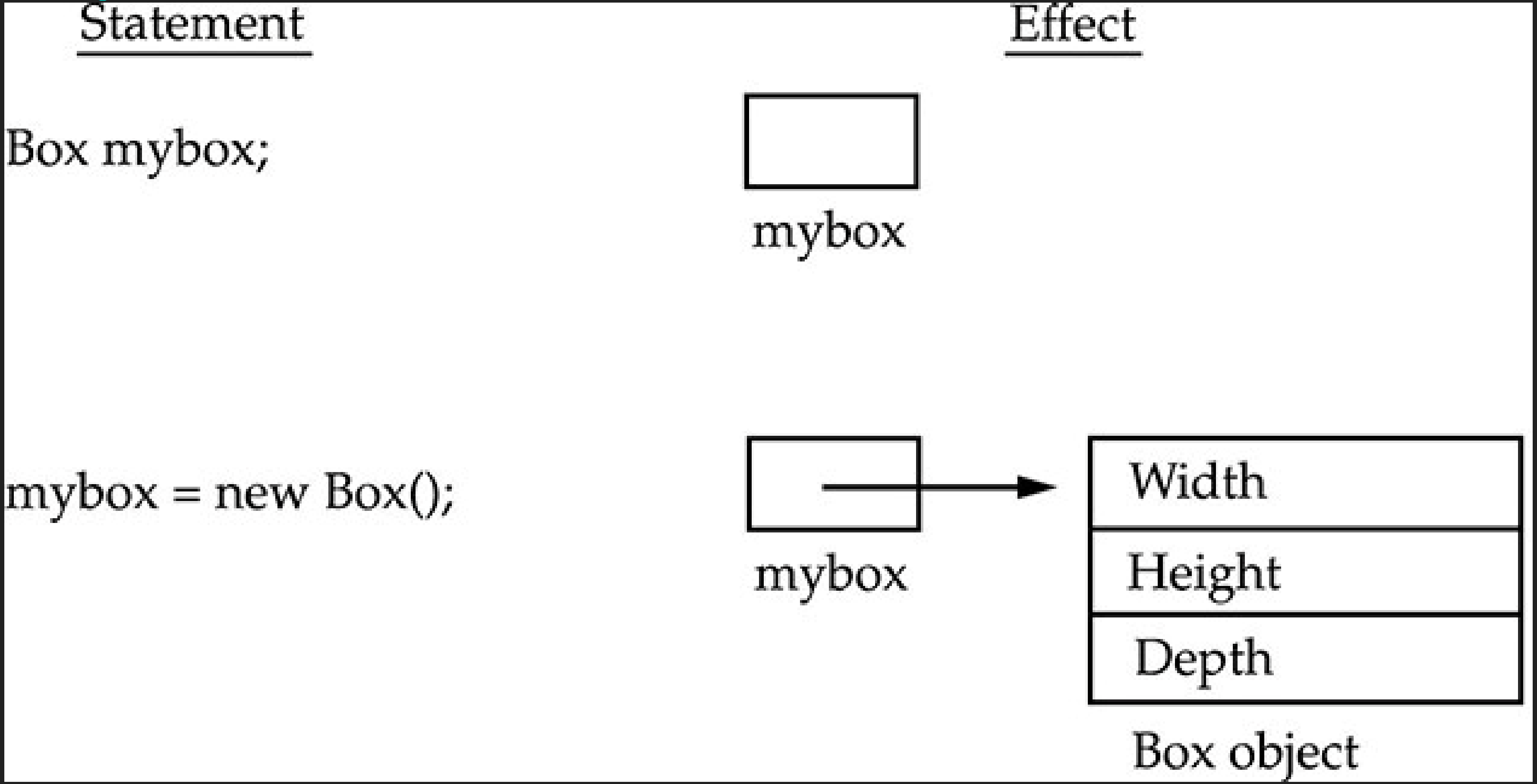
- In the preceding sample programs, a line similar to the following is used to declare an object of type Box:

```
Box mybox = new Box();
```

- This statement combines the two steps just described.
- It can be rewritten like this to show each step more clearly:

```
Box mybox; // declare reference to object
mybox = new Box(); // allocate a Box object
```

Declaring Objects

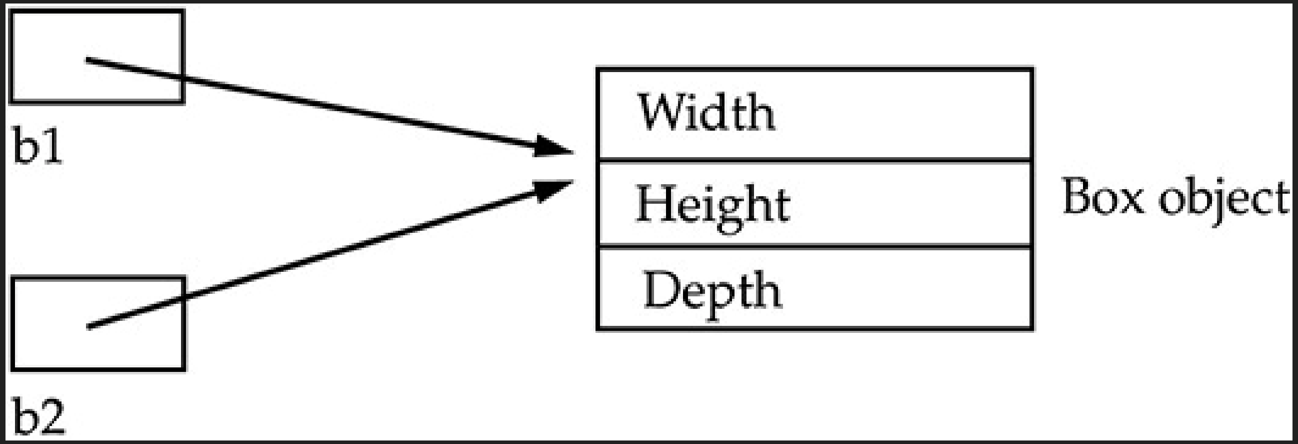


```
Box mybox;
```

```
mybox = new Box();
```

Output of statement?

```
Box b1 = new Box();
Box b2 = b1;
```

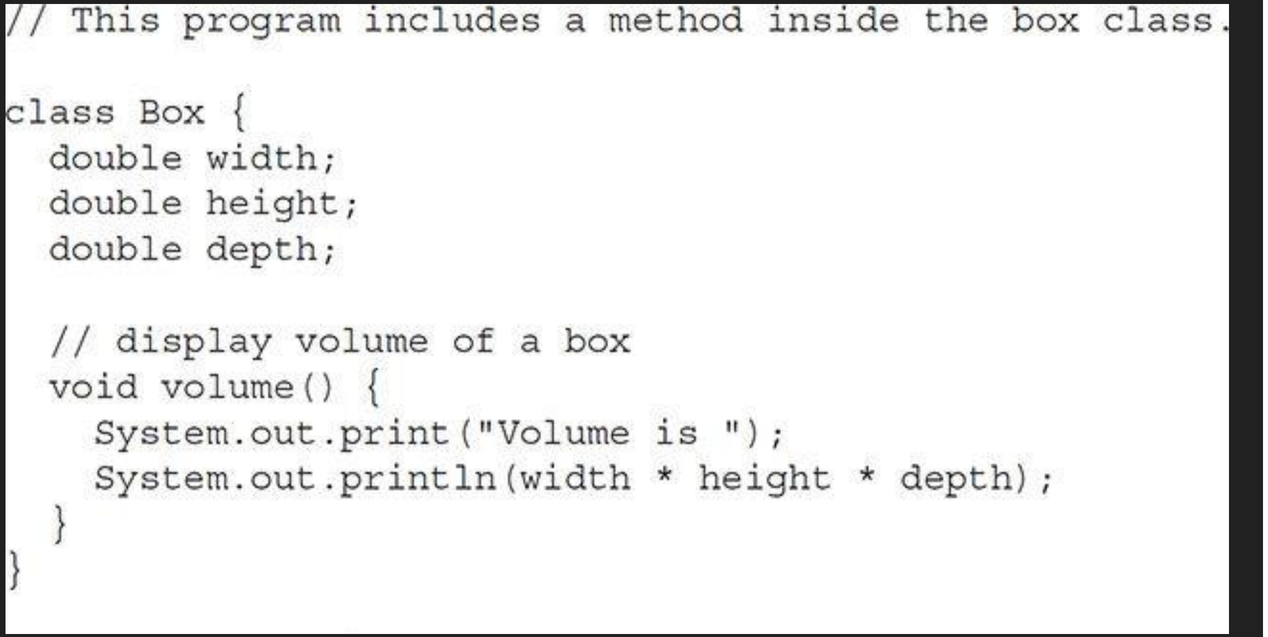


Introducing Methods

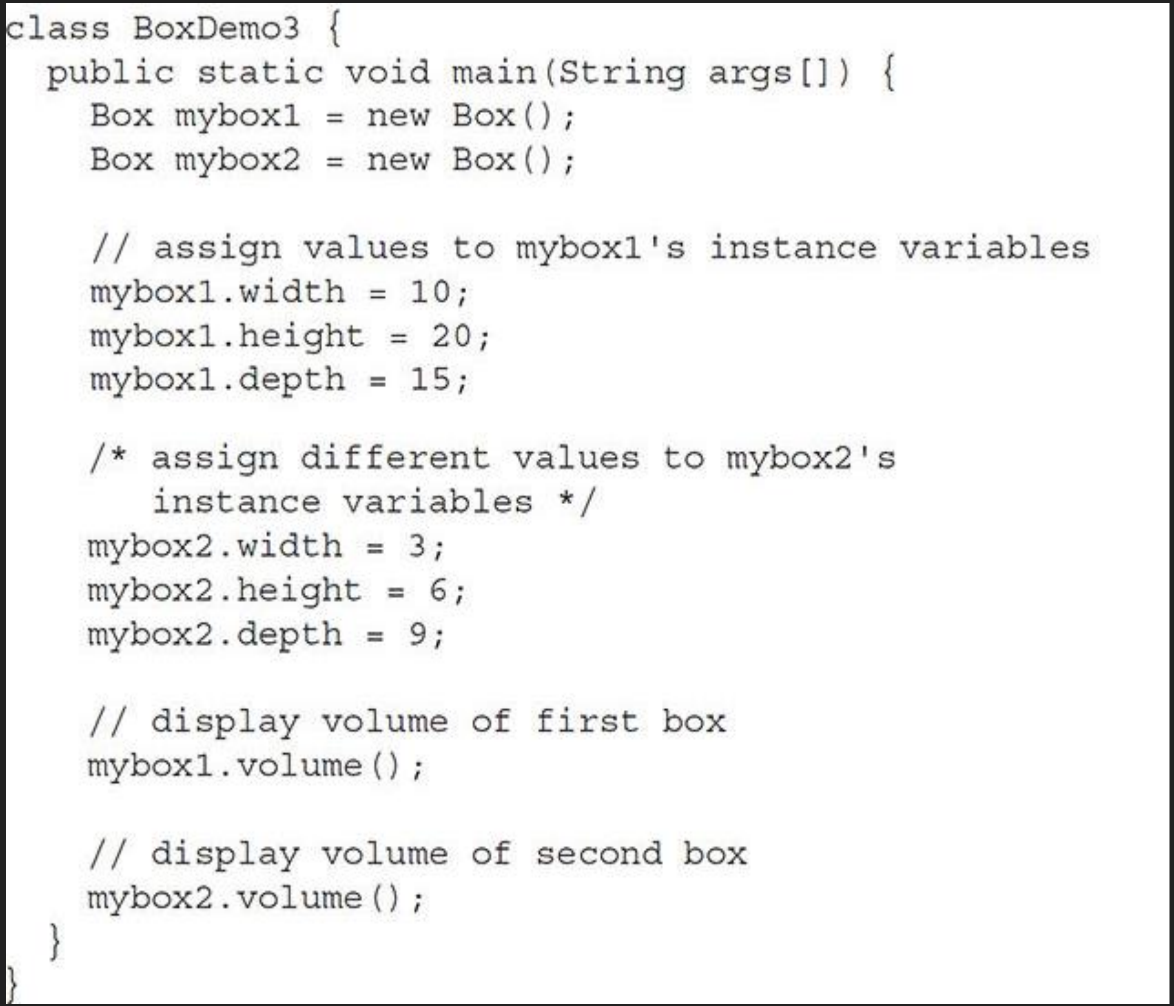
- Classes usually consist of two things:
    - instance variables, and
    - methods.
- This is the general form of a method:

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

Adding a Method to the Box Class

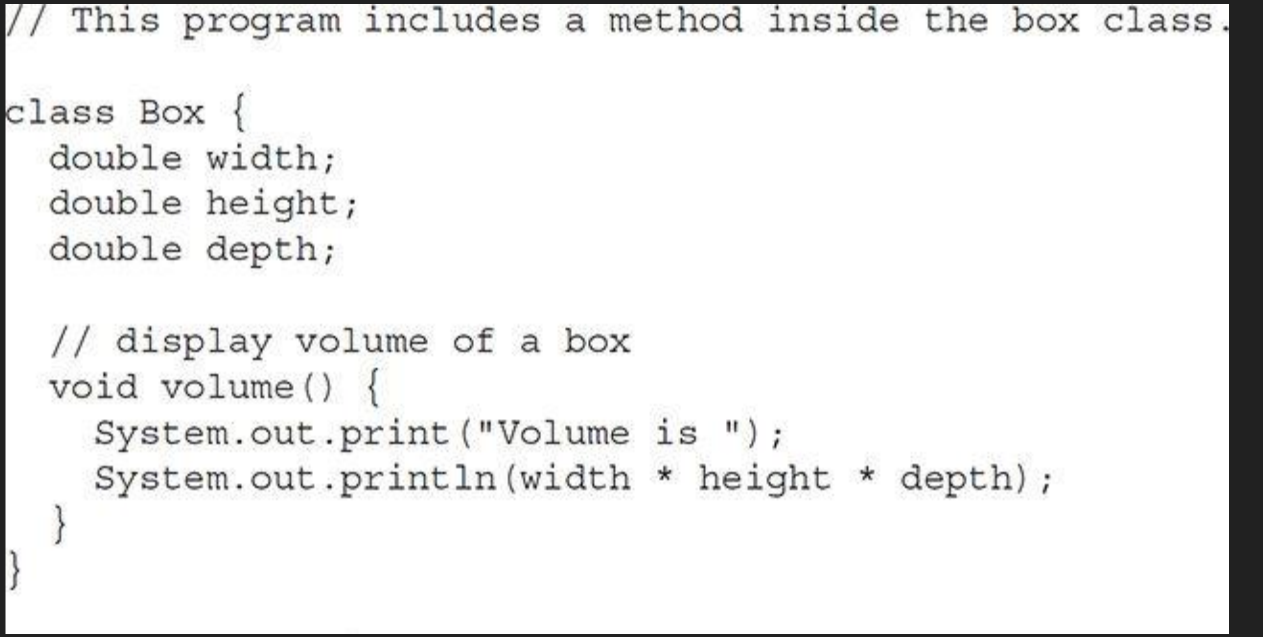


```
// This program includes a method inside the box class.

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

  // display volume of a box
  void volume() {
    System.out.print("Volume is ");
    System.out.println(width * height * depth);
  }
}
```

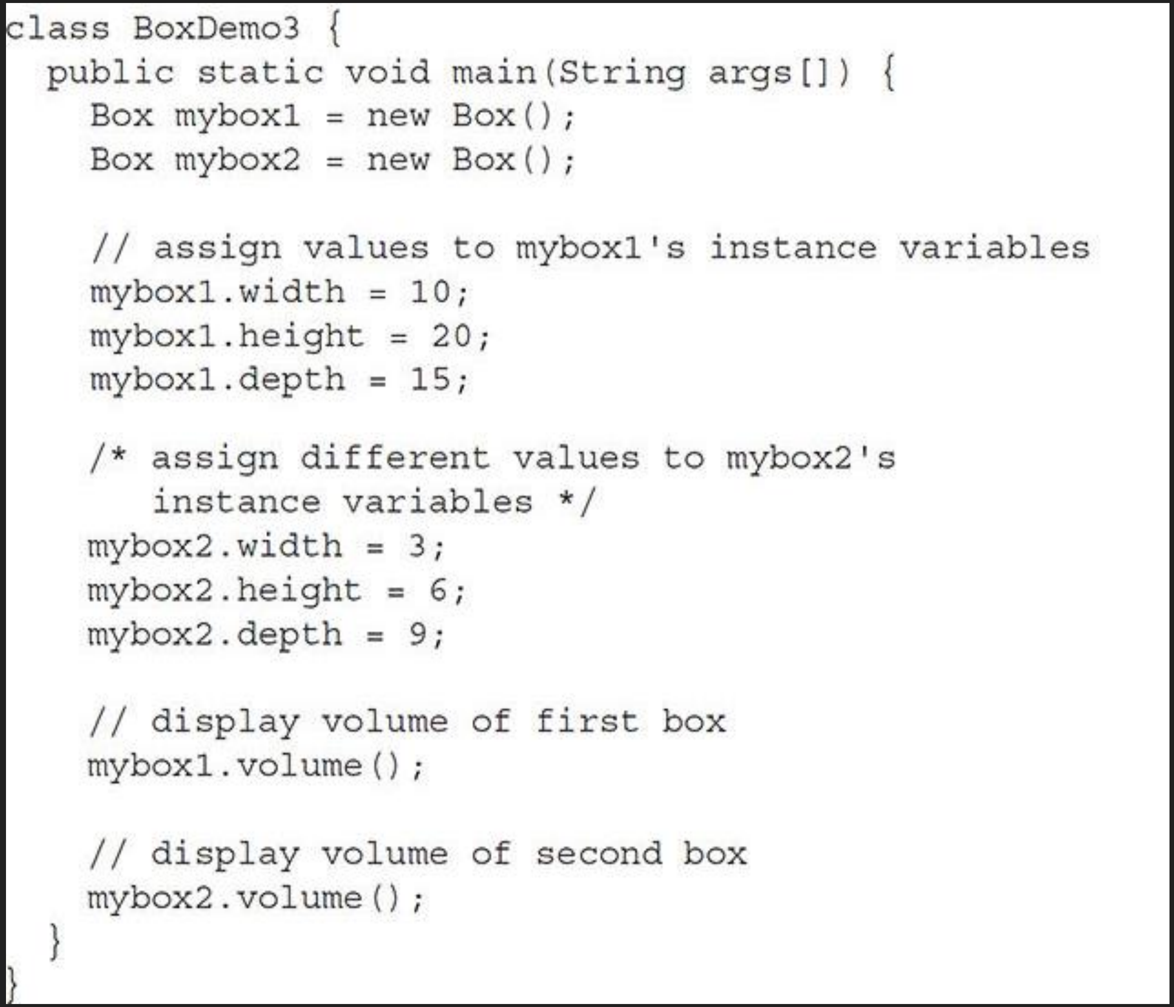


```
class BoxDemo3 {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();

    // assign values to mybox1's instance variables
    mybox1.width = 10;
    mybox1.height = 20;
    mybox1.depth = 15;

    /* assign different values to mybox2's
       instance variables */
    mybox2.width = 3;
    mybox2.height = 6;
    mybox2.depth = 9;

    // display volume of first box
    mybox1.volume();

    // display volume of second box
    mybox2.volume();
  }
}
```

Adding a Method to the Box Class

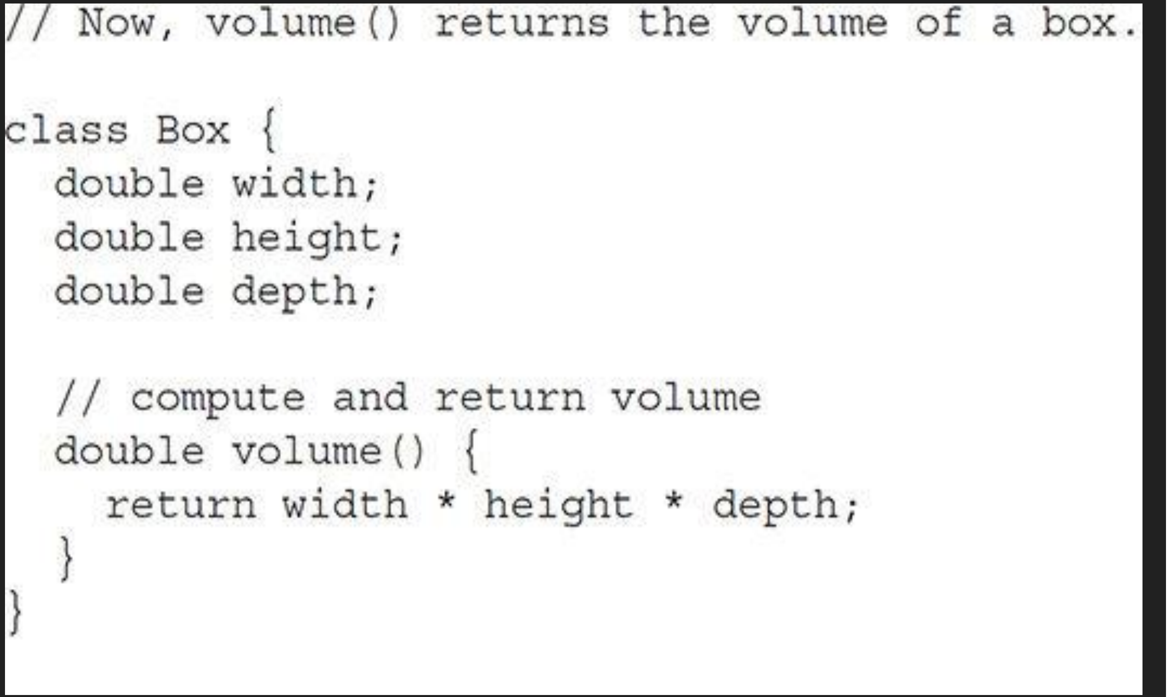


```
// This program includes a method inside the box class.

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

  // display volume of a box
  void volume() {
    System.out.print("Volume is ");
    System.out.println(width * height * depth);
  }
}
```

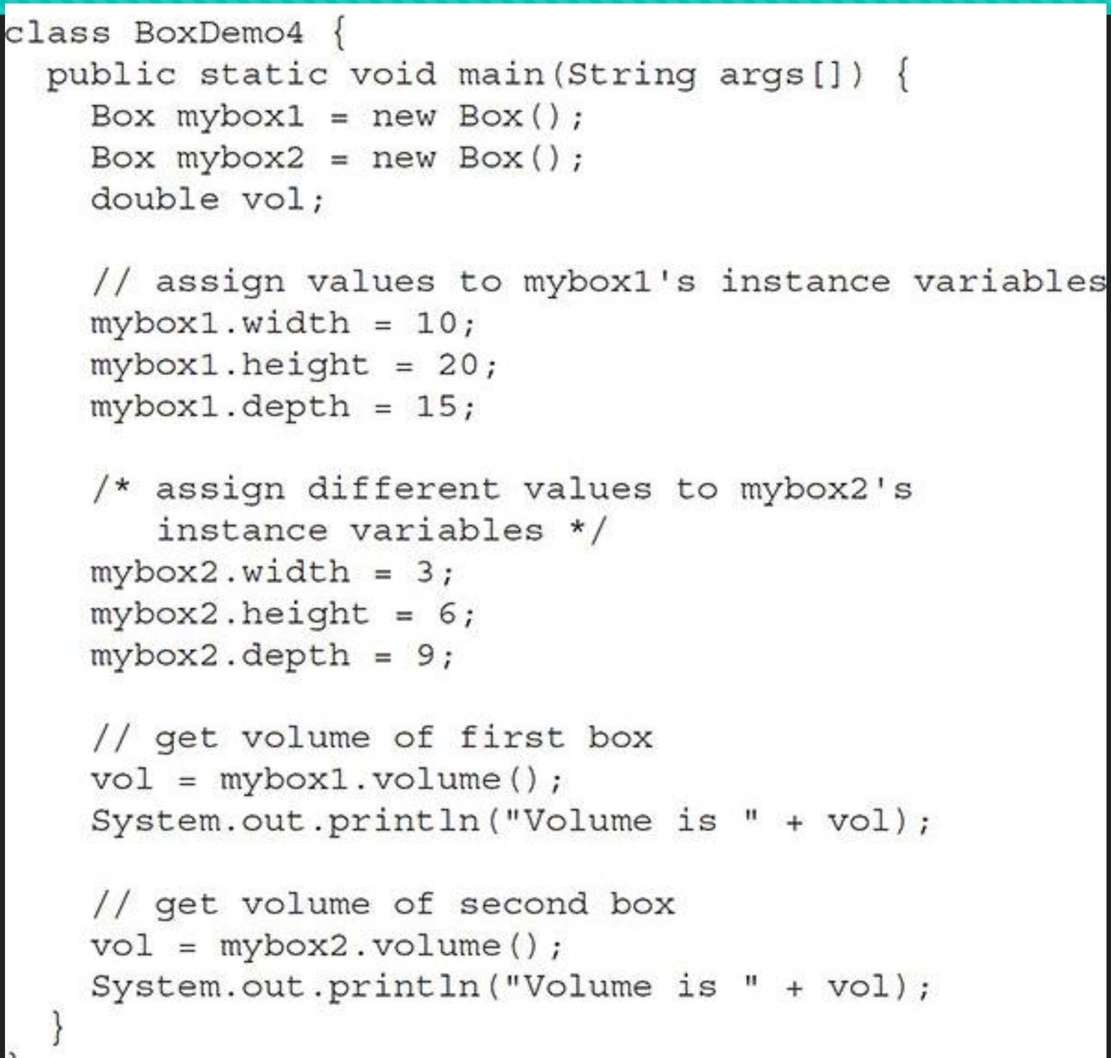


```
class BoxDemo3 {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();

    // assign values to mybox1's instance variables
    mybox1.width = 10;
    mybox1.height = 20;
    mybox1.depth = 15;

    /* assign different values to mybox2's
       instance variables */
    mybox2.width = 3;
    mybox2.height = 6;
    mybox2.depth = 9;

    // display volume of first box
    mybox1.volume();

    // display volume of second box
    mybox2.volume();
  }
}
```

Method with return type

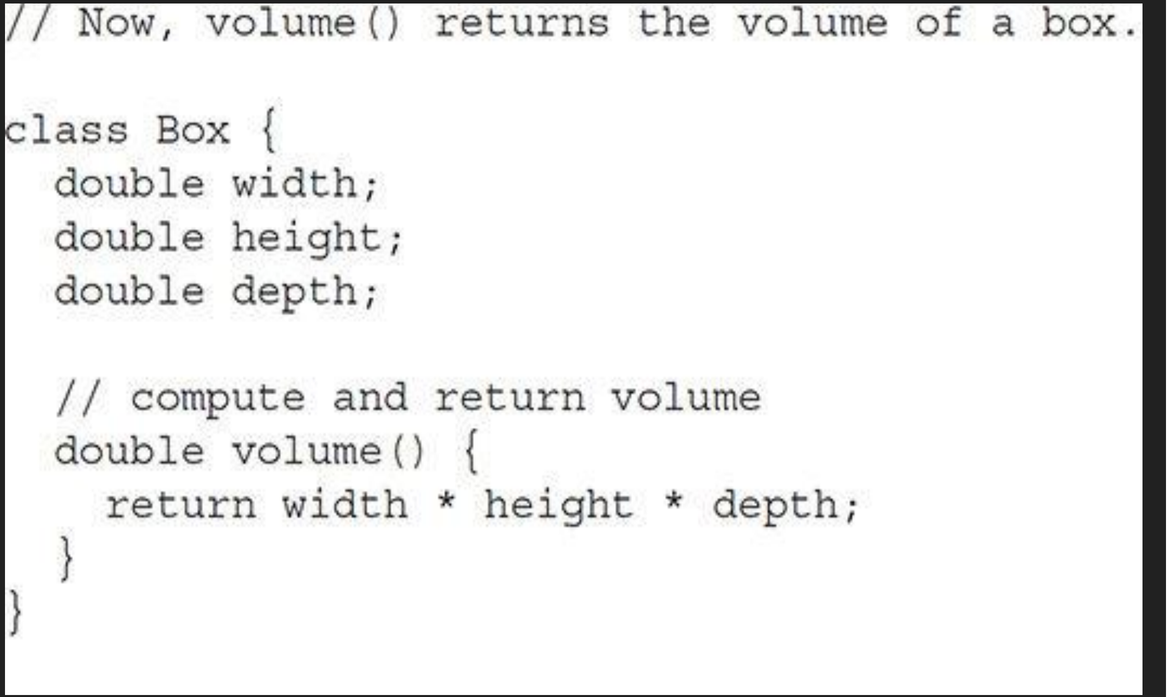


```
// Now, volume() returns the volume of a box.

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

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

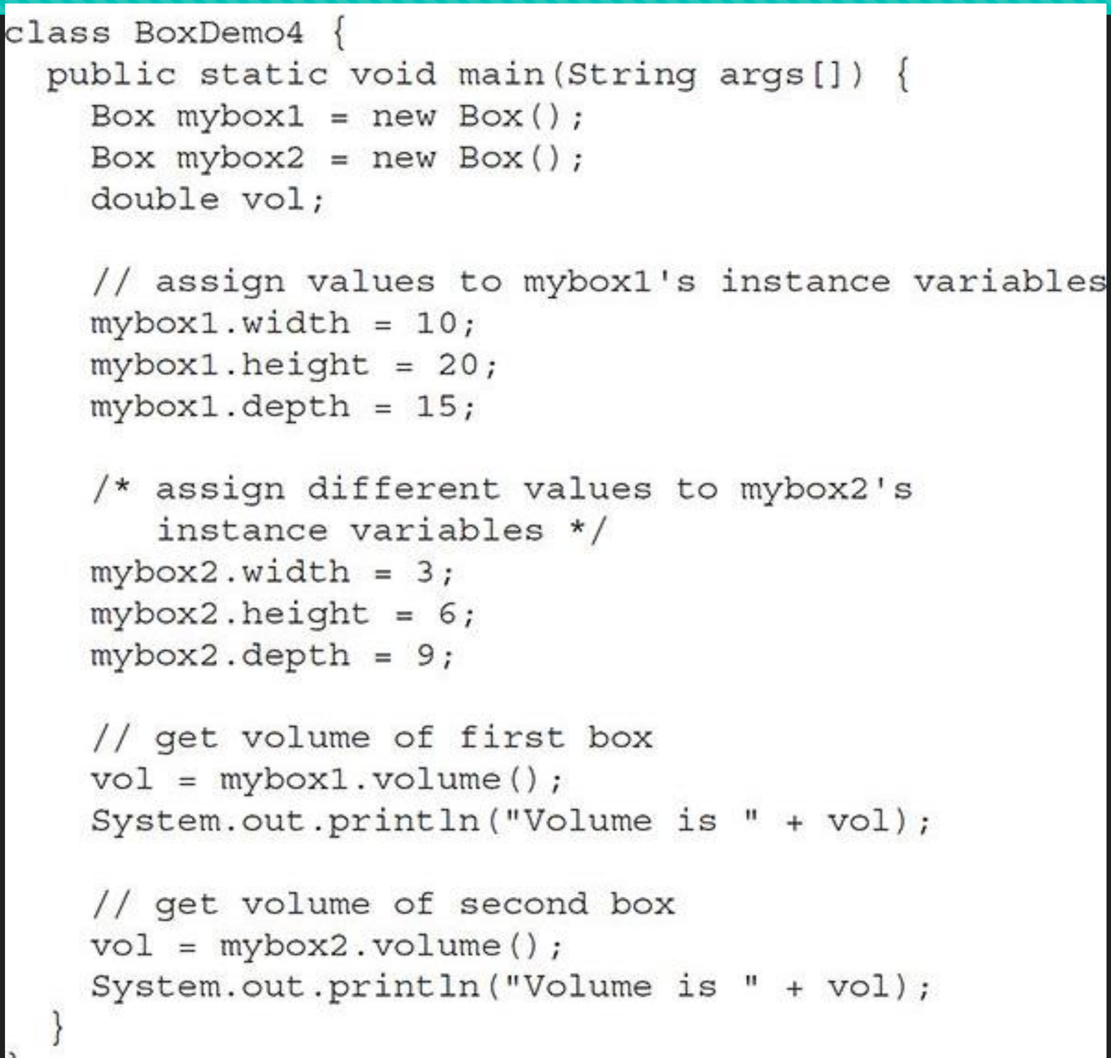


```
class BoxDemo4 {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
    double vol;

    // assign values to mybox1's instance variables
    mybox1.width = 10;
    mybox1.height = 20;
    mybox1.depth = 15;

    /* assign different values to mybox2's
       instance variables */
    mybox2.width = 3;
    mybox2.height = 6;
    mybox2.depth = 9;

    // 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);
  }
```

Write Method that Take Parameters



```
// Now, volume() returns the volume of a box.

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

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

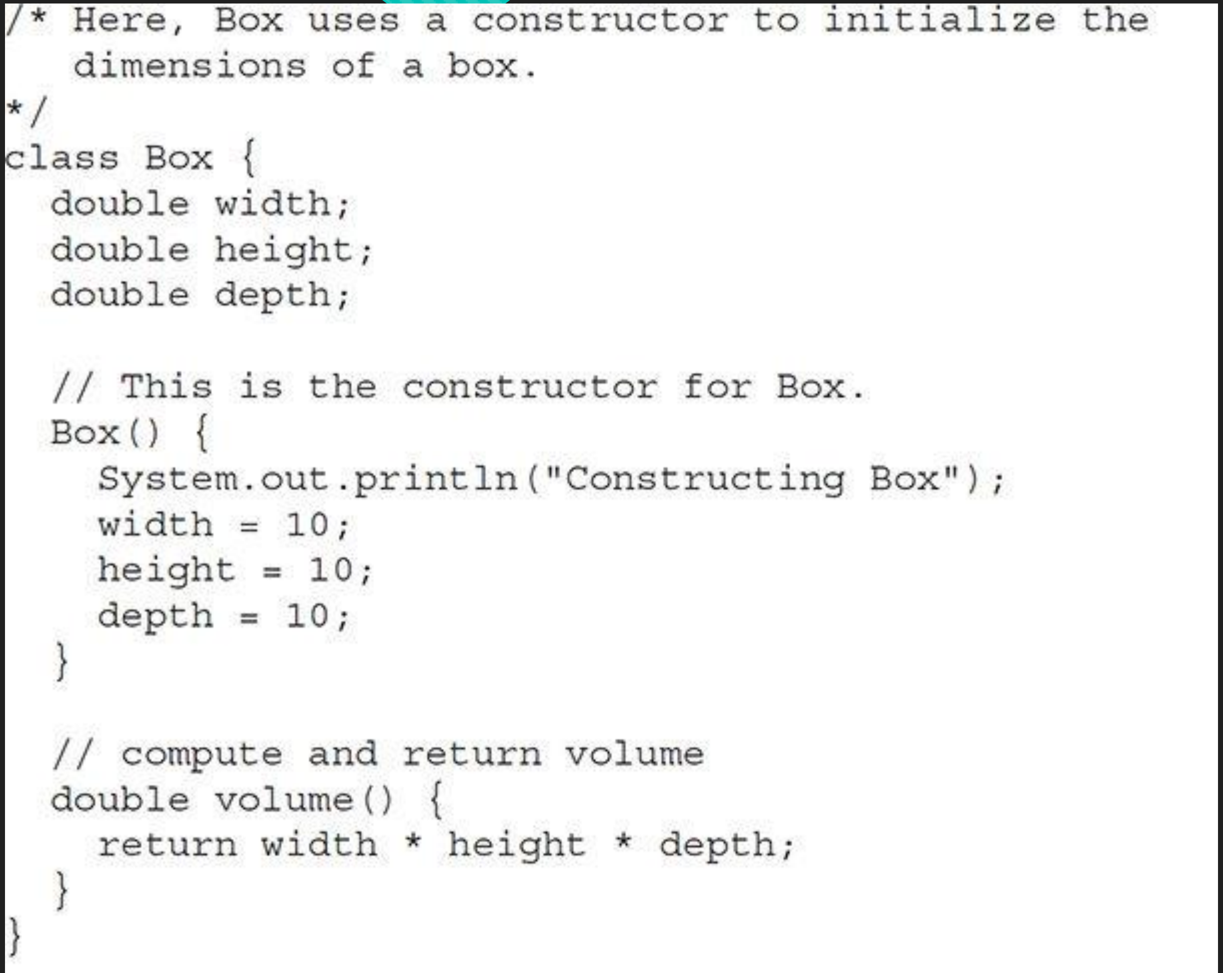


```
class BoxDemo4 {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
    double vol;

    // assign values to mybox1's instance variables
    mybox1.width = 10;
    mybox1.height = 20;
    mybox1.depth = 15;

    /* assign different values to mybox2's
       instance variables */
    mybox2.width = 3;
    mybox2.height = 6;
    mybox2.depth = 9;

    // 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);
  }
```

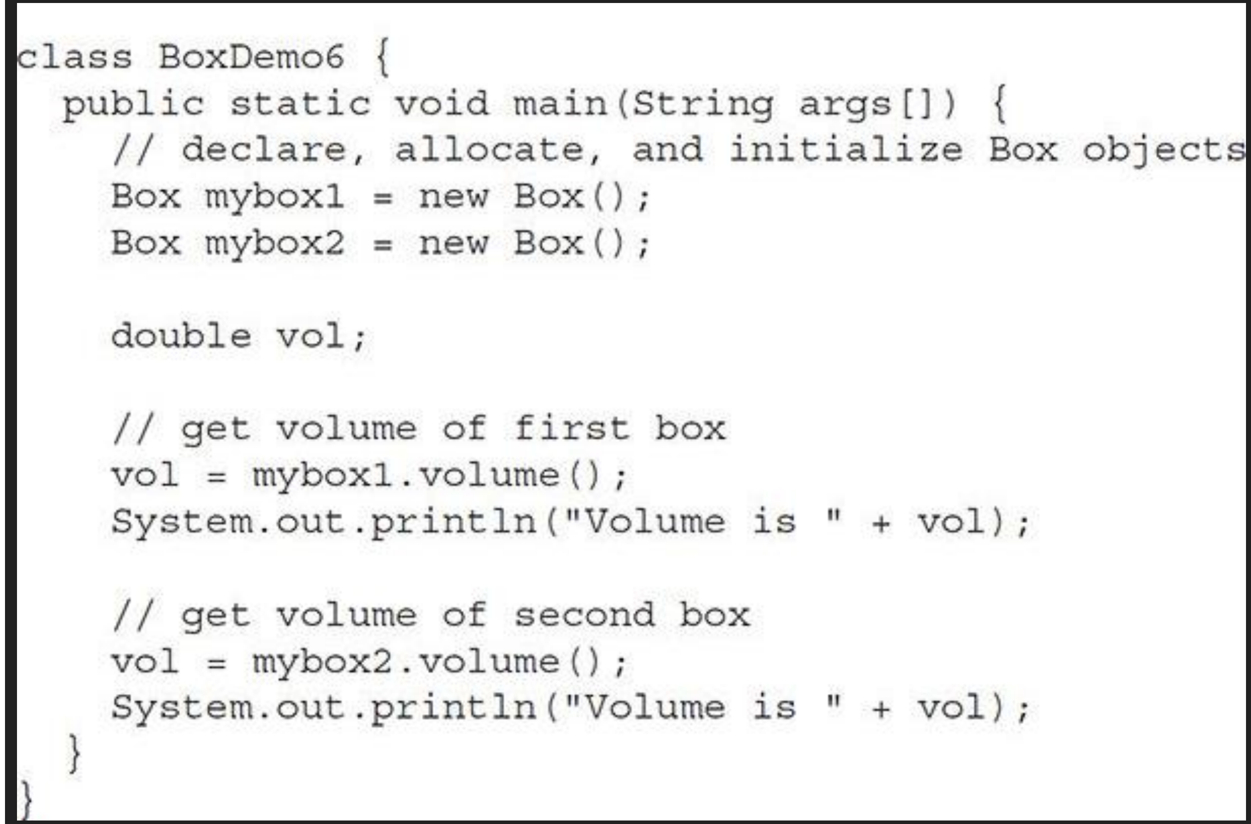
Constructors

- Java allows objects to initialize themselves when they are created.
- This automatic initialization is performed through the use of a constructor.
- A constructor initializes an object immediately upon creation.
- It has the same name as the class in which it resides and is syntactically similar to a method.
- Once defined, the constructor is automatically called when the object is created, before the new operator completes.
- Constructors look a little strange because they have no return type, not even void.

Constructors

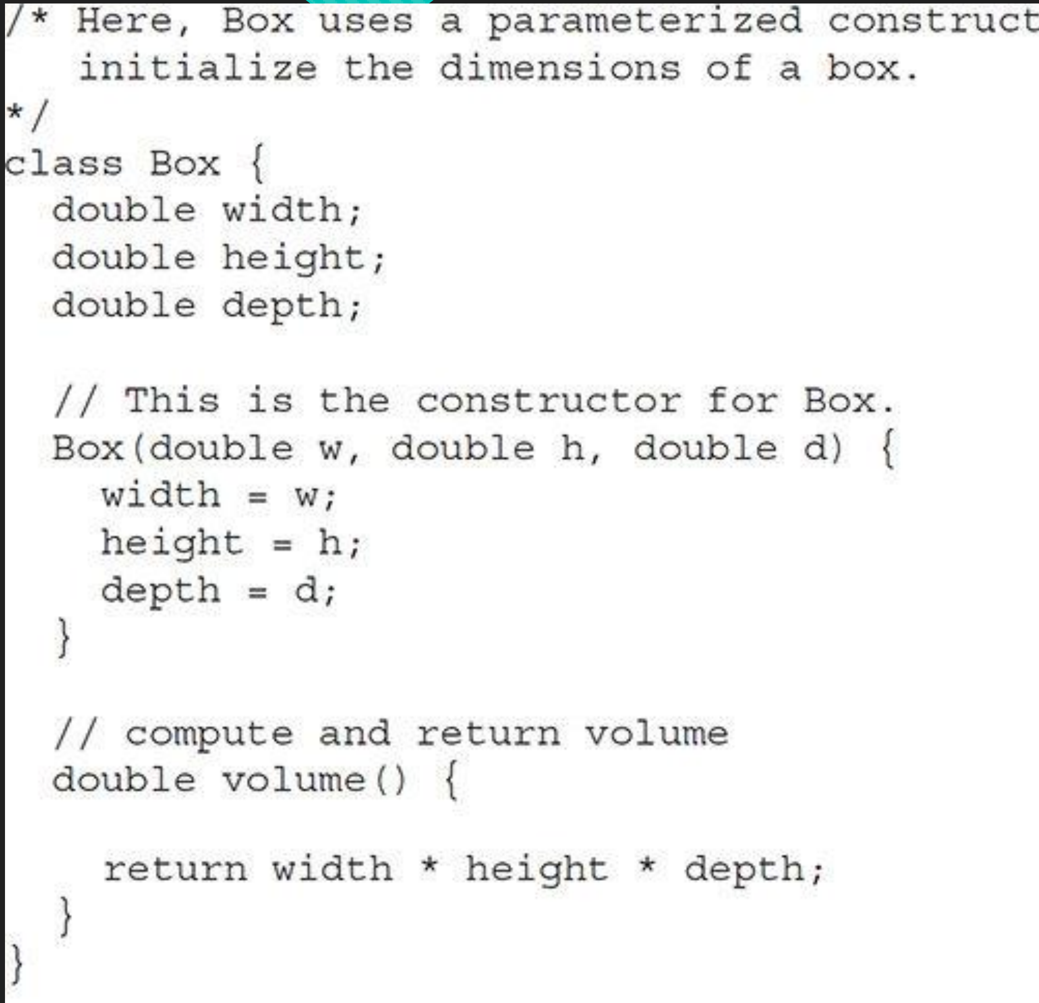


```
/* Here, Box uses a constructor to initialize the
   dimensions of a box.
*/
class Box {
  double width;
  double height;
  double depth;

  // 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;
  }
}
```

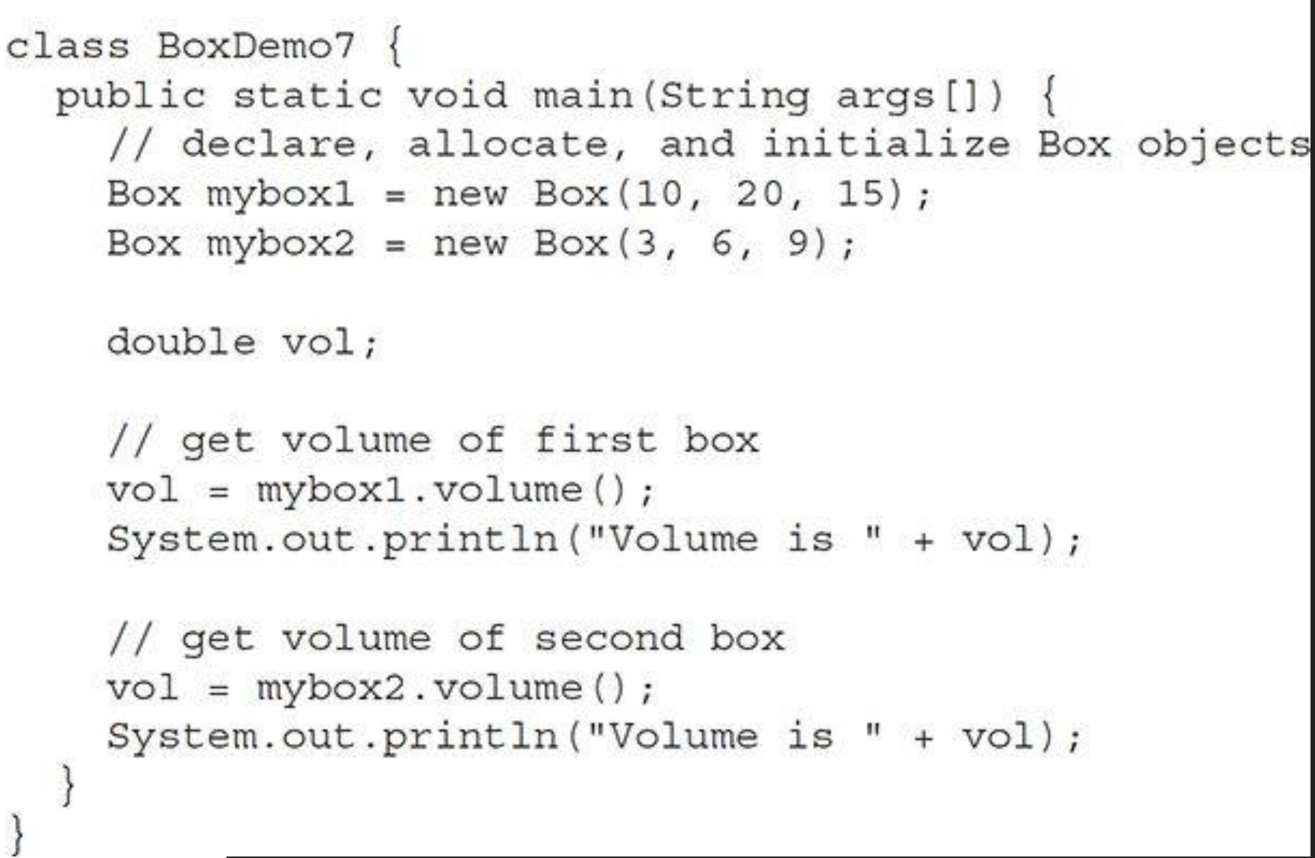


```
class BoxDemo6 {
  public static void main(String args[]) {
    // declare, allocate, and initialize Box objects
    Box mybox1 = new Box();
    Box mybox2 = new Box();

    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);
  }
}
```

Parametrized Constructors



```
/* 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);
  }
}
```