Constructor Overloading in Java

Difficulty Level : Easy Last Updated : 28 Jun, 2021

Prerequisite – Constructor, Overloading in java

In addition to overloading methods, we can also overload constructors in java. Overloaded constructor is called based upon the parameters specified when new is executed.

When do we need Constructor Overloading?

Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading. For example, Thread class has 8 types of constructors. If we do not want to specify anything about a thread then we can simply use default constructor of Thread class, however if we need to specify thread name, then we may call the parameterized constructor of Thread class with a String args like this:

```
Thread t= new Thread (" MyThread ");
```

Let us take an example to understand need of constructor overloading. Consider the following implementation of a class Box with only one constructor taking three arguments.

```
// An example class to understand need of
// constructor overloading.
class Box
{
    double width, height,depth;

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

    // compute and return volume
    double volume()
    {
```

```
        return width * height * depth;
    }
}
```

As we can see that the Box() constructor requires three parameters. This means that all declarations of Box objects must pass three arguments to the Box() constructor. For example, the following statement is currently invalid:

```
Box ob = new Box();
```

Since Box() requires three arguments, it's an error to call it without them. Suppose we simply wanted a box object without initial dimension, or want to initialize a cube by specifying only one value that would be used for all three dimensions. From the above implementation of Box class these options are not available to us.

These types of problems of different ways of initializing an object can be solved by constructor overloading. Below is the improved version of class Box with constructor overloading.

```
// Java program to illustrate
// Constructor Overloading
class Box
{
    double width, height, 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 = height = depth = 0;
    }

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

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

// Driver code
public class Test
{
    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);

        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 mycube is " + vol);
    }
}
```

Output:

```
Volume of mybox1 is 3000.0
Volume of mybox2 is 0.0
Volume of mycube is 343.0
```

Using this() in constructor overloading

this() reference can be used during constructor overloading to call default constructor implicitly from parameterized constructor. Please note, this() should be the first statement inside a constructor.

```java
// Java program to illustrate role of this() in
// Constructor Overloading
class Box
{
    double width, height, depth;
    int boxNo;

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

    // constructor used when no dimensions specified
    Box()
    {
        // an empty box
        width = height = depth = 0;
    }

    // constructor used when only boxNo specified
    Box(int num)
    {
        // this() is used for calling the default
        // constructor from parameterized constructor
        this();

        boxNo = num;
    }

    public static void main(String[] args)
    {
        // create box using only boxNo
        Box box1 = new Box(1);

        // getting initial width of box1
        System.out.println(box1.width);
    }
}
```

Output:

```
0.0
```

As we can see in the above program that we called Box(int num) constructor during object creation using only box number. By using this() statement inside it, the default constructor(Box()) is implicitly called from it which will initialize dimension of Box with 0.

Note : The constructor calling should be first statement in the constructor body. For example, following fragment is invalid and throws compile time error.

```
Box(int num)
{
    boxNo = num;

    /* Constructor call must be the first
       statement in a constructor */
    this();  /*ERROR*/
}
```

Important points to be taken care while doing Constructor Overloading :

- Constructor calling must be the **first** statement of constructor in Java.
- If we have defined any parameterized constructor, then compiler will not create default constructor. and vice versa if we don't define any constructor, the compiler creates the default constructor(also known as no-arg constructor) by default during compilation
- Recursive constructor calling is invalid in java.

Constructors overloading vs Method overloading

Strictly speaking, constructor overloading is somewhat similar to method overloading. If we want to have different ways of initializing an object using different number of parameters, then we must do constructor overloading as we do method overloading when we want different definitions of a method based on different parameters.

This article is contributed by **Gaurav Miglani**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.