Copy Constructor in Java

Difficulty Level : Easy Last Updated : 17 May, 2021

Prerequisite – Constructors in Java
Like C++, Java also supports copy constructor. But, unlike C++, Java doesn't create a default copy constructor if you don't write your own.

Following is an example Java program that shows a simple use of copy constructor.

```java
// filename: Main.java

class Complex {

    private double re, im;

    // A normal parameterized constructor
    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    // copy constructor
    Complex(Complex c) {
        System.out.println("Copy constructor called");
        re = c.re;
        im = c.im;
    }

    // Overriding the toString of Object class
    @Override
    public String toString() {
        return "(" + re + " + " + im + "i)";
    }
}

public class Main {

    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);

        // Following involves a copy constructor call
        Complex c2 = new Complex(c1);

        // Note that following doesn't involve a copy constructor call as
        // non-primitive variables are just references.
        Complex c3 = c2;

        System.out.println(c2); // toString() of c2 is called here
    }
}
```

Output:

```
Copy constructor called
(10.0 + 15.0i)
```

Now try the following Java program:

```
// filename: Main.java

class Complex {

    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }
}

public class Main {

    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(c1);  // compiler error here
    }
}
```

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

Only Java Can Get
The Job Done For You.
So Strengthen Your Foundations and
Start Learning