Java Copy Constructor Example

Like C++, Java also supports the **copy constructor**. But in C++ it is created by default. While in Java we define copy constructor our own. In this section, we will learn the **copy constructor in Java with an example**.

Constructor

In Java, a constructor is the same as a method but the only difference is that the constructor has the same name as the class name. It is used to create an instance of the class. It is called automatically when we create an object of the class. It has no return type. Remember that a constructor cannot be abstract, final, synchronized, and static. We cannot override a constructor. It occupies some space in memory when it is called.

Types of Constructor

- Default Constructor
- Parameterized Constructor

Except for the above two constructors, Java also supports the copy constructor. Let's discuss it in detail.

Why copy constructor is required?

Sometimes, we face a problem where we required to create an exact copy of an existing object of the class. There is also a condition, if we have made any changes in the copy it should not reflect in the original one and vice-versa. For such cases, Java provides the concept of a **copy constructor**.

Copy Constructor

In Java, a copy constructor is a special type of constructor that creates an object using another object of the same Java class. It returns a duplicate copy of an existing object of the class.

We can assign a value to the final field but the same cannot be done while using the clone() method. It is used if we want to create a deep copy of an existing object. It is easier to implement in comparison to the clone() method.

> Note: It cannot be inherited by the subclasses. If we try to initialize a child class object from a parent class reference, we face the casting problem when cloning it with the copy constructor.

Use of Copy Constructor

We can use the copy constructor if we want to:

- Create a copy of an object that has multiple fields.
- Generate a deep copy of the heavy objects.
- Avoid the use of the Object.clone() method.

Advantages of Copy Constructor

- If a field declared as final, the copy constructor can change it.
- There is no need for typecasting.
- Its use is easier if an object has several fields.
- Addition of field to the class is easy because of it. We need to change only in the copy constructor.

Creating a Copy Constructor

To create a copy constructor in Java, follow the steps given below:

- Create a constructor that accepts an object of the same class as a parameter.

```
public class Fruits
{
private double price;
private String name;
//copy constructor
public Fruits(Fruits fruits)
{
//getters
}
}
```

- Copy each field (variable) object into the newly created instance.

```
public class Fruits
{
private double price;
private String name;
//copy constructor
public Fruits(Fruits fruits)
{
//copying each filed
this.price = fruits.price; //getter
this.name = fruits.name; //getter
}
```

```
}
```

Example of Copy Constructor

CopyConstructorExample.java

```
public class Fruit
{
private double fprice;
private String fname;
//constructor to initialize roll number and name of the student
Fruit(double fPrice, String fName)
{
fprice = fPrice;
fname = fName;
}
//creating a copy constructor
Fruit(Fruit fruit)
{
System.out.println("\nAfter invoking the Copy Constructor:\n");
fprice = fruit.fprice;
fname = fruit.fname;
}
//creating a method that returns the price of the fruit
double showPrice()
{
return fprice;
}
//creating a method that returns the name of the fruit
String showName()
{
return fname;
}
//class to create student object and print roll number and name of the student
public static void main(String args[])
{
Fruit f1 = new Fruit(399, "Ruby Roman Grapes");
System.out.println("Name of the first fruit: "+ f1.showName());
System.out.println("Price of the first fruit: "+ f1.showPrice());
//passing the parameters to the copy constructor
Fruit f2 = new Fruit(f1);
System.out.println("Name of the second fruit: "+ f2.showName());
System.out.println("Price of the second fruit: "+ f2.showPrice());
}
```

```
}
```

Output:

```
Name of the first fruit: Ruby Roman Grapes
Price of the first fruit: 399.0

After invoking the Copy Constructor:

Name of the second fruit: Ruby Roman Grapes
Price of the second fruit: 399.0
```

Copy Constructor Vs clone() Method

Both the copy constructor and the clone() method are used to create a copy of an existing object of the class. But the use of copy constructor is easier and better in comparison to the clone() method because of the reasons given below:

- If we are using the clone() method it is necessary to import the **Cloneable** The method may throw the exception **CloneNotSupportException.** So, handling the exception in a program is a complex task. While in copy constructor there are no such complexities.
- We cannot assign a value if the fields are **final**. While in the copy constructor we can assign values to the final fields.
- The object returned by the clone() method must be **typecast**. While in copy constructor there is no such requirement.

← Prev Next →

Learn Latest Tutorials

Splunk tutorial
Splunk

SPSS tutorial
SPSS

Swagger tutorial
Swagger

T-SQL tutorial
Transact-SQL

Tumblr tutorial
Tumblr

React tutorial
ReactJS

Regex tutorial
Regex

Reinforcement learning tutorial
Reinforcement Learning

R Programming tutorial
R Programming

RxJS tutorial
RxJS

React Native tutorial
React Native

Python Design Patterns
Python Design Patterns

Python Pillow tutorial
Python Pillow

Python Turtle tutorial
Python Turtle

Keras tutorial
Keras

Preparation

Aptitude
Aptitude

Logical Reasoning
Reasoning

Verbal Ability
Verbal Ability

Interview Questions
Interview Questions

Company Interview Questions
Company Questions

Trending Technologies

Artificial Intelligence Tutorial
Artificial Intelligence

AWS Tutorial
AWS

Selenium tutorial
Selenium

Cloud Computing tutorial
Cloud Computing

Hadoop tutorial
Hadoop

B.Tech / MCA

Data Warehouse