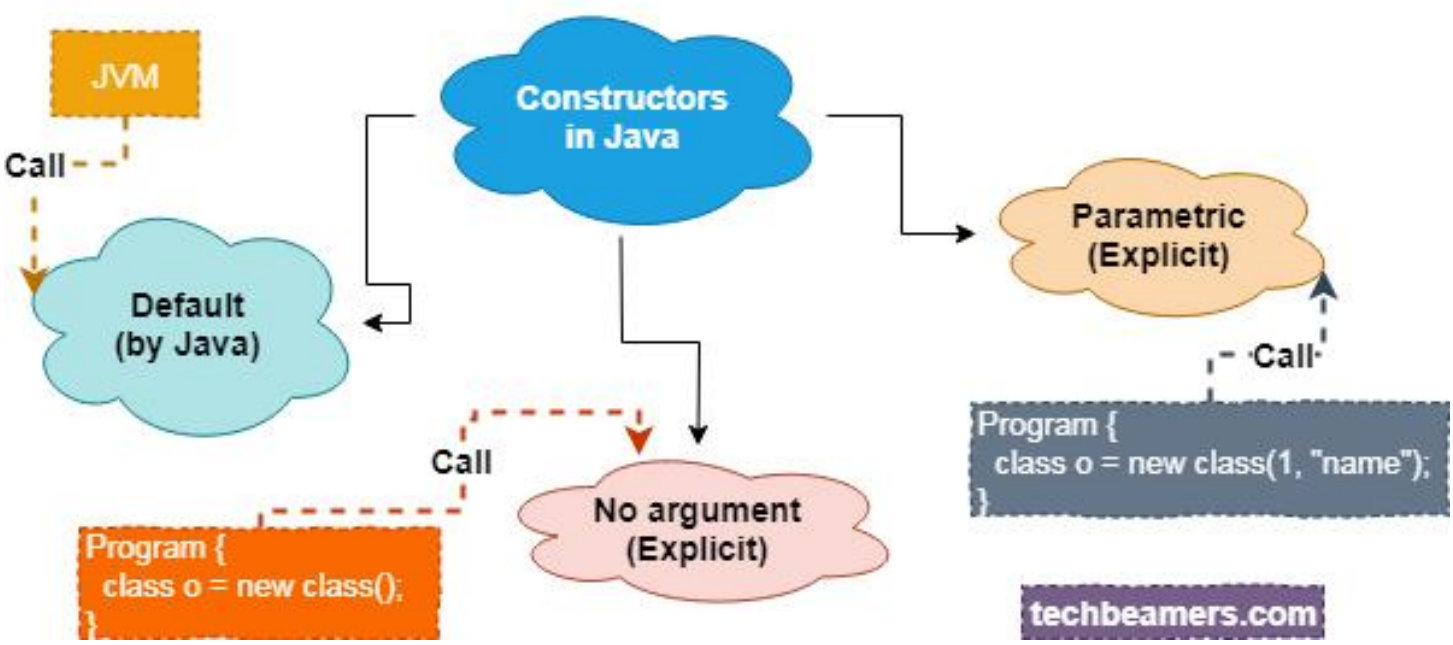
Java Constructor

A constructor in Java is a block of code that is called when an instance of an object is created and memory is allocated for the object. It is a special type of method used for initializing an object. Using access modifiers while declaring a constructor is also possible.

Constructors are an important part of learning Java effectively. So, let's kick-off this comprehensive guide to Java constructors with the rules pertaining to the creation of a Java constructor:



Rules for Creating a Java Constructor

- A Java constructor mustn't have an explicit return type
- It can't be abstract, final, static, or synchronized
- The constructor name must be the same as the one belonging to its class

Types of Java Constructors

There are two types of constructors in Java:

1. Default Constructor or no-arg constructor

The Java Default Constructor has no parameters. This is why it is also known as a no-arg constructor. The general syntax of a Java default constructor is:

<class_name>(){}

What's interesting to note is that if there is no constructor defined in a Java class, then the Java compiler automatically creates a default constructor for the class. Depending on the type of the object, the default constructor provides the default values to the object.

The drawback of using a default constructor that is automatically created by the javac is that afterwards the programmer is not able to set the initial values for object attributes.

Example:

```
class ConstructorDemo

{

ConstructorDemo()public static void main(String args[]){

ConstructorDemo a = new ConstructorDemo();

}

}
```

Output:

```
The constructor is created successfully!
```

2. Parameterized Constructor

Any Java constructor with a number of parameters is termed as a parameterized constructor. Although a parameterized constructor is generally used for providing distinct values to different Java objects, it can also provide the same values to distinct Java objects.

Example:

```
class ParaConst

{int id;

String name;

ParaConst(int i, String n)

{

id = i;

name = n;

}void display()public static void main(String args[]){

ParaConst s1 = new ParaConst(121, “Akhil”);

ParaConst s2 = new ParaConst(232, “Vijay”);

s1.display();

s2.display();

}

}
```

Output:

```
121 Akhil232 Vijay
```

Constructor Overloading

Like Java methods, it is possible to overload constructors in Java. With constructor overloading, one can have the same constructor but with different parameter lists. All of them are arranged in such a way that each of them performs a distinct task.

The Java compiler differentiates between the overloaded constructors by the total number of parameters in the list and their types. Following code snippet demonstrates constructor overloading in Java:

```
class OverloadConst{

int id;

String name;int age;

OverloadConst(int i,String n)

{

id = i;

name = n;

}

OverloadConst(int i, String n, int a)

{

id = i;

name = n;

age = a;

}void display()public static void main(String args[]){

OverloadConst s1 = new OverloadConst(121, “Akhil”);

OverloadConst s2 = new OverloadConst(232, “Vijay”,25);

s1.display();

s2.display();

}}
```

O

Constructor vs. Method in Java

A Java method is a piece of code that has some specific name. It can be invoked during any point in the program by simply using the method name. It can also be understood as a subprogram that operates on data and returns some value.

The Java constructor is a special type of method. Both are similar in many ways, but not identical. Here are some of the most important differences between a Java constructor and a Java method:

- **Invoking** – While the constructor is invoked implicitly, the method is invoked explicitly
- **Java Compiler** – The Java compiler never provides a Java method. However, the Java compiler provides a default constructor if one isn't defined in a Java class
- Naming Convention – The name of the constructor in Java must be the same as that of the class. However, the method may or may not have the same name as that of the class containing it
- **Number of Calls** – A Java constructor is called once and only during the time of object creation. A Java method, on the other hand, can be called as many times are required
- **Return Type** – A Java method must have a return type but having the same for a constructor isn't mandatory
- Use – While a method is used for exposing the behavior of a Java object, a constructor is used for initializing the state of the same

Copy Constructor in Java

Although there is no provision for a copy constructor in Java, it is possible to copy values from one Java object to the other just like using a copy constructor in C++.

Other than using a constructor for copying values from one object to another, the same can also be accomplished by:

- Assigning the values of one object to the other

Or

- By using the clone() method of the Object class

Following program demonstrates using a Java constructor for copying values from one Java object to the other:

```
class Copy

{int id;

String name;

Copy(int i,String n)

{

id = i;

name = n;

}

Copy(Copy s)

{

id = s.id;

name = s.name;

}void display()public static void main(Strong args[]){

Copy s1 = new Copy(121, ”Akhil”);

Copy s2 = new Coopy(s1);

s1.display();

s2.display();
```

```
}

}
```

Output:

```
121 Akhil121 Akhil
```

Some FAQs about the Java Constructor

Following are some of the most frequently asked questions regarding the Java constructor. Some of these are among the most popular Java interview questions:

Q: Does constructor return any value?
A: Although you can't use a return type with a Java constructor, it returns a value. A Java constructor returns the current class instance.

Q: What is Constructor Chaining in Java?
A: Constructor chaining is the technique of calling a constructor from some other constructor in Java programming language. While the this() method is used for calling the same class constructor, the super() method is used for calling the superclass constructor.

Q: Is it possible to call the subclass constructor from the superclass constructor in Java?
A: No.

Q: Does Java have destructors?
A: Java doesn't have destructors because it is a garbage collected language. It is not possible to predict when an object will be destroyed in Java.

Q: What tasks other than initialization can be performed by a Java constructor?
A: A Java constructor can perform any kind of operation that can be performed using a method. Some of the most popular tasks carried out using a Java constructor are:

- Calling a method
- Object creation
- Starting a thread

Q: When does the need for constructor overloading arise in Java?
A: Constructor overloading is used in Java typically when there is a requirement of initializing a Java object in several different ways.

Q: What will happen if you add a return type for a Java constructor?
A: A Java constructor with a return type will be treated as a typical Java method along with the Java compiler generating a "this method has a constructor name" warning.

Summary

So, that was all about the Java constructor. Learning how to effectively use constructors is among the important bits for building sound know-how of the high-level, general-purpose programming language.

Source : https://hackr.io/blog/java-constructor