Contents

[Constructors in Java 1](#_Toc124279920)

[Need of Constructor 1](#_Toc124279921)

[When is a Constructor called ? 1](#_Toc124279922)

[Types of constructor 2](#_Toc124279923)

[No-argument constructor 2](#_Toc124279924)

[Parameterized Constructor 2](#_Toc124279925)

[Constructor Overloading 3](#_Toc124279926)

[How constructors are different from methods in Java? 3](#_Toc124279927)

[Constructor Chaining 3](#_Toc124279928)

[Why do we need constructor chaining ? 3](#_Toc124279929)

[Constructor Chaining within same class using this() keyword : 4](#_Toc124279930)

[Rules of constructor chaining : 4](#_Toc124279931)

[Alternative method : using Init block : 4](#_Toc124279932)

# Constructors in Java

Constructors are used to initialize the object’s state. Like [methods](https://www.geeksforgeeks.org/methods-in-java/), a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation.

Need of Constructor  
Think of a Box. If we talk about a box class then it will have some class variables (say length, breadth, and height). But when it comes to creating its object(i.e Box will now exist in computer’s memory), then can a box be there with no value defined for its dimensions. The answer is no.  
So constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor).

When is a Constructor called ?  
Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members**of the same class.

## Constructor Properties

* Constructor(s) of a class must has same name as the class name in which it resides.
* A constructor in Java can not be abstract, final, static and Synchronized.
* Access modifiers can be used in constructor declaration to control its access i.e which other class can call the constructor.
* Constructor should not have return type

Since you cannot override a constructor you cannot provide body to it if it is made abstract. Therefore, you cannot use abstract keyword with the constructor

Whenever you make a method final, you cannot override it. i.e. you cannot provide implementation to the superclass's final method from the subclass.

i.e. The purpose of making a method final is to prevent modification of a method from outside (child class).

In inheritance whenever you extend a class. The child class inherits all the members of the superclass except the constructors.

In other words, constructors cannot be inherited in Java therefore you cannot override constructors.

So, writing final before constructors makes no sense. Therefore, java does not allow final keyword before a constructor.

If you try, make a constructor final a compile time error will be generated saying “modifier final not allowed here”

We know static keyword belongs to a class rather than the object of a class. A constructor is called when an object of a class is created, so no use of the static constructor. Another thing is that if we will declare static constructor then we can not access/call the constructor from a subclass. Because we know static is allowed within a class but not by a subclass.

Constructors cannot be synchronized — using the synchronized keyword with a constructor is a syntax error. Synchronizing constructors doesn't make sense, because only the thread that creates an object should have access to it while it is being constructed

## **Types of constructor**

There are two type of constructor in Java:

### No-argument constructor

A constructor that has no parameter is known as default constructor. If we don’t define a constructor in a class, then compiler creates **default constructor(with no arguments)** for the class. And if we write a constructor with arguments or no-arguments then the compiler does not create a default constructor.  
Default constructor provides the default values to the object like 0, null, etc. depending on the type.

### Parameterized Constructor

A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use a parameterized constructor.

class Geek

{

    // data members of the class.

    String name;

    int id;

    // constructor would initialize data members

    // with the values of passed arguments while

    // object of that class created.

    Geek(String name, int id)

    {

        this.name = name;

        this.id = id;

    }

}

# Constructor Overloading

Like methods, we can overload constructors for creating objects in different ways. Compiler differentiates constructors on the basis of numbers of parameters, types of the parameters and order of the parameters.

# How constructors are different from methods in Java?

* Constructor(s) must have the same name as the class within which it defined while it is not necessary for the method in java.
* Constructor(s) do not return any type while method(s) have the return type or void if does not return any value.
* Constructor is called only once at the time of Object creation while method(s) can be called any numbers of time.

# Constructor Chaining

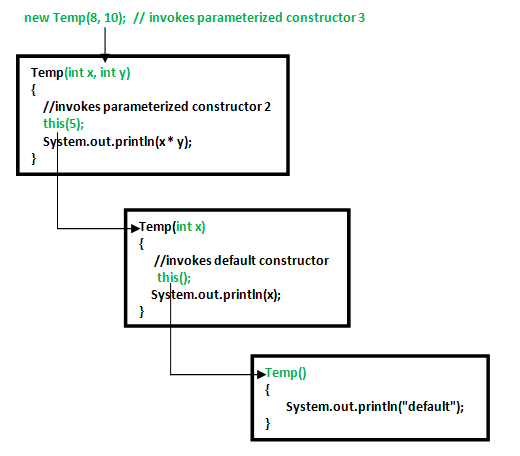
Constructor chaining is the process of calling one constructor from another constructor with respect to current object.  
Constructor chaining can be done in two ways:

* Within same class: It can be done using this() keyword for constructors in same class
* From base class: by using super() keyword to call constructor from the base class.

Constructor chaining occurs through **inheritance**. A sub class constructor’s task is to call super class’s constructor first. This ensures that creation of sub class’s object starts with the initialization of the data members of the super class. There could be any numbers of classes in inheritance chain. Every constructor calls up the chain till class at the top is reached.

Why do we need constructor chaining ?  
This process is used when we want to perform multiple tasks in a single constructor rather than creating a code for each task in a single constructor we create a separate constructor for each task and make their chain which makes the program more readable.

### Constructor Chaining within same class using this() keyword :

[](https://media.geeksforgeeks.org/wp-content/uploads/Constructor-Chaining-In-Java1.png)

## Rules of constructor chaining :

1. The this() expression should always be the first line of the constructor.
2. There should be at-least be one constructor without the this() keyword (constructor 3 in above example).
3. Constructor chaining can be achieved in any order.

Note : Similar to constructor chaining in same class, **super()** should be the first line of the constructor as super class’s constructor are invoked before the sub class’s constructor.

Alternative method : using Init block :  
When we want certain common resources to be executed with every constructor we can put the code in the **[init block](https://www.geeksforgeeks.org/g-fact-26-the-initializer-block-in-java/)**. Init block is always executed before any constructor, whenever a constructor is used for creating a new object.

# Constructor in Java Abstract Class

A constructor is used to initialize an object not to build the object. As we all know abstract classes also do have a constructor. So if we do not define any constructor inside the abstract class then [JVM](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) (Java Virtual Machine) will give a default constructor to the abstract class. If we want to know how to define user define constructors like constructors with argument or any kind of constructor inside the abstract class then you should follow the given procedure

* If you define your own constructor without arguments inside an abstract class but forget to call your own constructor inside its derived class constructor then JVM will call the constructor by default.
* So if you define your single or multi-argument constructor inside the abstract class then make sure to call the constructor inside the derived class constructor with the [super keyword](https://www.geeksforgeeks.org/super-keyword/).

# Constructor in an interface

* An Interface in Java doesn't have a constructor because all data members in interfaces are public static final by default, they are constants (assign the values at the time of declaration).
* There are no data members in an interface to initialize them through the constructor.
* In order to call a method, we need an object, since the methods in the interface don’t have a body there is no need for calling the methods in an interface.
* Since we cannot call the methods in the interface, there is no need of creating an object for an interface and there is no need of having a constructor in it.

# Enum Constructors

Check after Enum datatypes