**Java Constructors**

A constructor in Java is similar to a method that is invoked when an [object](https://www.programiz.com/java-programming/class-objects#objects) of the [class](https://www.programiz.com/java-programming/class-objects) is created.

Unlike [Java methods](https://www.programiz.com/java-programming/methods), a constructor has the same name as that of the class and does not have any return type. For example,

class Test {

Test() {

// constructor body

}

}

Here, Test() is a constructor. It has the same name as that of the class and doesn't have a return type.

**Also Read:** [Why do constructors not return values](https://stackoverflow.com/questions/1788312/why-do-constructors-not-return-values)

**Example: Java Constructor**

class Main {

private String name;

// constructor

Main() {

System.out.println("Constructor Called:");

name = "Programiz";

}

public static void main(String[] args) {

// constructor is invoked while

// creating an object of the Main class

Main obj = new Main();

System.out.println("The name is " + obj.name);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**:

Constructor Called:

The name is Programiz

In the above example, we have created a constructor named Main().

Inside the constructor, we are initializing the value of the name [variable](https://www.programiz.com/java-programming/variables-literals).

Notice the statement creating an object of the Main class.

Main obj = new Main();

Here, when the object is created, the Main() constructor is called. And the value of the name variable is initialized.

Hence, the program prints the value of the name variables as Programiz.

**Types of Constructor**

In Java, constructors can be divided into three types:

1. No-Arg Constructor
2. Parameterized Constructor
3. Default Constructor

**1. Java No-Arg Constructors**

Similar to methods, a Java constructor may or may not have any parameters (arguments).

If a constructor does not accept any parameters, it is known as a no-argument constructor. For example,

private Constructor() {

// body of the constructor

}

**Example: Java Private No-arg Constructor**

class Main {

int i;

// constructor with no parameter

private Main() {

i = 5;

System.out.println("Constructor is called");

}

public static void main(String[] args) {

// calling the constructor without any parameter

Main obj = new Main();

System.out.println("Value of i: " + obj.i);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**:

Constructor is called

Value of i: 5

In the above example, we have created a constructor Main().

Here, the constructor does not accept any parameters. Hence, it is known as a no-arg constructor.

**Notice that we have declared the constructor as private.**

Once a constructor is declared private, it cannot be accessed from outside the class.

So, creating objects from outside the class is prohibited using the private constructor.

Here, we are creating the object inside the same class.

Hence, the program is able to access the constructor. To learn more, visit [Java Implement Private Constructor](https://www.programiz.com/java-programming/examples/private-constructor-implementation).

However, if we want to create objects outside the class, then we need to declare the constructor as public.

**Example: Java Public no-arg Constructors**

class Company {

String name;

// public constructor

public Company() {

name = "Programiz";

}

}

class Main {

public static void main(String[] args) {

// object is created in another class

Company obj = new Company();

System.out.println("Company name = " + obj.name);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Company name = Programiz

**Also Read:** [Java Access Modifier](https://www.programiz.com/java-programming/access-modifiers)

**2. Java Parameterized Constructor**

A Java constructor can also accept one or more parameters. Such constructors are known as parameterized constructors (constructors with parameters).

**Example: Parameterized Constructor**

class Main {

String languages;

// constructor accepting single value

Main(String lang) {

languages = lang;

System.out.println(languages + " Programming Language");

}

public static void main(String[] args) {

// call constructor by passing a single value

Main obj1 = new Main("Java");

Main obj2 = new Main("Python");

Main obj3 = new Main("C");

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Java Programming Language

Python Programming Language

C Programming Language

In the above example, we have created a constructor named Main().

Here, the constructor takes a single parameter. Notice the expression:

Main obj1 = new Main("Java");

Here, we are passing the single value to the constructor.

Based on the argument passed, the language variable is initialized inside the constructor.

**3. Java Default Constructor**

If we do not create any constructor, the Java compiler automatically creates a no-arg constructor during the execution of the program.

This constructor is called the default constructor.

**Example: Default Constructor**

class Main {

int a;

boolean b;

public static void main(String[] args) {

// calls default constructor

Main obj = new Main();

System.out.println("Default Value:");

System.out.println("a = " + obj.a);

System.out.println("b = " + obj.b);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Default Value:

a = 0

b = false

Here, we haven't created any constructors.

Hence, the Java compiler automatically creates the default constructor.

The default constructor initializes any uninitialized instance variables with default values.

|  |  |
| --- | --- |
| Type | Default Value |
| boolean | false |
| byte | **0** |
| short | **0** |
| int | **0** |
| long | **0L** |
| char | \u0000 |
| float | **0.0f** |
| double | **0.0d** |
| object | Reference null |

To learn more, visit [Java Data Types](https://www.programiz.com/java-programming/variables-primitive-data-types).

In the above program, the variables a and b are initialized with default value **0** and false respectively.

The above program is equivalent to:

class Main {

int a;

boolean b;

Main() {

a = 0;

b = false;

}

public static void main(String[] args) {

// call the constructor

Main obj = new Main();

System.out.println("Default Value:");

System.out.println("a = " + obj.a);

System.out.println("b = " + obj.b);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Default Value:

a = 0

b = false

**Important Notes on Java Constructors**

* Constructors are invoked implicitly when you instantiate objects.
* The two rules for creating a constructor are:  
  1. The name of the constructor should be the same as the class.  
  2. A Java constructor must not have a return type.
* If a class doesn't have a constructor, the Java compiler automatically creates a **default constructor** during run-time. The default constructor initializes instance variables with default values. For example, the int variable will be initialized to 0
* Constructor types:  
  **No-Arg Constructor** - a constructor that does not accept any arguments  
  **Parameterized constructor** - a constructor that accepts arguments  
  **Default Constructor** - a constructor that is automatically created by the Java compiler if it is not explicitly defined.
* A constructor cannot be abstract or static or final.
* A constructor can be overloaded but can not be overridden.

**Constructors Overloading in Java**

Similar to [Java method overloading](https://www.programiz.com/java-programming/method-overloading), we can also create two or more constructors with different parameters. This is called constructor overloading.

**Example: Java Constructor Overloading**

class Main {

String language;

// constructor with no parameter

Main() {

this.language = "Java";

}

// constructor with a single parameter

Main(String language) {

this.language = language;

}

public void getName() {

System.out.println("Programming Language: " + this.language);

}

public static void main(String[] args) {

// call constructor with no parameter

Main obj1 = new Main();

// call constructor with a single parameter

Main obj2 = new Main("Python");

obj1.getName();

obj2.getName();

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Programming Language: Java

Programming Language: Python

In the above example, we have two constructors: Main() and Main(String language).

Here, both the constructors initialize the value of the variable language with different values.

Based on the parameter passed during object creation, different constructors are called, and different values are assigned.

It is also possible to call one constructor from another constructor. To learn more, visit [Java Call One Constructor from Another](https://www.programiz.com/java-programming/examples/call-one-constructor-from-another).

**Note**: We have used this keyword to specify the variable of the class. To know more about this keyword, visit [Java this keyword](https://www.programiz.com/java-programming/this-keyword).