Constructors in Java

In [Java](https://www.javatpoint.com/java-tutorial), a constructor is a block of codes similar to the method. It is called when an instance of the [class](https://www.javatpoint.com/object-and-class-in-java) is created. At the time of calling constructor, memory for the object is allocated in the heap memory.

**It is a special type of method which is used to initialize the object**.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

**Note:** It is called constructor because it constructs the values at the time of object creation.

**It is not necessary to write a constructor for a class**. It is because java compiler creates a default constructor if your class doesn't have any.

### Rules for creating Java constructor

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Note: We can use [access modifiers](https://www.javatpoint.com/access-modifiers) while declaring a constructor. It controls the object creation. In other words, we can have private, protected, public or default constructor in Java.

## Types of Java constructors

There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

## Java Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

### Syntax of default constructor:

1. <class\_name>(){}

## Example of default constructor

|  |
| --- |
|  |

1. //Java Program to create and call a default constructor
2. **class** Bike1{
3. //creating a default constructor
4. Bike1(){System.out.println("Bike is created");}
5. //main method
6. **public** **static** **void** main(String args[]){
7. //calling a default constructor
8. Bike1 b=**new** Bike1();
9. }
10. }

#### Rule: If there is no constructor in a class, compiler automatically creates a default constructor.

### Q) What is the purpose of a default constructor?

The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

### Example of default constructor that displays the default values

1. //Let us see another example of default constructor
2. //which displays the default values
3. **class** Student3{
4. **int** id;
5. String name;
6. Student3() { id=20; name=”STCET Student”}
7. //method to display the value of id and name
8. **void** display(){System.out.println(id+" "+name);}
10. **public** **static** **void** main(String args[]){
11. //creating objects
12. Student3 s1=**new** Student3();
13. Student3 s2=**new** Student3();
14. //displaying values of the object
15. s1.display();
16. s2.display();
17. }
18. }

**Explanation:**In the above class,you are not creating any constructor so compiler provides you a default constructor. Here 0 and null values are provided by default constructor.

### Java Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.

### Why use the parameterized constructor?

The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

### Example of parameterized constructor

In this example, we have created the constructor of Student class that have two parameters. We can have any number of parameters in the constructor.

1. //Java Program to demonstrate the use of the parameterized constructor.
2. **class** Student4{
3. **int** id;
4. String name;
5. //creating a parameterized constructor
6. Student4(**int** i,String n){
7. id = i;
8. name = n;
9. }
10. //method to display the values
11. **void** display(){System.out.println(id+" "+name);}
13. **public** **static** **void** main(String args[]){
14. //creating objects and passing values
15. Student4 s1 = **new** Student4(111,"abhinav");
16. Student4 s2 = **new** Student4(222,"sayan");
17. //calling method to display the values of object
18. s1.display();
19. s2.display();
20. }
21. }

## Constructor Overloading in Java

In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.

Constructor [overloading in Java](https://www.javatpoint.com/method-overloading-in-java) is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

### Example of Constructor Overloading

1. //Java program to overload constructors
2. **class** Student5{
3. **int** id;
4. String name;
5. **int** age;
6. //creating two arg constructor
7. Student5(**int** i,String n){
8. id = i;
9. name = n;
10. }
11. //creating three arg constructor
12. Student5(**int** i,String n,**int** a){
13. id = i;
14. name = n;
15. age=a;
16. }
17. **void** display(){System.out.println(id+" "+name+" "+age);}
19. **public** **static** **void** main(String args[]){
20. Student5 s1 = **new** Student5(111,"Karan");
21. Student5 s2 = **new** Student5(222,"Aryan",25);
22. s1.display();
23. s2.display();
24. }
25. }

## Difference between constructor and method in Java

There are many differences between constructors and methods. They are given below.

|  |  |
| --- | --- |
| **Java Constructor** | **Java Method** |
| A constructor is used to initialize the state of an object. | A method is used to expose the behavior of an object. |
| A constructor must not have a return type. | A method must have a return type. |
| The constructor is invoked implicitly. | The method is invoked explicitly. |
| The Java compiler provides a default constructor if you don't have any constructor in a class. | The method is not provided by the compiler in any case. |
| The constructor name must be same as the class name. | The method name may or may not be same as the class name. |

Array of objects

class Main{

   public static void main(String args[]){

     //create array of employee object

    Employee[] obj = new Employee[2] ;

     //create & initialize actual employee objects using constructor

     obj[0] = new Employee(100,"ABC");

     obj[1] = new Employee(200,"XYZ");

     //display the employee object data

     System.out.println("Employee Object 1:");

     obj[0].showData();

     System.out.println("Employee Object 2:");

     obj[1].showData();

  }

}

//Employee class with empId and name as attributes

class Employee{

  int empId;

  String name;

  //Employee class constructor

  Employee(inteid, String n){

     empId = eid;

     name = n;

  }

public void showData(){

   System.out.print("EmpId = "+empId + "  " + " Employee Name = "+name);

   System.out.println();

 }

}

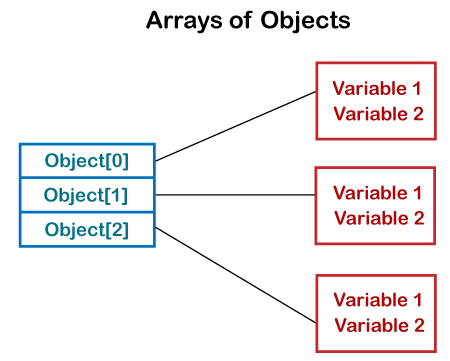
# How to Create Array of Objects in Java

In this section, we will learn **how to create and initialize an array of objects in Java**.

## Array of Objects in Java

Java is an object-oriented programming language. Most of the work done with the help of **objects**. We know that an array is a collection of the same data type that dynamically creates objects and can have elements of primitive types. Java allows us to store objects in an array. In [Java](https://www.javatpoint.com/java-tutorial)

, the class is also a user-defined data type. An array that conations **class type elements** are known as an **array of objects**. It stores the reference variable of the object.



## Creating an Array of Objects

Before creating an array of objects, we must create an instance of the class by using the new keyword. We can use any of the following statements to create an array of objects.

1. **public** **class** ArrayOfObjects
2. {
3. **public** **static** **void** main(String args[])
4. {
5. //create an array of product object
6. Product[] obj = **new** Product[5] ;
7. //create & initialize actual product objects using constructor
8. obj[0] = **new** Product(23907,"Dell Laptop");
9. obj[1] = **new** Product(91240,"HP 630");
10. obj[2] = **new** Product(29823,"LG OLED TV");
11. obj[3] = **new** Product(11908,"MI Note Pro Max 9");
12. obj[4] = **new** Product(43590,"Kingston USB");
13. //display the product object data
14. System.out.println("Product Object 1:");
15. obj[0].display();
16. System.out.println("Product Object 2:");
17. obj[1].display();
18. System.out.println("Product Object 3:");
19. obj[2].display();
20. System.out.println("Product Object 4:");
21. obj[3].display();
22. System.out.println("Product Object 5:");
23. obj[4].display();
24. }
25. }
26. //Product class with product Id and product name as attributes
27. **class** Product
28. {
29. **int** pro\_Id;
30. String pro\_name;
31. //Product class constructor
32. Product(**int** pid, String n)
33. {
34. pro\_Id = pid;
35. pro\_name = n;
36. }
37. **public** **void** display()
38. {
39. System.out.print("Product Id = "+pro\_Id + "  " + " Product Name = "+pro\_name);
40. System.out.println();
41. }
42. }

**Through loop**

import java.util.\*;

public class ArrayOfObjects

{

public static void main(String[] args)

{

// Create an array to hold rectangle's data.

Rectangle[] rectangles = new Rectangle[5];

double length; //to hold length of rectangle

double width; //to hold width of rectangle

// Create a Scanner object for keyboard input.

Scanner console = new Scanner(System.in);

Rectangle[] rectangles = new Rectangle[5];

for (int i = 0; i < 5; i++)

{

System.out.print("Enter the length of Rectangle " + (i + 1)

+ ": ");

length = console.nextDouble();

System.out.print("Enter the width of Rectangle " + (i + 1)

+ ": ");

width = console.nextDouble();

rectangles[i] = new Rectangle(length, width);

System.out.println();

}

for (int i = 0; i < 5; i++)

{

System.out.println("Area of Rectangle " + (i + 1) + ": "

+ rectangles[i].getArea());

}

}

}