Constructor in Java

In Java, constructor is a block of code similar to

a method. when an instance of object is

created and memory is allocated for the object then the constructor is called.

It is a special type of method which is used to initialize

the object.

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 constructor if your class

doesn't have any.

Rules for creating java constructor

There are basically two rules defined for the

constructor.

1.Constructor name must be same as its class name(case sensitive)

2.Constructor must have no return type

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:

public Bike1(){

}

Example of default constructor

In this example, we are creating the no-arg

constructor in the Bike class. It will be invoked at the

time of object creation.

public class Bike1 {

Bike1() {

System.out.println("Bike is created.....");

}

public static void main(String args[]) {

Bike1 b = new Bike1();

}

}

Output:

Bike is created.....

Rule: If there is no constructor in a class,

compiler automatically creates a default

constructor.

public class Bike2 {

public static void main(String args[]) {

Bike2 b = new Bike2();

}

}

---------------------------------------------

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Java 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. //method to display the value of id and name 7. **void** display(){System.out.println(id+" "+name);} 9. **public** **static** **void** main(String args[]){ 10. //creating objects 11. Student3 s1=**new** Student3(); 12. Student3 s2=**new** Student3(); 13. //displaying values of the object 14. s1.display(); 15. s2.display(); 16. } 17. }   Output:  0 null  0 null  **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 the 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 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,"Karan"); 16. Student4 s2 = **new** Student4(222,"Aryan"); 17. //calling method to display the values of object 18. s1.display(); 19. s2.display(); 20. } 21. }   Output:  111 Karan  222 Aryan 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 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 in java 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. }   Output:  111 Karan 0  222 Aryan 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 class name. |   Java Constructors vs Methods Copying values without constructor We can copy the values of one object into another by assigning the objects values to another object. In this case, there is no need to create the constructor.   1. **class** Student7{ 2. **int** id; 3. String name; 4. Student7(**int** i,String n){ 5. id = i; 6. name = n; 7. } 8. Student7(){} 9. **void** display(){System.out.println(id+" "+name);} 11. **public** **static** **void** main(String args[]){ 12. Student7 s1 = **new** Student7(111,"Karan"); 13. Student7 s2 = **new** Student7(); 14. s2.id=s1.id; 15. s2.name=s1.name; 16. s1.display(); 17. s2.display(); 18. } 19. }   Output:  111 Karan  111 Karan Q) Does constructor return any value? Yes, it is the current class instance (You cannot use return type yet it returns a value). Can constructor perform other tasks instead of initialization? Yes, like object creation, starting a thread, calling a method, etc. You can perform any operation in the constructor as you perform in the method. |

## J