CONSTRUCTOR IN JAVA

Constructors

- A constructor is a special member function of a class that initializes the data members of that class.
- Whenever we are creating an object, some piece of code will be executed automatically to perform initialization. This piece of code is nothing but a constructor. Hence, the main objective of the constructor is to perform initialization for the newly created object.
- A constructor is invoked automatically whenever an object of that class is created.
- In Java, a constructor is a block of codes similar to the method. It is called when an instance of the is created, and memory is allocated for the object.
- It is called a constructor because it constructs the values at the time of object creation.

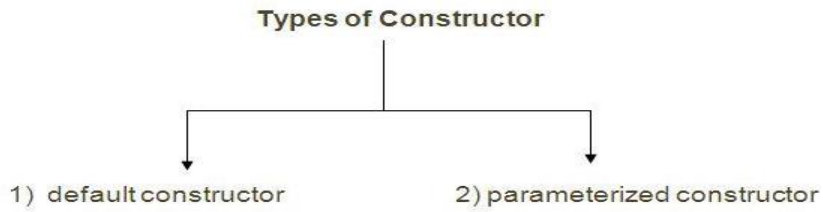
Rules to define Constructor:

- The constructor's name must be the same as its class name.
- A constructor does not have any return type (even void also), By mistake if we declare a return type for the constructor, the compiler will treat it as a normal method.
- The only applicable modifiers for the constructor are private, protected, public, or default (PPPD), if we are trying any other modifier, we will get compile time error.-"Modifier xxxxxx, is not allowed here".
- A Java constructor cannot be abstract, static, final……, and synchronized.

Types of constructors

There are two types of constructors:

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



DEFAULT CONSTRUCTOR

A constructor that has no parameter is known as a default constructor.

- Every time an object is created using a new () keyword, at least one constructor is called. It is called a default constructor.
- If we are not writing any constructor in class then the compiler will automatically generate the default constructor.
- If we are writing at least one constructor in class then the compiler will not generate the default constructor.
- Hence, a class can contain either programmer written constructor or compiler generated constructor but not both simultaneously.

Syntax of default constructor:

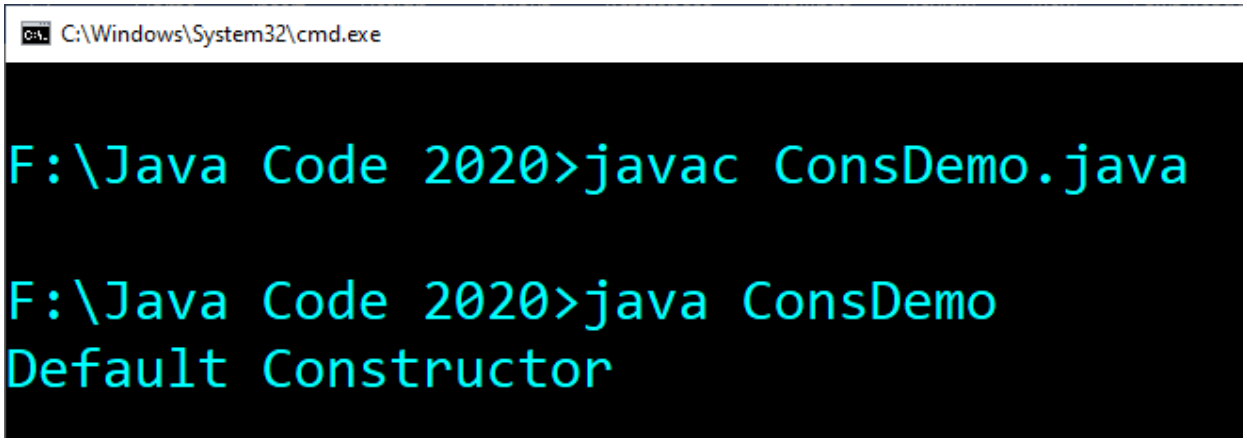
```
class_name()
{
Statement/s
}
```

EXAMPLE OF DEFAULT CONSTRUCTOR

In this example, we are creating the no-args constructor in the ConsDemo class. It will be invoked at the time of object creation.

```
class ConsDemo
{
 ConsDemo() //default constructor
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();

}
}
```



```java
 class ConsDemo
{
ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
ConsDemo c1=new ConsDemo();
}
}
```

```
D:\Java Code 2k23>java ConsDemo
Default Constructor
Default Constructor
```

```java
 class ConsDemo
{
ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();

c.ConsDemo();
}
}
```

```
D:\Java Code 2k23>javac ConsDemo.java
ConsDemo.java:11: error: cannot find symbol
c.ConsDemo();
 ^
  symbol:   method ConsDemo()
  location: variable c of type ConsDemo
1 error
```

The behavior of the Constructor with access modifiers.
Example

```
class ConsDemo
{
 private ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```

```
F:\Java Code 2020>javac ConsDemo.java

F:\Java Code 2020>java ConsDemo
Default Constructor
```

```
class ConsDemo
{
 public ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```

```
F:\Java Code 2020>javac ConsDemo.java

F:\Java Code 2020>java ConsDemo
Default Constructor
```

```
class ConsDemo
{
 final ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```

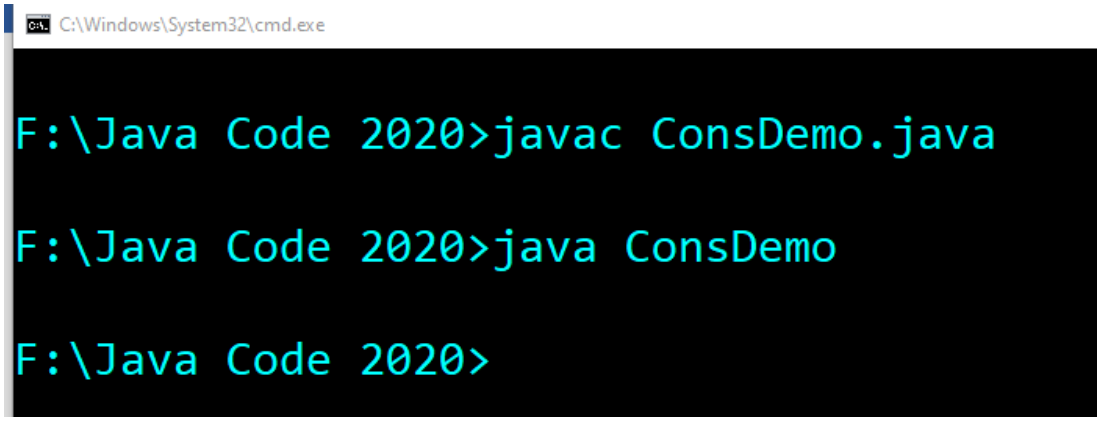
```
F:\Java Code 2020>javac ConsDemo.java
ConsDemo.java:3: error: modifier final not allowed here
 final ConsDemo()
       ^
1 error
```

```
class ConsDemo
{
 transient ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```

```
F:\Java Code 2020>javac ConsDemo.java
ConsDemo.java:3: error: modifier transient not allowed here
 transient ConsDemo()
           ^
1 error
```

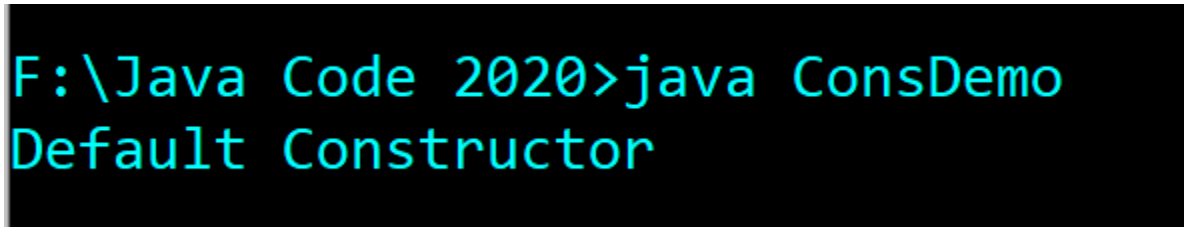
The behavior of the Constructor with return type:

```
class ConsDemo
{
 void ConsDemo()//This is a method, not a constructor
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```



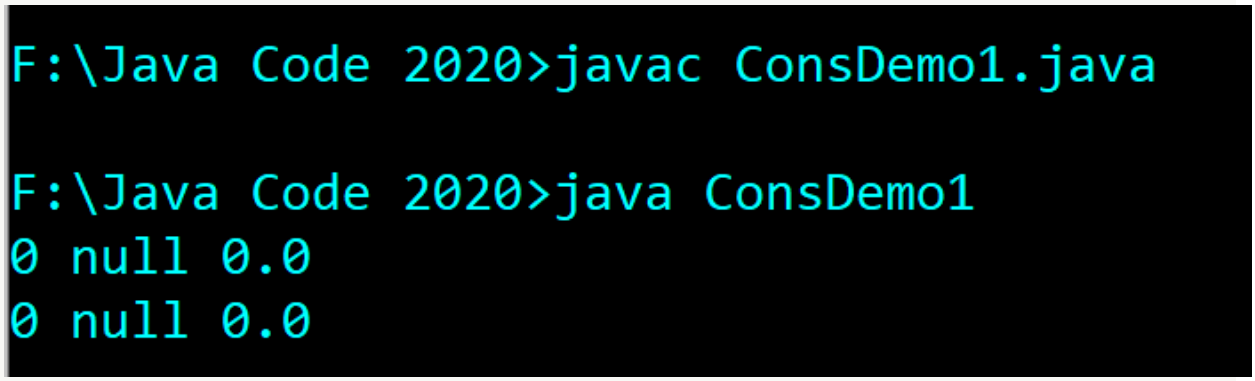
void ConsDemo()- Compiler treated as a normal method, not a constructor.

```
class ConsDemo
{
 void ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
c.ConsDemo();//explicit call
}
}
```



EXAMPLE OF A DEFAULT CONSTRUCTOR THAT DISPLAYS THE DEFAULT VALUES

```
class ConsDemo1
{
	int id;
	String name;
	double marks;
ConsDemo1()
{
}
	void display()
	{
	System.out.println(id+" "+name+" " +marks);
	}
	public static void main(String args[])
	{
	ConsDemo1 c1=new ConsDemo1();
	ConsDemo1 c2=new ConsDemo1();
	c1.display();
	c2.display();
	}
}
```



Here 0 ,null and 0.0 values are provided by the default constructor.

NO USER-DEFINED CONSTRUCTOR IS PRESENT IN THIS CLASS

```java
public class ConsDemo1
{

    int id;
    String name;
    double marks;

    void display()
    {
    System.out.println(id+" "+name+" " +marks);
    }
    public static void main(String args[])
    {
    ConsDemo1 c1=new ConsDemo1();
    ConsDemo1 c2=new ConsDemo1();
    c1.display();
    c2.display();
    }
}
```

```
D:\Java Code 2k23>javac ConsDemo1.java

D:\Java Code 2k23>java ConsDemo1
0 null 0.0
0 null 0.0
```

Here 0 ,null and 0.0 values are provided by the compiler-generated default constructor.

In this class, we are not creating any constructor, so the compiler generates a default constructor.

Note:
If we are not writing any constructor in class, then the compiler will automatically generate the default constructor.

If we are writing at least one constructor in class then the compiler will not generate the default constructor.

PARAMETERIZED CONSTRUCTOR

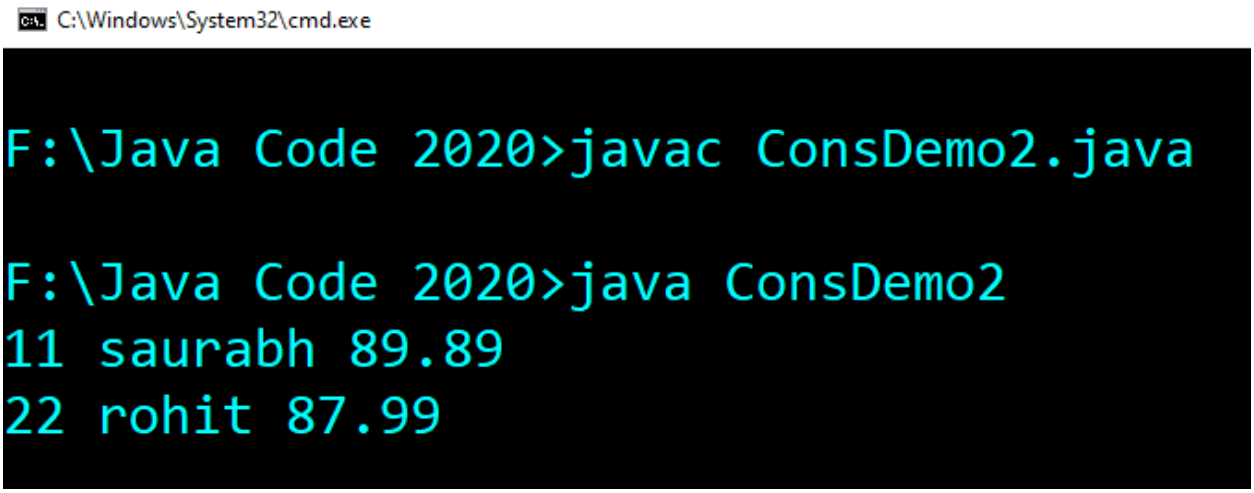
A constructor that has parameters is known as a parameterized constructor. Parameterized constructor is used to provide different values to the distinct objects.

EXAMPLE OF PARAMETERIZED CONSTRUCTOR

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

```
class ConsDemo2
{
        int id;
        String name;
        double marks;
   ConsDemo2(int i,String n, double m)
        {
        id = i;
        name = n;
        marks=m;
   }

        void display()
        {
        System.out.println(id+" "+name+" " +marks);
        }
        public static void main(String args[])
        {
        ConsDemo2 c1=new ConsDemo2(11,"saurabh",89.89);
        c1.display();
        ConsDemo2 c2=new ConsDemo2(22,"rohit", 87.99);
        c2.display();
        }
}
```



CONSTRUCTOR OVERLOADING

Constructor overloading in Java is a concept that allows a class to have more than one constructor method with the same name but with different parameters (in terms of their number, type, or both). When you overload constructors, you provide multiple ways to initialize objects of a class, allowing for greater flexibility in how objects are created.

EXAMPLE OF CONSTRUCTOR OVERLOADING

```
public class ConsOverload
{
		int id;
		String name;
		int age;
		ConsOverload(int i,String n)
			{  //constructor with 2 parameters
		id = i;
		name = n;
		}
		ConsOverload(int i,String n,int a)
			{  //constructor with 3 parameters
		id = i;
		name = n;
		age=a;
		}
		void display(){System.out.println(id+" "+name+" "+age);}
```

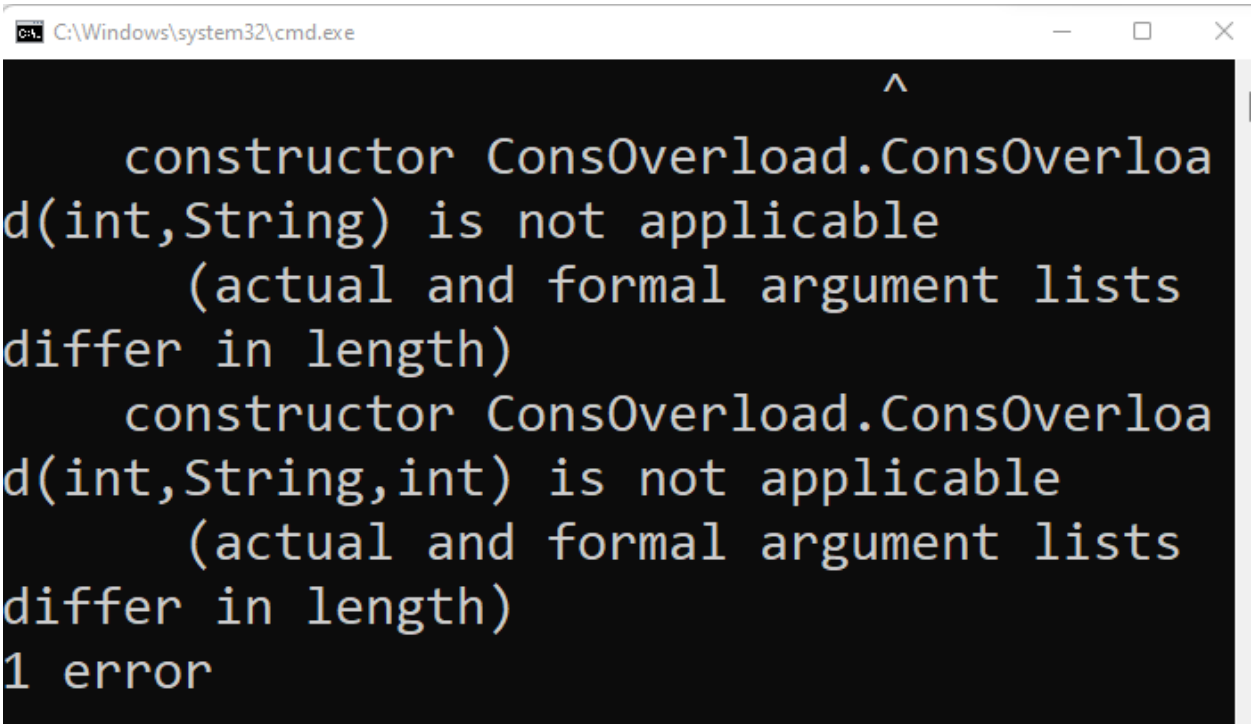
```
        public static void main(String args[]){
        ConsOverload s1 = new ConsOverload(111,"Karan");
        ConsOverload s2 = new ConsOverload(222,"Aryan",25);
        s1.display();
        s2.display();
       }
}
```

```
F:\Java Code 2020>javac ConsOverload.java

F:\Java Code 2020>java ConsOverload
111 Karan 0
222 Aryan 25
```

```
class ConsOverload
{
        int id;
        String name;
        int age;
        ConsOverload(int i,String n)
                {  //constructor with 2 parameters
        id = i;
        name = n;
        }
        ConsOverload(int i,String n,int a)
                {  //constructor with 3 parameters
        id = i;
        name = n;
        age=a;
        }
        void display(){System.out.println(id+" "+name+" "+age);}

        public static void main(String args[]){
        ConsOverload s = new ConsOverload();  // output??
        ConsOverload s1 = new ConsOverload(111,"Karan");
        ConsOverload s2 = new ConsOverload(222,"Aryan",25);
        s1.display();
        s2.display();
       }
}
```



Difference between constructor and method

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

Constructor	Method
A constructor is used to initialize the state of an object.	The method is used to expose the behavior of an object.
The constructor must not have a return type.	The 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.	The method is not provided by the compiler in any case.
The constructor's name must be the same as the class name.	The method name may or may not be the same as the class name.