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 constructor. Hence, the main objective of 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 object is created, and memory is allocated for the object.
- It is called constructor because it constructs the values at the time of object creation.

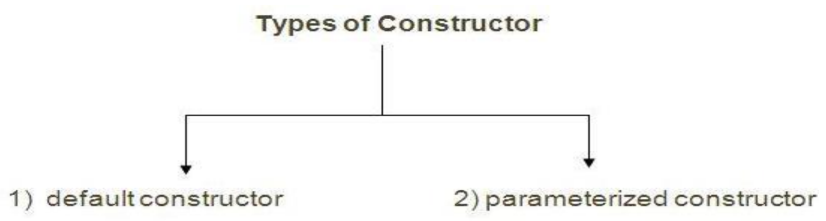
Rules to define Constructor:

- Constructor 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 return type for constructor, we will get Compile time/run time error, because compiler treat it as method.~~
- The only applicable modifiers for ~~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-arg constructor)
2. parameterized constructor



DEFAULT CONSTRUCTOR

A constructor that have no parameter is known as default constructor.

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

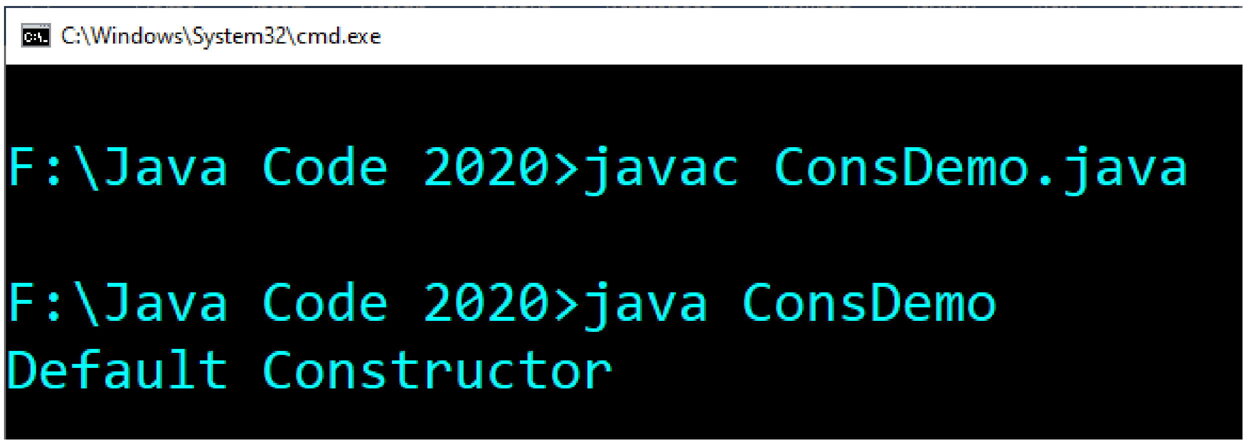
Syntax of default constructor:

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

EXAMPLE OF DEFAULT CONSTRUCTOR

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

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

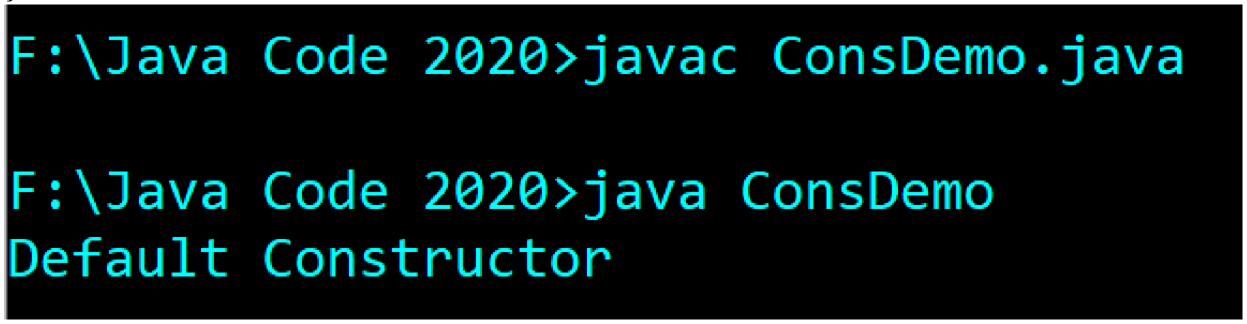


C:\Windows\System32\cmd.exe

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

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

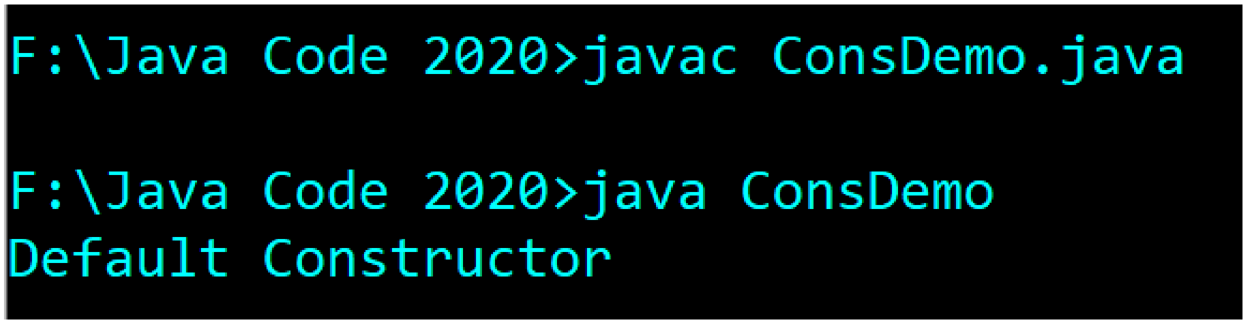
Behavior of Constructor with access modifiers.
Example

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



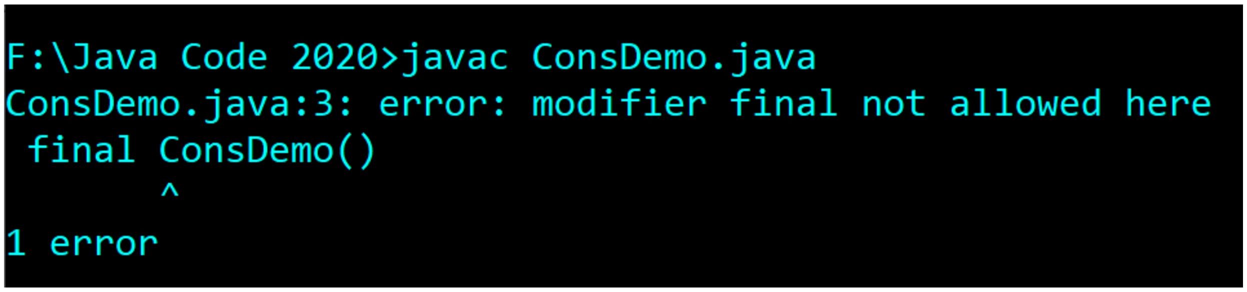
```
F:\Java Code 2020>javac ConsDemo.java

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();
}
}
```



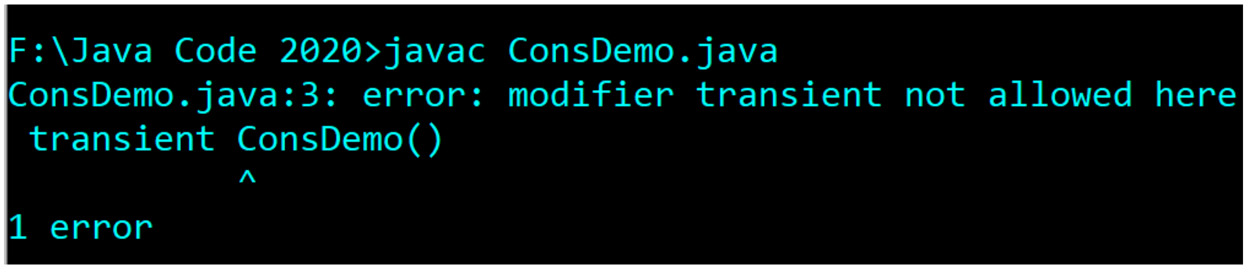
```
F:\Java Code 2020>javac ConsDemo.java

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
```

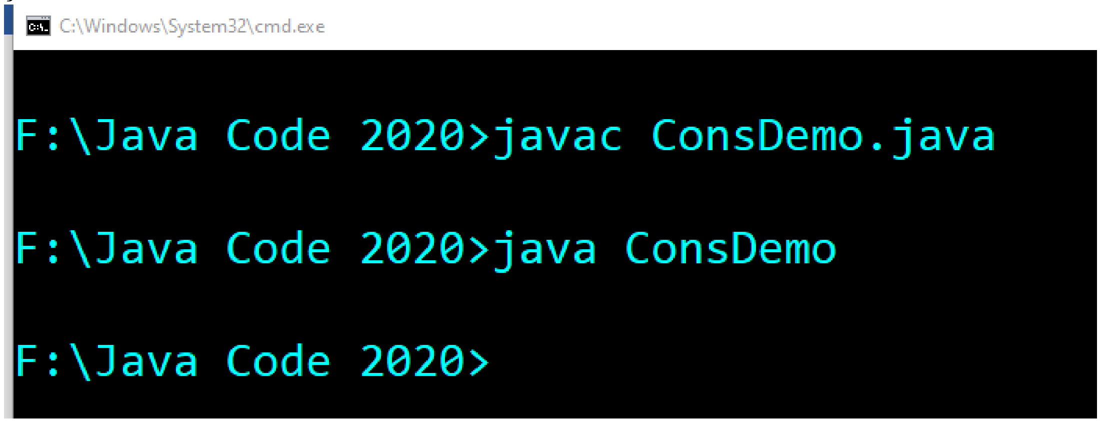
```
class ConsDemo
{
 transient ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```



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

Behavior of Constructor with return type:

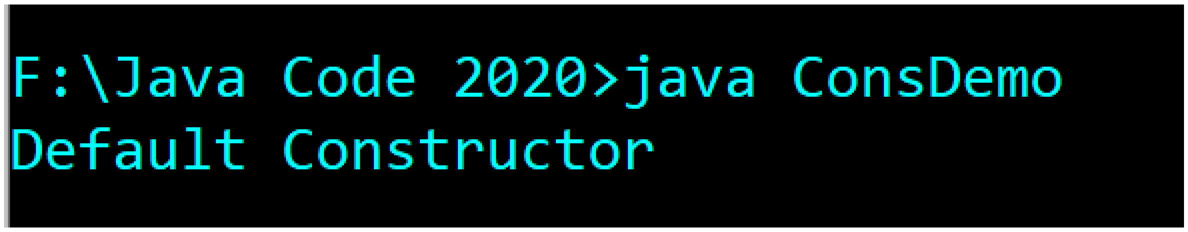
```
class ConsDemo
{
 void ConsDemo()
{
System.out.println("Default Constructor");
}
public static void main(String args[])
{
ConsDemo c=new ConsDemo();
}
}
```



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

F:\Java Code 2020>java ConsDemo

F:\Java Code 2020>
```

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
}
}
```



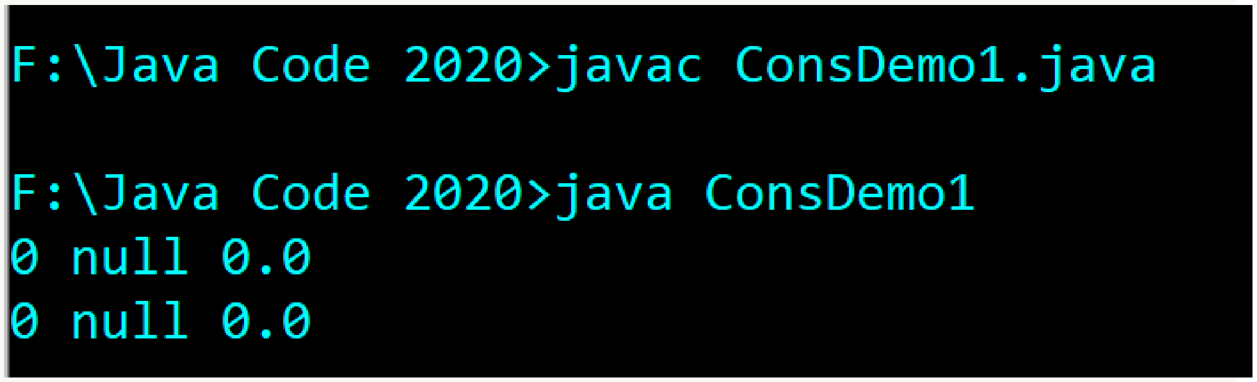
```
F:\Java Code 2020>java ConsDemo
Default Constructor
```

EXAMPLE OF DEFAULT CONSTRUCTOR THAT DISPLAYS THE DEFAULT VALUES

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

//NO USER DEFINED CONSTRUCTOR PRESENT IN THIS CLASS

```
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();
            }
}
```



```
F:\Java Code 2020>javac ConsDemo1.java

F:\Java Code 2020>java ConsDemo1
0 null 0.0
0 null 0.0
```

If we are not writing any constructor in class then compiler will automatically generate default constructor.
Here 0 ,null and 0.0 values are provided by default constructor.

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

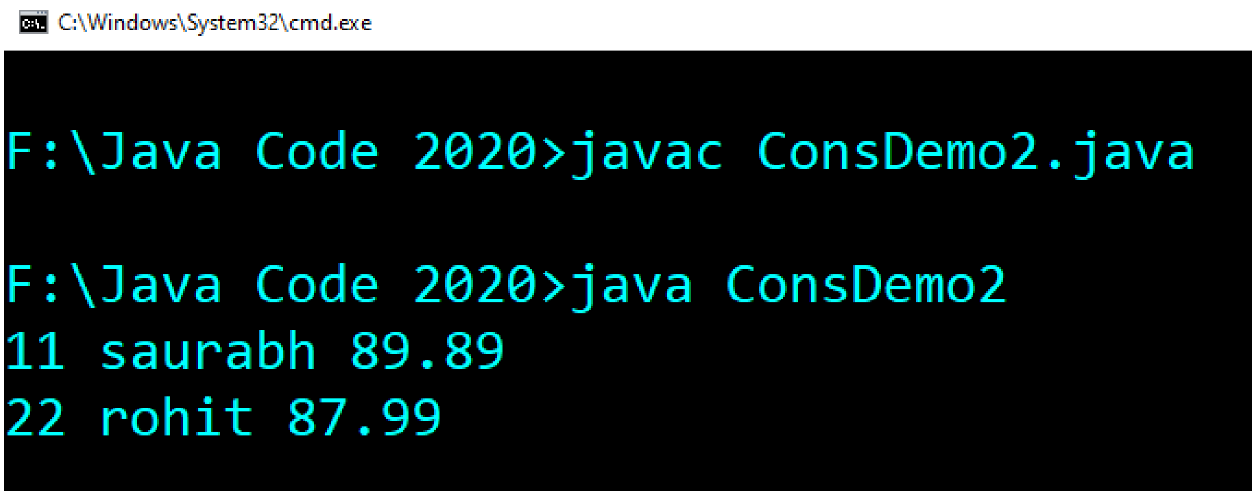
PARAMETERIZED CONSTRUCTOR

A constructor that have parameters is known as 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 Student class that have two 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);
        ConsDemo2 c2=new ConsDemo2(22,"rohit", 87.99);
        c1.display();
        c2.display();
        }
}
```



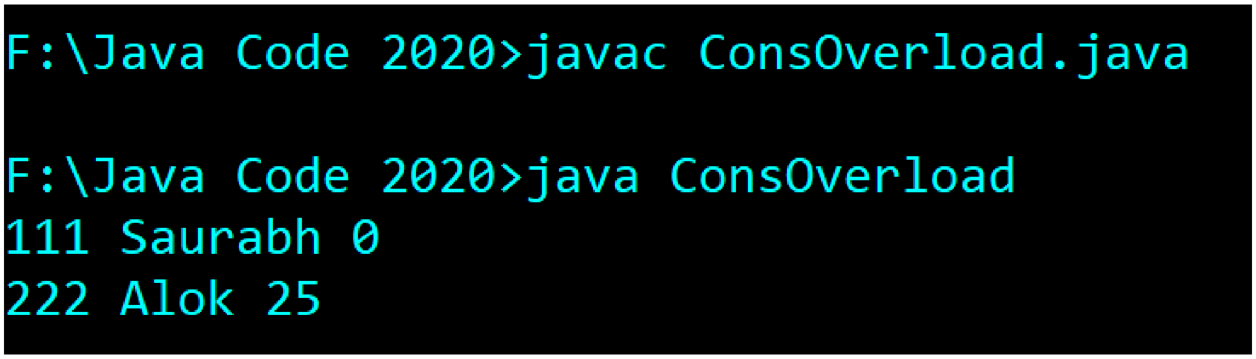
```
C:\Windows\System32\cmd.exe

F:\Java Code 2020>javac ConsDemo2.java

F:\Java Code 2020>java ConsDemo2
11 saurabh 89.89
22 rohit 87.99
```

CONSTRUCTOR OVERLOADING

Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

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,"Saurabh");
                ConsOverload s2 = new ConsOverload(222,"Alok",25);
                s1.display();
                s2.display();
               }
}
```



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

F:\Java Code 2020>java ConsOverload
111 Saurabh 0
222 Alok 25
```

Difference between constructor and method

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

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