

Subject: Java Lab
Semester: III

Class: SE-IT
A.Y. 23-24

Note on Types of Constructors in Java

Constructor in java is a special type of method that is used to initialize the object. Java constructor is invoked at the time of object creation. It constructs the values i.e. initializes values for the member variables that is why it is known as constructor.

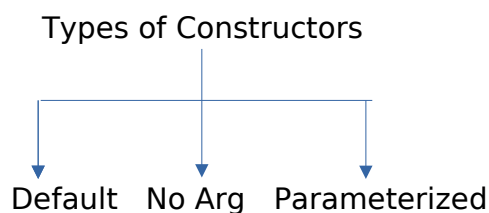
There are basically two rules defined for the constructor.

- 1) Constructor name must be same as its class name
- 2) Constructor must have no explicit return type

Difference between constructor and method in java

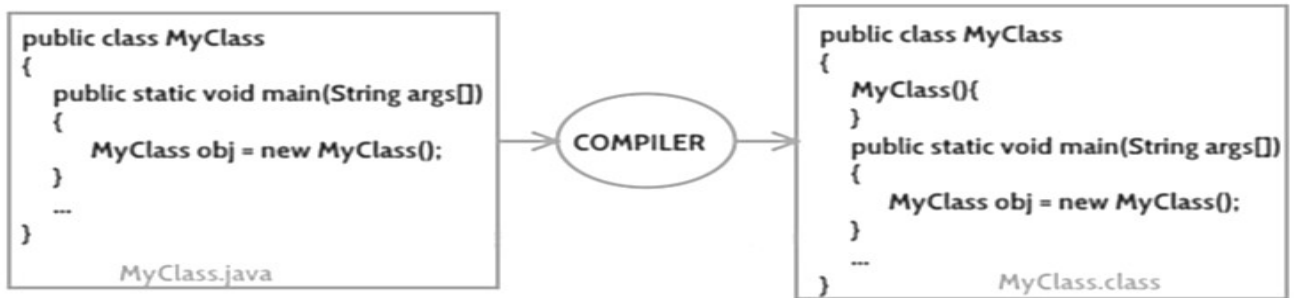
| Java Constructor | Java Method |
|--|--|
| Use to initialize state of an object | Used to expose behaviour of an object |
| Constructor must not have return type | Methods must have return type |
| Constructor is invoked implicitly | Method is invoked explicitly |
| Java compiler provides default constructor if we don't have constructor in program | 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 |

Types of constructors :



1] Default Constructor :

- The java class without having any constructor implemented within it is called as default constructor .
- If you do not implement any constructor in your class, Java compiler inserts a default constructor into your code on your behalf.
- You would not find it in your source code(the java file) as it would be inserted into the code during compilation and exists in .class file. This process is shown in the diagram below:



If you implement any constructor then you no longer receive a default constructor from Java compiler. Consider code shown below:

```

class DefaultConstructor {

    int a;
    boolean b;

    public static void main(String[] args) {

        DefaultConstructor obj = new DefaultConstructor();

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

    }
}

```

Output :

```

a = 0
b = false

```

Default constructor provides the default values to the object like 0, null etc. depending on the type.

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

2] No-Argument Constructor :

- Constructor with no arguments/parameter is known as no-arg constructor.
- The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.
- Although you may see some people claim that default and no-arg constructor is same but in fact they are not, even if you write `public Demo() { }` in your class `Demo` it cannot be called default constructor since you have written the code of it.

Check Example of no-arg constructor below :

```
class NoArgCtor {
    int i;

    // constructor with no parameter
    private NoArgCtor(){
        i = 5;
        System.out.println("Object created and i = " + i);
    }

    public static void main(String[] args) {
        NoArgCtor obj = new NoArgCtor();
    }
}
```

Output :

Object created and i = 5

2] Parameterized constructor

- A constructor that have parameters is known as parameterized constructor.\
- Parameterized constructor is used to provide different values to the distinct objects.
- If we want to initialize fields of the class with your own values, then use parameterized constructor.
- We can have any number of parameters in the constructor.

Check Example of Parameterized constructor below :

```
class Vehicle {
    int wheels;
    private Vehicle(int wheels){
        wheels = wheels;
        System.out.println(wheels + " wheeler vehicle created.");
    }

    public static void main(String[] args) {
        Vehicle v1 = new Vehicle(2);
        Vehicle v2 = new Vehicle(3);
        Vehicle v3 = new Vehicle(4);
    }
}
```

Output :

```
2 wheeler vehicle created.
3 wheeler vehicle created.
4 wheeler vehicle created.
```

Here we have passed argument of data type int (ie. no. of wheels) to the constructor during object creation.

Concept of Constructor Chaining:

- Calling a constructor from the another constructor of same class is known as Constructor chaining.
- The real purpose of Constructor Chaining is that you can pass parameters through a bunch of different constructors, but only have the initialization done in a single place.
- This allows you to maintain your initializations from a single location, while providing multiple constructors to the user.
- If we don't chain, and two different constructors require a specific parameter, you will have to initialize that parameter twice, and when the initialization changes, you'll have to change it in every constructor, instead of just the one.
- `this()` and `super()` are used to call constructors explicitly. Where, using `this()` you can call the current class's constructor and using `super()` you can call the constructor of the super class.
- From normal (default) constructor you can call the parameterized constructors of the same class using `this()` and, from the sub class you can call the constructor of the super class using `super()`

