KONKAN GYANPEETH COLLEGE OF ENGINEERING, KARJAT
(Affiliated to University of Mumbai, Approved by A.I.C.T.E., New Delhi.)
Konkan Gyanpeeth Shaikshanik Sankul, Vengaon Road, Dahivali, Karjat, Dist.-Raigad-410201. (M.S.)
● Department of Information Technology ●
SE IT Java SLB Lab Experiment Writing Instructions

Experiment No: 04

Aim: **U**nderstand concept of constructor, learn about types of constructors and constructor chaining in Java

Theory:

Concept of Constructor in Java

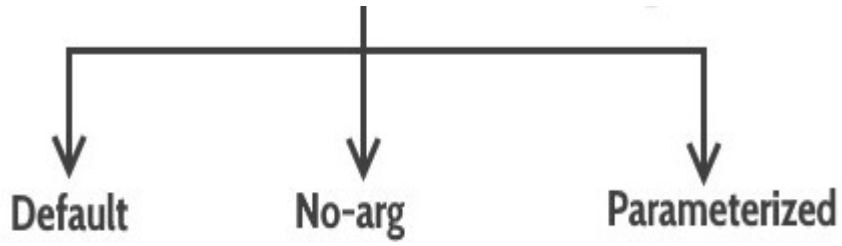
Constructor in java is a *special type of method* that is used to initialize the object.

- Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object 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

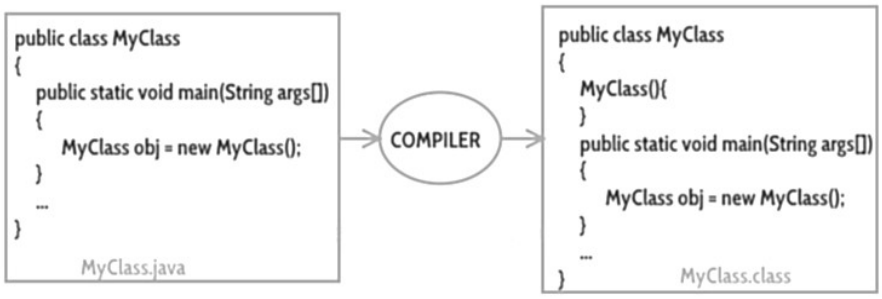
Constructor	Method
Use to initialize state of an object	Used to expose behavior of an object
Constructor must not have return type	Methods must have return type
Constructor is invoked implicitly	Method is invoked explicitly
Java compiler provides default constructor if we don't have constructor in program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



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 we create our own constructor java compiler wont give a default constructor.

```
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);
    }
}
```

```
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

No-Argument Constructor :

- Constructor with no arguments 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.

Example: no-arg constructor

```
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

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.

Example: Parameterized constructor

```
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);
     }
}
```

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

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()

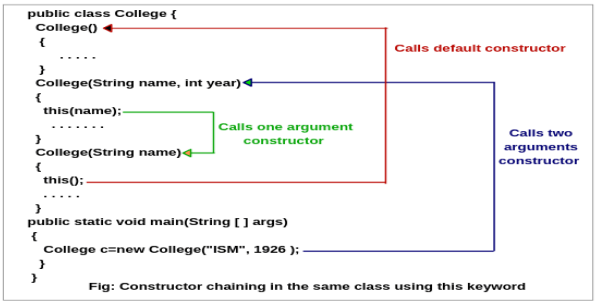


Fig: Constructor chaining in the same class using this keyword

[Summarize above theory into herein 2½ page]

Implementation : [write code from github repository Module II with name Constructor_chaining.java write output on unrules page**]**

Conclusion: Thus we have studied concept of Classes and objects in java programming language.