

**1.What is a Constructor?**

**=>*Constructor***

Object creation is not enough, compulsorily we should perform initialization then only the object is in a position to provide the response properly.

Whenever we are creating an object some piece of the code will be executed automatically to perform initialization of an object. This piece of code is nothing but a constructor.

Main objective of the constructor is nothing but initialisation of Object.

***Rules for writing a constructor***

* Name of the constructor and name of the class must be the same.
* Return type concept not applicable for constructor, even if we provide it won't result in compile time errors, if we do so then the Java language will treat this as "normal method".

**Eg**   
class Test{

void Test(){   
System.out.println("Hello");// It is not a constructor ,it is a method.

}

}

* It is not a good practice to take the method name same as that of the classname .
* The modifiers applicable for constructors are private, public, protected, default.
* The other modifiers if we use, it would result in compile time error.

class Test{

static Test(){

}

}

**2. What is Constructor Chaining?**

**=>**Constructor chaining is the process of calling one constructor from another constructor with respect to current object. One of the main use of constructor chaining is to avoid duplicate codes while having multiple constructor (by means of constructor overloading) and make code more readable.

A class can contain more than one constructor and all these constructors have the same name they differ only in the type of argument, hence these constructors are considered as "Overloaded constructor".

**Eg**   
class Test {

Test(double d) {   
System.out.println("double argument constructor");

}

Test(int i) {

this(10.5);

System.out.println("int argument constructor");

}

Test() {

this(10);

System.out.println("no argument constructor");

}

}

public class MainApp {   
public static void main(String[] args) throws Exception {

}

}

**3.Can we call a subclass constructor from a superclass constructor?**

**=>** No. We cannot call a subclass constructor from a superclass constructor. The reason being, that an instance of a subclass is a instance of the superclass while the vice-a-versa is not true.

**4.What happens if you keep a return type for a constructor?**

**=>**  If we add a return type to a constructor, then it will become a method of the class. This is the way java runtime distinguish between a normal method and a constructor.

**5. What is No-arg constructor?**

**=>** ***No-argument constructor***

A constructor that has no parameter is known as the No-argument or Zero argument constructor. If we don’t define a constructor in a class, then the compiler creates a constructor(with no arguments) for the class. And if we write a constructor with arguments or no arguments then the compiler does not create a default constructor.

**Example:**

// no-argument constructor

class Demo {

int num;

String name;

// this would be invoked while an object of that class is created.

Demo() {

System.out.println("Constructor called");

}

}

class Sample {

public static void main(String[] args)

{

// this would invoke default constructor.

Demo d= new Demo();

// Default constructor provides the default

// values to the object like 0, null

System.out.println(d.name);

System.out.println(d.num);

}

}

**6. How is a No-argument constructor different from the default Constructor?**

**=>*Default Constructor in java:***

* When we write a class without any constructor then at compilation time java compiler creates a default constructor in our class.
* The accessibility modifier of the default constructor is same as accessibility modifier of class.
* The allowed accessibility modifier are public and default.
* Default constructor added by java compiler this constructor does not have anything except super(); call.
* If our class have any constructor then java compiler does not create default constructor

***No-argument Constructor in java:***

* As a developer we can create our own constructor with no arguments is known as no-argument constructor.
* It can have all four accessibility modifiers as it is defined by developer.
* So allowed accessibility modifiers are public, private, protected and default
* It can have logic including super call.
* The common point between default and no-argument constructor
* Both does not have any arguments.
* And one more point we need to remember that in no-argument constructor also by default first statement will be super() call which is added by java compiler if it does not have.

**Note:**Every default constructor is a 0 argument constructor but every 0 argument constructor is not a default constructor.

**7.When do we need Constructor Overloading?**

**=>**  Constructor overloading is a technique in object-oriented programming where a class has multiple constructors with different parameter lists. We may need constructor overloading in the following situations:

**1.Initialization of objects with different values:** If we have a class that represents a real-world entity and that entity can be initialized with different sets of data, we can provide multiple constructors that accept different parameters. For example, a class representing a person could have a constructor that takes just the person's name, and another constructor that takes the person's name, age, and address.

**2.Convenience and ease of use:** Providing multiple constructors can make it easier for clients of the class to create objects without having to provide all of the required data in a single constructor call. This can make the code more readable and easier to understand.

**3.Enhancing code reusability:** By overloading constructors, you can reuse the same code to initialize objects in different ways. This can help to reduce code duplication and make the code more maintainable

and extensible.

**4.Providing default values:** By overloading constructors, you can provide default values for parameters that are not supplied by the user. For example, a constructor that takes an integer parameter could have a default value of 0, which is used when the user does not provide a value.

Overall, constructor overloading can be useful in situations where we need to create objects with different sets of data or provide clients with multiple options for initializing objects. It can also make the code more modular and easier to maintain over time.

**8.What is Default constructor Explain with an Example ?**

**=>** ***Default constructor:***

For every java class constructor concept is applicable.

If we don't write any constructor, then the compiler will generate a default constructor.

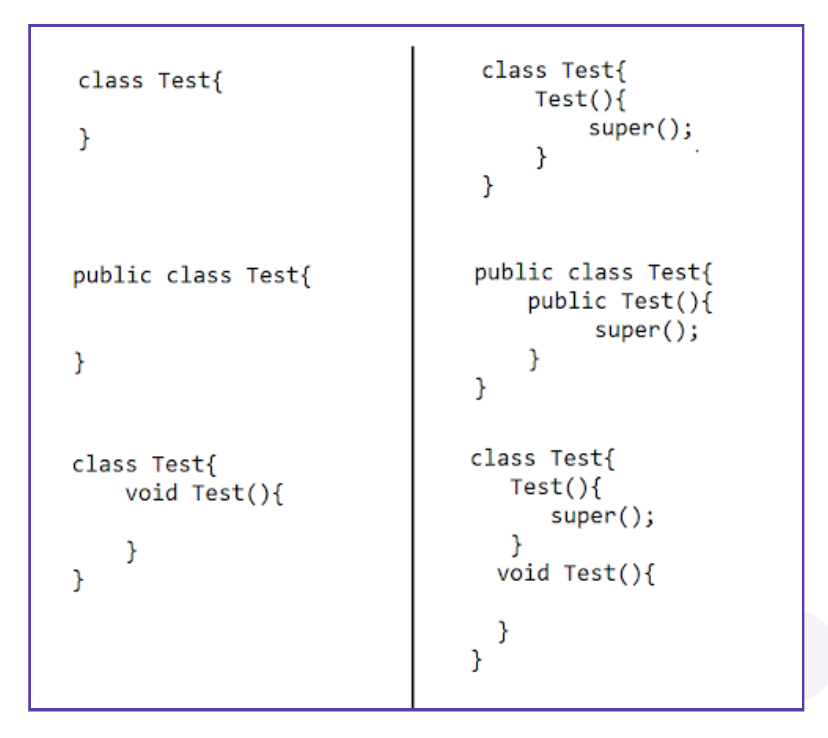
If we write at least one constructor then the compiler won't generate any default constructor, so we say every java class will have a compiler generated default constructor or programmer written constructor but not both simultaneously.

***Prototype of default constructor:***

There is always no argument constructor.

The access modifier of the default constructor is the same as the class modifier. [applicable for public and default.

Default constructor contains one line, super(). It is a call to super class constructor.



***Example:***

// Main.java

class Test {

int i;

boolean b;

byte bt;

float ft;

}

public class Main {

public static void main(String args[]) {

Test t = new Test(); // default constructor is called.

System.out.println(t.i); //output:0

System.out.println(t.b); //output:false

System.out.println(t.bt); //output:0

System.out.println(t.ft); //output:0.0

}

}