**Constructor Assignment Question**

1.what is the constructor?

 a constructor is a block of codes similar to the method. It is called when an instance of the [class](https://www.javatpoint.com/object-and-class-in-java) is created. At the time of calling the constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

Rules for creating Java constructor

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

2.What is constructor Chaning?

**constructor chaining** is a sequence of invoking [constructors](https://www.javatpoint.com/java-constructor) upon initializing an object. It is used when we want to invoke a number of constructors, one after another by using only an instance. In this section, we will discuss **constructor chaining in Java in detail with proper examples.** Let's have a quick look at **what is a constructor in Java.**

In constructor chain, a constructor is called from another constructor in the same class this process is known as **constructor chaining.**

We can achieve constructor chaining in two ways:

* **Within the same class:** If the constructors belong to the same class, we use **this**
* **From the base class:** If the constructor belongs to different classes (parent and child classes), we use the **super** keyword to call the constructor from the base class.

Rules of Constructor Chaining

* An expression that uses **this** keyword must be the first line of the constructor.
* **Order** does not matter in constructor chaining.
* There must exist at least one constructor that does not use **this**

**package** com.pw.java.Constructor;

**public** **class** ChaningConstructor {

ChaningConstructor()

{

**this**("saurabh");// 1st it call the parameterized constructor which can have the only one string as input

System.***out***.println("this is default Constructor");

}

ChaningConstructor(String str)

{

System.***out***.println("this is parameterized constructor");

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ChaningConstructor c=**new** ChaningConstructor(); // it will be call the default Constructor

}

}

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

No. We cannot call a subclass constructor from a superclass constructor.

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

In Java, constructors do not have a return type. They are a special type of method used for initializing objects. If you try to specify a return type for a constructor, you will get a compilation error.

example of a valid constructor in Java:

public class MyClass {

public MyClass() {

// Constructor logic here

}

}

5.What is No-arg constructor?

A no-arg constructor is a constructor in Java that does not take any arguments. It is also known as a default constructor. If a class does not explicitly define any constructors, Java provides a default no-arg constructor for that class.

public class MyClass {

public MyClass() {

// Constructor logic here

}

}

In this example, MyClass has a no-arg constructor because it does not accept any arguments.

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

In Java, a no-argument constructor and a default constructor are essentially the same thing. Both refer to a constructor that does not take any arguments. However, there can be some confusion in terminology, so let's clarify:

**No-argument constructor:** This is a constructor that explicitly defines no parameters. It's also known as a zero-argument constructor. You write this constructor explicitly in your code, like so:

public class MyClass {

public MyClass() {

// Constructor logic here

}

}

**Default constructor:** This term can refer to two different things:

a. If a class does not have any constructors defined, Java provides a default no-argument constructor automatically. This constructor is created by the Java compiler and is similar to the one you would write explicitly:

```java

public class MyClass {

// Compiler-provided default constructor

public MyClass() {

super(); // Implicit call to superclass constructor

// Default constructor logic here

}

}

```

```java

public class MyClass {

// Compiler-provided default constructor

public MyClass() {

super(); // Implicit call to superclass constructor

// Default constructor logic here

}

}

no-argument constructor is a constructor that takes no arguments, and it can be explicitly defined or provided by the compiler. The term "default constructor" can refer to either the compiler-provided no-argument constructor or a constructor that initializes fields to default values.

7.When do we need Constructor Overloading?

Constructor overloading is useful when you want to create objects of a class with different initial states. It allows you to create multiple constructors in a class, each with a different signature (i.e., a different number or type of parameters).

**public** **class** Student {

//instance variables of the class

**int** id;

String name;

Student(){

System.out.println("this a default constructor");

}

Student(**int** i, String n){

id = i;

name = n;

}

**public** **static** **void** main(String[] args) {

//object creation

Student s = **new** Student();

System.out.println("\nDefault Constructor values: \n");

System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);

System.out.println("\nParameterized Constructor values: \n");

Student student = **new** Student(15, "saurabh");

System.out.println("Student Id : "+student.id + "\nStudent Name : "+student.name);

}

}

8.What is default constructor Explain with an Example

If a constructor have no any parameter then it is called as default Constructor

**class** Test1{

//creating a default constructor

Test1(){System.out.println("Default");}

//main method

**public** **static** **void** main(String args[]){

//calling a default constructor

Bike1 b=**new** Bike1();

}

}