# **Constructors**

**Define a constructor?**

A constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules for constructor are:

Constructor Name should be same as class name.

A constructor must have no return type.

**When does the compiler supply a default constructor for a class?**

The compiler supplies a default constructor for a class if no other constructors are provided.

**Will the compiler create a default constructor if I have a parameterized constructor in the class?**

No, compiler won’t create default constructor if there is parameterized constructor in the class. For example, if I have a class with no constructors, then compiler will create default constructor. For Example:

public class Car {

}

In the above Car class there are no constructors so compiler creates a default constructor.

public class Car {

Car(String name) {

}

}

In this example compiler won’t create any default constructor because already there is one constructor in the Car class.

**Can we have a method name same as class name in java?**

Yes. we can have method name same as class name it won’t throw any compilation error but it shows a warning message that method name is same as class name.

**Can we override constructors in java?**

Only methods can be overridden in java. Constructors can’t be inherited in java. So, there is no point of overriding constructors in java.

**How to call one constructor from the other constructor?**

* Within same class: It can be done using this() keyword for constructors in same class
* From base class: by using super() keyword to call constructor from the base class.

This is also called constructor chaining.

Example:

**package** intquestions;

//Java program to illustrate Constructor Chaining

//within same class Using this() keyword

**class** testclass

{

// default constructor 1

// default constructor will call another constructor

// using this keyword from same class

testclass()

{

// calls constructor 2

**this**(5);

System.***out***.println("The Default constructor");

}

// parameterized constructor 2

testclass(**int** x)

{

// calls constructor 3

**this**(5, 15);

System.***out***.println(x);

}

// parameterized constructor 3

testclass(**int** x, **int** y)

{

System.***out***.println(x \* y);

}

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

{

// invokes default constructor first

**new** testclass();

}

}

If we want to access parent class constructor, use super method.

**package** intquestions;

**class** baseclass{

String name;

baseclass(String name){

**this**.name = name;

System.***out***.println(name);

}

}

**class** testclass **extends** baseclass

{

testclass(){

**this**("constructor calling");

System.***out***.println("No argument constructor");

}

testclass(String str){

**super**(str);

}

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

testclass t = **new** testclass();

}

}

If we are calling a constructor from other constructor using this keyword, the keyword should be first. Otherwise it gives error.

**package** intquestions;

**class** testclass

{

testclass(){

System.***out***.println("No argument constructor");

**this**("constructor calling");

}

testclass(String str){

System.***out***.println(str);

}

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

testclass t = **new** testclass();

}

}

The above gives an error that “Constructor call must be the first statement in a constructor”.