**Why do we need a Constructor?**

Constructors initialize the new object; that is, they set the startup property values for the object. They might also do other things necessary to make the object usable. You can distinguish constructors from other methods of a class because constructors always have the same name as the class.

public returntype(void) methodname();

public classname()

**Rules for Constructors:**

The constructor’s name must be the same as that of the class name in which it resides.

Constructors must not have a return type. If you keep the return type for the constructor, it will be treated as a method.

Every class should have at least one constructor. If you don’t write a constructor for your class, the compiler will give a default constructor.

A constructor in Java cannot be abstract, final, static, and Synchronized.

Access modifiers can be used in constructor declaration to control its access, i.e., which other class can call the constructor.

**package Demo;**

**import java.io.\*;**

**class Student**

**{**

**// data members of the class.**

**String name;**

**int id;**

**// constructor would initialize data members**

**// with the values of passed arguments while**

**// object of that class created.**

**Student (String name, int id)**

**{**

**this.name = name;**

**this.id = id;**

**}**

**}**

**public class ParameterizedConstructor**

**{**

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

**{**

**//This would invoke the parameterized constructor.**

**Student student1 = new Student ("Ashok", 101);**

**System.out.println ("Student Name: " + student1.name +" and Student Id: " + student1.id);**

**}**

**}**

**Default Constructor in Java:**

**package Demo;**

**public class DefaultConstructor**

**{**

**public DefaultConstructor ()**

**{**

**System.out.println ("This is a no-argument constructor");**

**}**

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

**{**

**new DefaultConstructor();**

**}**

**}**

**Copy Constructor in Java:**

A copy constructor is used for copying one object’s values to another.

**package Demo;**

**public class CopyConstructor**

**{**

**String web;**

**CopyConstructor (String w)**

**{**

**web = w;**

**}**

**/\* This is the Copy Constructor, it**

**\* copies the values of one object**

**\* to the another object (the object**

**\* that invokes this constructor)**

**\*/**

**CopyConstructor (CopyConstructor cc)**

**{**

**web = cc.web;**

**}**

**void disp ()**

**{**

**System.out.println ("Constructor: " + web);**

**}**

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

**{**

**CopyConstructor obj1 =new CopyConstructor ("Example of Copy Constructor in Java");**

**/\* Passing the object as an argument to the constructor**

**\* This will invoke the copy constructor**

**\*/**

**CopyConstructor obj2 = new CopyConstructor (obj1);**

**obj1.disp ();**

**obj2.disp ();**

**}**

**}**

**Calling a Constructor from another Constructor in Java using this Keyword**

In Java, calling a constructor from inside another constructor is possible. When you call a constructor from inside another constructor, you need to use this keyword to refer to the constructor.

this keyword can be very useful in the handling of Variable Hiding. We cannot create two instances/local variables with the same name. However, it is legal to create one instance variable & one local variable or method parameter with the same name. In this scenario, the local variable will hide the instance variable this is called Variable Hiding.

**package Demo;**

**public class VariabeHiding**

**{**

**int variable = 5;**

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

**{**

**VariabeHiding obj = new VariabeHiding ();**

**obj.method (20);**

**obj.method ();**

**}**

**void method (int variable)**

**{**

**variable = 10;**

**System.out.println ("Value of Instance variable: " + this.variable);**

**System.out.println ("Value of Local variable: " + variable);**

**}**

**void method ()**

**{**

**int variable = 40;**

**System.out.println ("Value of Instance variable: " + this.variable);**

**System.out.println ("Value of Local variable: " + variable);**

**}**

**}**