**Java Constructor Examples**

* Constructors are required to create objects for a class. Constructors are used to initialize the instance variables of an object.
* Constructor declaration looks like method declaration. It must have the same name as that of the class and have no return type.
* Constructors can be classified into two types, default constructors and parametarized constructors.
* If you don't define a constructor, then the compiler creates a default constructor. Default constructors do not contain any parameters. Default constructors are created only if there are no constructors defined by us.
* Prametarized constructors are required to pass parameters on creation of objects. We can overload constructors with different datatypes as its parameters.
* Use 'this()' to communicate from one constructor to another constructor in the same class.
* Use 'super()' to communicate with super class constructor.

**Java Constructor Sample Code Examples**

* [Default Constructor Example](http://java2novice.com/java_constructor_examples/default_constructor/)
* [Parametarized Constructor Example](http://java2novice.com/java_constructor_examples/constructor_with_parameters/)
* [Constructor Overloading Example](http://java2novice.com/java_constructor_examples/constructor_overloading/)
* [Constructor Chaining Example](http://java2novice.com/java_constructor_examples/constructor_chaining/)
* [SingleTon Example (Private Constructor)](http://java2novice.com/java_constructor_examples/singleton/)

**Java Default Constructor Examples**

* Default constructor refers to a constructor that is automatically created by compiler in the absence of explicit constructors.
* You can also call a constructor without parameters as default constructor because all of its class instance variables are set to default values.

**package com.myjava.constructors;**

**public class MyDefaultConstructor {**

**public MyDefaultConstructor(){**

**System.out.println("I am inside default constructor...");**

**}**

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

**MyDefaultConstructor mdc = new MyDefaultConstructor();**

**}}**

I am inside default constructor...

# Java Parameterized Constructor Examples

* Parameterized constructors are required to pass parameters on creation of objects.
* If we define only parameterized constructors, then we cannot create an object with default constructor. This is because compiler will not create default constructor. You need to create default constructor explicitly.

**package com.myjava.constructors;**

**public class MyParameterizedConstructor {**

**private String name;**

**public MyParameterizedConstructor(String str){**

**this.name = str;**

**System.out.println("I am inside parameterized constructor.");**

**System.out.println("The parameter value is: "+str);**

**}**

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

**MyParameterizedConstructor mpc = new MyParameterizedConstructor("Madhu Raj");**

**}}**

I am inside parameterized constructor.

The parameter value is: Madhu Raj

# Java Constructor Overloading Examples

Like method overloading we can overload constructors also. Along with default constructor, we can have constructors with parameters. The no of parameters can be same, and it can have different datatypes. Below example gives sample code for constructors overloading.

**package com.myjava.constructors;**

**public class MyOverloading {**

**public MyOverloading(){**

**System.out.println("Inside default constructor");**

**}**

**public MyOverloading(int i){**

**System.out.println("Inside single parameter constructor with int value");**

**}**

**public MyOverloading(String str){**

**System.out.println("Inside single parameter constructor with String object");**

**}**

**public MyOverloading(int i, int j){**

**System.out.println("Inside double parameter constructor");**

**}**

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

**MyOverloading mco = new MyOverloading();**

**MyOverloading spmco = new MyOverloading(10);**

**MyOverloading dpmco = new MyOverloading(10,20);**

**MyOverloading dpmco = new MyOverloading("java2novice");**

**}}**

Inside default constructor

Inside single parameter constructor with int value

Inside double parameter constructor

Inside single parameter constructor with String object

# Java Constructor Chaining Examples

Calling another constructor in the same class from another constructor is called constructor chaining. By using this() we can call another constructor in the same class. Incase we want to call another constructor, this() should be the first line in the constructor. Below example shows code for constructor chaining.

**package com.myjava.constructors;**

**public class MyChaining {**

**public MyChaining(){**

**System.out.println("In default constructor...");**

**}**

**public MyChaining(int i){**

**this();**

**System.out.println("In single parameter constructor...");**

**}**

**public MyChaining(int i,int j){**

**this(j);**

**System.out.println("In double parameter constructor...");**

**}**

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

**MyChaining ch = new MyChaining(10,20);**

**}**

**}**

In default constructor...

In single parameter constructor...

In double parameter constructor..

# Java Singleton Class Example Using Private Constructor

* We can make constructor as private. So that We can not create an object outside of the class.
* This property is useful to create singleton class in java.
* Singleton pattern helps us to keep only one instance of a class at any time.
* **The purpose of singleton is to control object creation by keeping private constructor.**

**package com.myjava.constructors;**

**public class MySingleTon {**

**private static MySingleTon myObj;**

**/\*\***

**\* Create private constructor**

**\*/**

**private MySingleTon(){**

**}**

**/\*\***

**\* Create a static method to get instance.**

**\*/**

**public static MySingleTon getInstance(){**

**if(myObj == null){**

**myObj = new MySingleTon();**

**}**

**return myObj;**

**}**

**public void getSomeThing(){**

**// do something here**

**System.out.println("I am here....");**

**}**

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

**MySingleTon st = MySingleTon.getInstance();**

**st.getSomeThing();**

**}**

**}**

**--------------------------------------x-----------------------------**