# Constructors in Java – A complete study!!

Constructor is a block of code, which runs when you use **new** keyword in order to**instantiate an object***. It looks like a method, however it is not a method. Methods have return type but constructors don’t have any return type.*

How to call a constructor?The constructor gets called when we create an object of a class (i.e. **new keyword** followed by class name). For e.g. Demo obj =  new Demo(); (here Demo() is a default constructor of Demo class).

**How does it look?**

**Default constructor:**It is also known as no-arg constructor. Constructor with no arguments is known as default constructor.

class Demo

{

public Demo()

{

System.out.println("This is a default constructor");

}

}

**Parameterized constructor**: Constructor with argument list is known as parameterized constructor.

class Demo

{

public Demo(int num, String str)

{

System.out.println("This is a parameterized constructor");

}

}

**Object creation using constructor –**

Lets say class name is “Demo”

1) I’m declaring a **reference variable** of class Demo-

class Demo;

2) **Object creation** – Calling default constructor for creating the object of class Demo (new keyword followed by Demo() constructor)

new Demo();

3) Now, I’m assigning the object to the reference –

class Demo = new Demo();

**What if we don’t initialize a constructor?**

Before I explain this lets have a look at the below example –

class Example

{

public void demoMethod()

{

System.out.println("hello");

}

public static void main(String args[])

{

Example obj = new Example();

obj.demoMethod();

}

}

If you notice the code in public static void main then you may find that I am creating an object of class Example and in order to do that I have invoked the default constructor of class Example.**But where did I defined the constructor?**If you don’t declare the constructor, **compiler creates one for you**.

**If you understood the example then I may need to test your skills – Guess the output of below Java program**

class Example2

{

private int var;

public Example2()

{

//code for default one

var = 10;

}

public Example2(int num)

{

//code for parameterized one

var = num;

}

public int getValue()

{

return var;

}

public static void main(String args[])

{

Example2 obj2 = new Example2();

System.out.println("var is: "+obj2.getValue());

}

}

Output:

var is: 10

Now replace the code of public static void main with the below code and guess answer –

Example2 obj2 = new Example2(77);

System.out.println("var is: "+obj2.getValue());

**Output**:

var is: 77

**Explanation**: you must be wondering why the answer is 77- Let me explain why is it so – Observe that while creating object I have passed 77 as a parameter inside parenthesis, that’s why instead of default one, parameterized constructor with integer argument gets invoked, where I have assigned the argument value to variable var.

Again, replace the code with the below code and try to find the answer –

Example2 obj3 = new Example2();

Example2 obj2 = new Example2(77);

System.out.println("var is: "+obj3.getValue());

Output:

var is: 10

**Most Important Point to Note: If I don’t define the constructor within the class, will compiler declare one for me every time?**

Answer is: No **Why?**Consider the below example –

class Example3

{

private int var;

public Example3(int num)

{

var=num;

}

public int getValue()

{

return var;

}

public static void main(String args[])

{

Example3 myobj = new Example3();

System.out.println("value of var is: "+myobj.getValue());

}

}

**Output**: It will throw a **compilation error!!. The reason is** when you don’t define any constructor in your class, compiler defines default one for you, however whenever you declare any constructor (in above example I have already defined a parameterized constructor), compiler doesn’t do it for you. Since I have defined a constructor in above code, compiler didn’t create default one. While creating object I am invoking default one, which doesn’t exist in above code. The code gives a compilation error.

**Constructor Chaining**

Constructor chaining is nothing but a scenario where in one constructor calls the constructor of its super class implicitly or explicitly. Suppose there is a class which inherits another class, in this case if you create the object of child class then first super class(or parent class) constructor will be invoked and then child class constructor will be invoked.

Have a look at the below example –

class Human

{

String s1, s2;

public Human()

{

s1 ="Super class";

s2 ="Parent class";

}

public Human(String str)

{

s1= str;

s2= str;

}

}

class Boy extends Human

{

public Boy()

{

s2 ="Child class";

}

public void disp()

{

System.out.println("String 1 is: "+s1);

System.out.println("String 2 is: "+s2);

}

public static void main(String args[])

{

Boy obj = new Boy();

obj.disp();

}

}

Output:

String 1 is: Super class

String 2 is: Child class

**Explanation of above example:**  
Human is a super class of Boy class. In above program I have created an object of Boy class, As per the rule super class constructor (Human()) invoked first which set the s1 & s2 value, later child class constructor(Boy()) gets invoked, which overridden s2 value.

**Note:**Whenever child class constructor gets invoked it implicitly invokes the constructor of parent class. In simple terms you can say that compiler puts a**super(); statement**in the child class constructor.

**Question you may ask**: In the above example no-arg constructor of super class invoked, I want to invoke arg constructor(Parameterized).

Its so simple.

change the code of public Boy() to something like this –

Note: super() should be the first statement in the constructor.

public Boy()

{

super("calling super one");

s2 ="Child class";

}

**Understood till now – Here are few more points about constructors**

1. Every class has a constructor whether it’s normal one or an abstract class.
2. As stated above, constructor are not methods and they don’t have any return type.
3. Constructor name and class name should be the same.
4. Constructor can use any access specifier, they can be declared as private also. Private constructors are possible in java but there scope is within the class only.
5. **Like constructors method can also have name same as class name, but still they have return type, though which we can identify them that they are methods not constructors.**
6. If you don’t define any constructor within the class, compiler will do it for you and it will create a no-arg (default) constructor for you.
7. **this() and super() should be the first statement in the constructor code.** If you don’t mention them, compiler does it for you accordingly.
8. Constructor overloading is possible but overriding is not possible. Which means we can have overloaded constructor in our class but we can’t override a constructor.
9. Constructors cannot be inherited.
10. If Super class doesn’t have a no-arg (default) constructor then compiler would not define a default one in child class as it does in normal scenario.
11. Interfaces do not have constructors.
12. Abstract can have constructors and these will get invoked when a class, which implements interface, gets instantiated. ( i.e. object creation of concrete class).
13. A constructor can also invoke another constructor of the same class – By using this(). If you wanna invoke a arg-constructor then give something like: **this(parameter list)**.

# Java – Constructor Chaining with example

Calling a constructor from the another constructor of same class is known as Constructor chaining. We will understand this with the help of below program.

## Example Program

In the below example the class “ChainingDemo” has 4 constructors and we are calling one constructor from another using **this() statement.**For e.g. in order to call a constructor with single string argument we have supplied a string in this() statement like **this(“hello”).**

**Note**: this() should always be the first statement in **constructor** otherwise you will get the below error message:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Constructor call must be the first statement in a constructor

**Complete Code:**

package beginnersbook.com;

public class ChainingDemo {

//default constructor of the class

public ChainingDemo(){

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

}

public ChainingDemo(String str){

this();

System.out.println("Parametrized constructor with single param");

}

public ChainingDemo(String str, int num){

//It will call the constructor with String argument

this("Hello");

System.out.println("Parametrized constructor with double args");

}

public ChainingDemo(int num1, int num2, int num3){

// It will call the constructor with (String, integer) arguments

this("Hello", 2);

System.out.println("Parametrized constructor with three args");

}

public static void main(String args[]){

//Creating an object using Constructor with 3 int arguments

ChainingDemo obj = new ChainingDemo(5,5,15);

}

}

Output:

Default constructor

Parametrized constructor with single param

Parametrized constructor with double args

Parametrized constructor with three args