**Java – parameterized constructor with example**

Earlier we discussed [**static constructor**](http://beginnersbook.com/2013/05/static-constructor/), [**private constructor**](http://beginnersbook.com/2013/12/java-private-constructor-example/), [**overloading of constructors**](http://beginnersbook.com/2013/05/constructor-overloading/) and [**chaining of constructors**](http://beginnersbook.com/2013/12/java-constructor-chaining-with-example/). In this post we will discuss**parameterized constructors** in Java with the help of few examples: A constructor with arguments is known as parameterized constructor.

Consider the below code: Here I have declared three constructors: one is default and two are parameterized. As you can see when we created the objects of class, these constructor got invoked, based on the parameters (or arguments) we provided during object creation.

class Example{

//Default constructor

Example(){

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

}

/\* Parameterized constructor with

\* two integer arguments

\*/

Example(int i, int j){

System.out.print("parameterized constructor");

System.out.println(" with Two parameters");

}

/\* Parameterized constructor with

\* three integer arguments

\*/

Example(int i, int j, int k){

System.out.print("parameterized constructor");

System.out.println(" with Three parameters");

}

public static void main(String args[]){

//This will invoke default constructor

Example obj = new Example();

/\* This will invoke the constructor

\* with two arguments

\*/

Example obj2 = new Example(12, 12);

/\* This will invoke the constructor

\* with three arguments

\*/

Example obj3 = new Example(1, 2, 13);

}

}

Output:

Default constructor

parameterized constructor with Two parameters

parameterized constructor with Three parameters

**Example 2**: A weird case of compilation error.  
In the last tutorial when we were discussing [**default constructor**](http://beginnersbook.com/2014/01/default-constructor-java-example/), we saw that when we do not create a default constructor for a class, compiler by default creates that for that class. However this is not always true. See the below example:

package beginnersbook.com;

class Example{

Example(int i, int j){

System.out.print("parameterized constructor");

}

Example(int i, int j, int k){

System.out.print("parameterized constructor");

}

public static void main(String args[]){

Example obj = new Example();

}

}

Output:

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

problem: The constructor Example() is undefined

The program threw compilation error because we didn’t declare the default constructor. However if you remove the two parameterized constructors from the above code then the program will run fine, even without declaring the default constructor. **The reason is**: When we do not declare **any constructor** in a class, the compiler creates default one for that class, but if we declare a constructor regardless of what it is **default** or **parameterized**, the compiler does not declare the default constructor itself.