**Java static constructor – Is it really Possible to have them in Java?**

Have you heard of **static constructor in Java**? I guess yes but the fact is that they are not allowed in Java. A constructor can not be static in Java.  Before I explain the reason let’s have a look at the below piece of code:

public class StaticTest

{

/\* See below - I have marked the constructor as static \*/

public static StaticTest()

{

System.out.println("Static Constructor of the class");

}

public static void main(String args[])

{

/\*Below: I'm trying to create an object of the class

which would intern call the constructor\*/

StaticTest obj = new StaticTest();

}

}

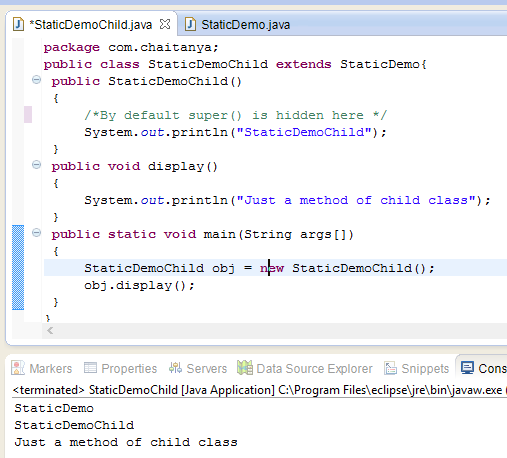
**Output:** You would get the below error message when you try to run the above java code.

**“modifier static not allowed here”**

**Why java doesn’t support static constructor?**

It’s actually pretty simple to understand – Everything that is marked static belongs to the class only, for example **static method cannot be inherited in the sub class because they belong to the class in which they have been declared. Refer**[**static keyword**](http://beginnersbook.com/2013/04/java-static-class-block-methods-variables/)**.**  
Lets back to constructors, Since each constructor is being called by its subclass during creation of the object of its subclass, so if you mark constructor as static the subclass will not be able to access the constructor of its parent class because it is marked static and thus belong to the class only. This will violate the whole purpose of inheritance concept and that is reason why a constructor cannot be static.

Let’s understand this with the help of an example –



public class StaticDemo

{

public StaticDemo()

{

/\*Constructor of this class\*/

System.out.println("StaticDemo");

}

}

public class StaticDemoChild extends StaticDemo

{

public StaticDemoChild()

{

/\*By default this() is hidden here \*/

System.out.println("StaticDemoChild");

}

public void display()

{

System.out.println("Just a method of child class");

}

public static void main(String args[])

{

StaticDemoChild obj = new StaticDemoChild();

obj.display();

}

}

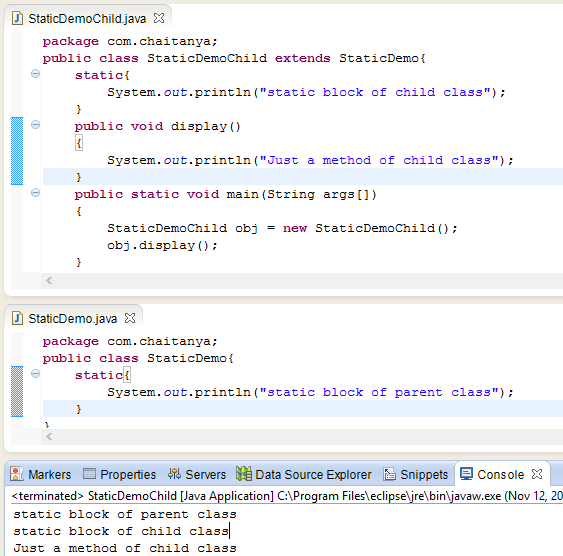
**Output:**  
StaticDemo  
StaticDemoChild  
Just a method of child class

**Did you notice?**We just created the object of child class and as a result it first called the constructor of parent class and then the constructor of it’s own class. It happened because the object creation calls constructor implicitly and every **child class constructor by default has super()** as first statement which calls it’s parent class’s constructor. The statement super() is used to call the parent class(base class) constructor.

Above explanation is the reason why constructor cannot be static – Because if we make them static they cannot be called from child class thus object of child class couldn’t be created.

**Static Constructor Alternative – Static Blocks**

Java has static blocks which can be treated as static constructor. Let’s consider the below program –



public class StaticDemo{

static{

System.out.println("static block of parent class");

}

}

public class StaticDemoChild extends StaticDemo{

static{

System.out.println("static block of child class");

}

public void display()

{

System.out.println("Just a method of child class");

}

public static void main(String args[])

{

StaticDemoChild obj = new StaticDemoChild();

obj.display();

}

}

**Output:**  
static block of parent class  
static block of child class  
Just a method of child class

In the above example we have used static blocks in both the classes which worked perfectly. We cannot use static constructor so it’s a good alternative if we want to perform a static task during object creation.