Static kintamieji ir metodai

Statinis blokas (1 iš 2)

- **Static block is mostly used for changing the default values of static variables.**
- This block gets executed when the class is loaded in the memory.
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

Statinis blokas (2 iš 2)

Static blocks are nothing but a normal block of code, enclosed in braces { }, preceded with static keyword.

These static blocks will be called when JVM loads the class into memory.

Incase a class has multiple static blocks across the class, then JVM combines all these blocks as a single block of code and executes it.

Static blocks will be called only once, when it is loaded into memory.

These are also called initialization blocks.

```java
import java.util.ArrayList;
import java.util.List;

public class MyStaticBlock {

    private static List<String> list;

    static{
        //created required instances
        //create ur in-memory objects here
        list = new ArrayList<String>();
        list.add("one");
        list.add("two");
        list.add("three");
        list.add("four");
    }

    public void testList(){
        System.out.println(list);
    }

    public static void main(String a[]){
        MyStaticBlock msb = new MyStaticBlock();
        msb.testList();
    }
}
```

Description: Example for static block vs constructor.

- Java static blocks will be called when JVM loads the class into memory, means it will be called only once.
- But constructor will be called every time when you create an object. The below example can give you an idea about execution.

Program: Example for static block vs constructor.

```java
public class MyStaticVsConstructor {

    static {
        System.out.println("I am in static block");
        System.out.println("Static block will be called first than constructor!!!
        System.out.println("Static block will be called only once.");
    }

    public MyStaticVsConstructor(){
        System.out.println("I am in constructor");
    }

    public static void main(String a[]){
        MyStaticVsConstructor ksv1 = new MyStaticVsConstructor();
        MyStaticVsConstructor ksv2 = new MyStaticVsConstructor();
        MyStaticVsConstructor ksv3 = new MyStaticVsConstructor();
        MyStaticVsConstructor ksv4 = new MyStaticVsConstructor();
    }
}
```

Program: Example for static block vs constructor. Output

Output:

```
I am in static block Static block
will be called first than constructor!!!
Static block will be called only once.
I am in constructor
I am in constructor
I am in constructor
I am in constructor
```

- http://beginnersbook.com/2013/04/java-static-class-block-methods-variables/

- http://stackoverflow.com/questions/7486012/static-classes-in-java

Program: Example for singleton class using static block.

- Since static blocks will be called only once, we can use static blocks to develop **singleton class.**
- Below example shows how to create singleton classes using static block. To create singleton class, make constructor as private, so that you cannot create object outside of the class. Create a private static variable of same class type, so that created object will be pointed to this reference. Now create static block, and create object inside static block. Since static block will be called only once, the object will be created only once.

```java
public class MyStaticSingleton {

    public static void main(String a[]){
        MySingleton ms = MySingleton.getInstance();
        ms.testSingleton();
    }
}
class MySingleton{

    private static MySingleton instance;

    static{
        instance = new MySingleton();
    }
    private MySingleton(){
        System.out.println("Creating MySingleton object...");
    }
    public static MySingleton getInstance(){
        return instance;
    }
    public void testSingleton(){
        System.out.println("Hey.... Instance got created...");
    }
}
```

```
Creating MySingleton object...
Hey.... Instance got created...
```

Statiniai kintamieji

- Static variables are also known as Class Variables.
- Such variables get default values based on the data type.
- Data stored in static variables is common for all the objects(or instances) of that Class.
- **Memory allocation for such variables only happens once when the class is loaded in the memory.**
- These variables can be accessed in any other class using class name.
- Unlike **non-static variables**, such variables can be accessed directly in static and non-static methods.

Statiniai metodai

- Static Methods can access class variables without using object of the class.
- **It can access non-static methods and non-static variables by using objects.**
- Static methods can be accessed directly in static and non-static methods.

Example 1: public static void main itself is a static method

```
class Example5{
static int i;
static String s;

public static void main(String args[]) //Its a Static Method {

Example5 obj=new Example5(); //Non Static variables accessed
using object obj System.out.println("i:"+obj.i);

System.out.println("s:"+obj.s); } }
```

Output:

```
i:0 s:null
```

- In general, any class from same package can be called without importing it. Incase, if the class is not part of the same package, we need to provide the import statement to access the class. We can access any static fields or methods with reference to the class name. Here comes the use of static imports. Static imports allow us to import all static fields and methods into a class and you can access them without class name reference.

  The syntax for static imports are given below:

```
// To access all static members of a class
import static package-name.class-name.*;
//To access specific static variable of a class
import static package-name.class-name.static-variable;
//To access specific static method of a class
import static package-name.class-name.static-method;
```

Static Import Example:

- Here we have two classes. The class MyStaticMembClass has one static variable and one static method. And anther class **MyStaticImportExmp** imports these two static members.

```java
package
com.java2novice.stat.imp.pac1;

public class MyStaticMembClass {

    public static final int
INCREMENT = 2;

    public static int
incrementNumber(int number){
        return number+INCREMENT;
    }
}
```

```java
package com.java2novice.stat.imp.pac2;

import static
com.java2novice.stat.imp.pac1.MyStaticMembClass.*;

public class MyStaticImportExmp {

    public static void main(String a[]){
        System.out.println("Increment value:
"+INCREMENT);
        int count = 10;
        System.out.println("Increment count:
"+incrementNumber(count));
        System.out.println("Increment count:
"+incrementNumber(count));
    }
}
```

Output:

```
Increment value: 2
Increment count: 12
Increment count: 12
```

JAVA NESTED (INNER) CLASSES EXAMPLES

- If a class is declared within another class or interface is called nested class.
- Types of nested classes are as given below:
- ***Static Member Classes***: It is defined as static member in a class or an interface.
- ***Non-static Member Classes***: It is defined as instance members of another classes.
- ***Local Classes***: It is defined in a block, like inside a method body or a local block.
- ***Anonymous Classes***: It can be defined as expressions and instantiated on the fly.

A static member class must me declared as a static member of top level class.
Static member class can not have its name as top level class.

```java
public class MyBasicStaticMemberClass {

    public static class MyStaticMemberExampleClass {

        public void printStatus() {
            System.out.println("Hey I am inside static member class");
        }
    }

    public static void main(String a[]) {
        StaticMemberTestClass smt = new StaticMemberTestClass();
        smt.testMemberClass();
    }
}

class StaticMemberTestClass {

    public void testMemberClass() {
        MyBasicStaticMemberClass.MyStaticMemberExampleClass msme
                    = new MyBasicStaticMemberClass.MyStaticMemberExampleClass();
        msme.printStatus();
    }
}
```

Output:

```
Hey I am inside static member class
```

A static member interface must me declared as a static member of top level class. Static member interface can not have its name as top level class.

```
public class MyStaticMemberInterfaceTest
            implements MyStaticMemberInterface.MyStaticInterface{

    public void implementMe(){
        System.out.println("Hey I have implemented successfully!!!");
    }

    public static void main(String a[]){
        MyStaticMemberInterfaceTest msi = new MyStaticMemberInterfaceTest();
        msi.implementMe();
    }
}

class MyStaticMemberInterface{

    public static interface MyStaticInterface{
        public void implementMe();
    }
}
```

Since static code does not have a this reference, you can not access instance variable of a top level class. You can only access static members of a top level class.

```
public class MyStatMemClassAcsVars {

    public static String staticVar = "You can access me!!!";
    private String privVar = "You cannot access me";

    public static class ChildClass{

        public void myMethod(){
            //you can access all static members of
            //top level class
            System.out.println(staticVar);
            //you cannot access instance members of
            //top level class
            //below line gives compile error
            //System.out.println(privVar);
        }
    }

    public static void main(String a[]){
        MyStatMemClassAcsVars.ChildClass cc = new MyStatMemClassAcsVars.ChildClass();
        cc.myMethod();
```

You can create an instance for non-static member class by creating instance for top level class.

```java
public class MyBasicNonStaticMemberClass {

    public class ChildClass{

        public void myMethod(){
            System.out.println("Hey you have called me!!!");
        }
    }

    public ChildClass getInnerInstance(){
        return this.new ChildClass();
    }

    public static void main(String a[]){
        MyBasicNonStaticMemberClass mbn = new MyBasicNonStaticMemberClass();
        ChildClass cc = mbn.getInnerInstance();
        cc.myMethod();
    }
}
```

You can create an instance for non-static member class by creating instance for top level class.

```
public class MyNonStaticMemberClassInstance {

    public static void main(String a[]){
        ParentClass pc = new ParentClass();
        ParentClass.ChildClass cc = pc.getInnerInstance();
        cc.myMethod();
        ParentClass.ChildClass cc1 = pc.new ChildClass();
        cc1.myMethod();
    }
}
class ParentClass{

    public class ChildClass{

        public void myMethod(){
            System.out.println("Hey you have called me!!!");
        }
    }

    public ChildClass getInnerInstance(){
        return this.new ChildClass();
    }
```