[Home](#)[All Tutorials](#)[Core Java](#)[OOPs](#)[Collections](#)[Java I/O](#)[JSON](#)[DBMS](#)

# Class, Block, Methods and Variables

ORDER: **OOPS CONCEPT**

used with class, variable, method and block. Static members belong to the class instance, this means if you make a member static, you can access it without creating an object. Let's take an example to understand this:

For example, for a method `myMethod()`, we can call this method without any object because it is static. If we remove the static keyword and make it non-static, we need to create an object of the class in order to call it.

Static members are common for all the instances(objects) of the class but non-static members are unique to each instance of class.

Example:

*method*

`()`

```
ln("myMethod");
```

```
ain(String[] args)
```

*that we are calling this method without creating any object.*

initializing the static variables. This block gets executed when the class is loaded. A class can have multiple Static blocks, which will execute in the same order as they have been written into the program.

## static block

Before the static variables were initialized before we accessed them in the main

```
keyword in Java";
```

```
in(String args[])
```

```
("Value of num: "+num);
```

```
("Value of mystr: "+mystr);
```

keyword **in Java**

## Static blocks

static blocks work in Java. They execute in the given order which means the first static block executes before the second static block. That's the reason, values initialized by the first static block are available to the second block.

```
("Static Block 1");
```

```
("Static Block 2");
```

```
main(String args[])
```

```
    ("Value of num: "+num);
```

```
    ("Value of mystr: "+mystr);
```

## Variables

Static variables are shared by all the instances (or objects) of the class because it is a class level variable. In other words, you can say that only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded into memory.

Static variables are also known as Class Variables.

**variables**, such variables can be accessed directly in static and non-static

## variables can be accessed directly in Static method

ethod disp() and two static variables var1 and var2. Both the variables are static method.

*hod*

```
("Var1 is: "+var1);  
("Var2 is: "+var2);
```

```
n(String args[])
```

## variables are shared among all the instances of class

variable is non-static and integer variable is Static. As you can see in the static variable is different for both the objects but the static variable is shared reason the changes made to the static variable by object ob2 reflects in both

*able*

*variable*

```
in(String args[])
```

```
= new JavaExample();
```

```
= new JavaExample();
```

*les can be accessed directly without  
. Just to demonstrate that static variables  
am accessing them using objects so that  
that the changes made to static variables  
, reflects when we access them using other*

*value to static variable using object ob1*

```
ject1";
```

*rwrite the value of var1 because var1 has a single  
mong both the objects.*

```
ject2";
```

```
ln("ob1 integer:"+ob1.var1);
```

```
ln("ob1 String:"+ob1.var2);
```

```
ln("ob2 integer:"+ob2.var1);
```

```
ln("ob2 STring:"+ob2.var2);
```

r: **Java – static variable**

## Methods

Access class variables(static variables) without using object(instance) of the class. Static methods and non-static variables can only be accessed using objects. Static variables can be accessed directly in static and non-static methods.

Accessed by return type, followed by method name.

```
getName();
```

## method main is accessing static variables without object

```
ginnnersbook";  
thod  
in(String args[])  
  
n("i:"+i);  
n("s:"+s);
```

## method accessed directly in static and non-static method

```
innersbook";
```

```
"i:"+i);  
"i:"+s);
```

*Lled in non-static method*

```
n(String args[])
```

```
j = new JavaExample();
```

*have object to call this non-static method*

*Lled in another static method*

## Static vs non-static Method in Java

Static only if it is a nested class.

Static doesn't need reference of Outer class  
Static cannot access non-static members of the Outer class

Points with the help of an example:

Example

```
String str = "BeginnersBook";
```

```
class StaticClass{  
    static void  
    () {
```

*If you try to access the str variable of outer class  
then you will get compilation error  
because nested static class cannot access non-  
static members of the outer class.*

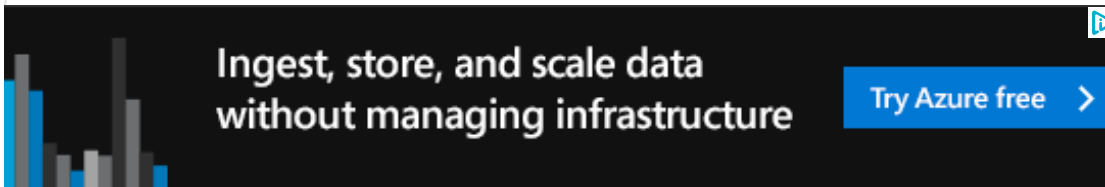
```
        System.out.println(str);
```

```
    public static void main(String args[])
```

*ance of nested class we didn't need the outer  
but for a regular nested class you would need  
instance of outer class first*

```
stedClass obj = new JavaExample.MyNestedClass();
```

Next >



ays

0, 2016 AT 4:43 AM

planation of these all topic .....so lots of thank u.....



2016 AT 5:18 PM

ur work has been really helpful for my interview. Simple yet thorough.

---

**says**

016 AT 5:47 AM

a lot of tutorials/lectures but this is so easy and comprehensible. Thank

---

**Biswal says**

2016 AT 6:05 AM

tutorial.

when we can access the class variable via class also why we need to  
t to access it. can you please explain in details.

---

**says**

Y 8, 2017 AT 4:49 AM

ing all your tutorials and they are actually a beginner's guide for their

mention a correction in the above post:

nder static methods, you have created static variables and then using  
access them. I think the commented line should instead be "Static  
sed without using class object" and following lines should be:

```
tlN(Example5.i);
```

```
tlN(Example5.s);
```

**apon says**

4, 2017 AT 9:37 AM

static block example why could not show the first static block's value?

**1 says**

2017 AT 1:53 AM






aded by JVM then static block executed or class been called in any  
tatic block executed ?

will not be published. Required fields are marked \*

---

## Java Tutorial

---

-  [Java Index](#)
-  [Java Introduction](#)
-  [JVM - Java Virtual Machine](#)
-  [First Java Program](#)
-  [Variables](#)

[➡ Data Types](#)[➡ Operators](#)

---

## Java Control Statements

[➡ Java If-else](#)[➡ Java Switch-Case](#)[➡ Java For loop](#)[➡ Java while loop](#)[➡ Java do-while loop](#)[➡ Continue statement](#)[➡ break statement](#)

---

## OOPs Concepts

[➡ OOPs Concepts](#)[➡ Constructor](#)[➡ Static keyword](#)[➡ Inheritance](#)[➡ Types of inheritance](#)[➡ Aggregation](#)[➡ Association](#)[➡ Super Keyword](#)[➡ Method overloading](#)[➡ Method overriding](#)[➡ Overloading vs  
Overriding](#)[➡ Polymorphism](#)[➡ Types of  
polymorphism](#)[➡ Static and dynamic  
binding](#)

➡ Abstract class and methods

➡ Interface

➡ Abstract class vs interface

➡ Encapsulation

➡ Packages

➡ Access modifiers

➡ Garbage Collection

➡ Inner classes

➡ Static import

➡ Static constructor

➡ Java Interview Q

---

### MORE ...

➡ Java 8 Features

➡ Java 9 Features

➡ Java Conversion

➡ Java String

➡ Exception handling

➡ Java Multithreading

➡ Java I/O

➡ Java Serialization

➡ Java Regex

➡ Java AWT

➡ Java Swing

➡ Java Enum

➡ Java Annotations

