**[Where are static methods and static variables stored in Java?](https://stackoverflow.com/questions/8387989/where-are-static-methods-and-static-variables-stored-in-java)**

Static methods (in fact all methods) as well as static variables are stored in the PermGen section of the heap; since they are part of the reflection data (class related data, not instance related).

Update for clarification:

Note that only the variables and their technical values (primitives or references) are stored in PermGen space.

If your static variable is a reference to an object that objects itself is stored in the normal sections of the heap (young/old generation or survivor space). Those objects (unless they are interal objects like classes etc.) are not stored in PermGen space.

**static** **int** *i* = 1; //the value 1 is stored in the permgen section

**static** Object *o* = **new** SomeObject();

//the reference (pointer/memory address) is stored in the permgen section, the object itself is not

**Java static variable**

If we declare a variable with “**static**” keyword, then it is called as **static variable**. For example

**static int y=0;**

All the instance of the class share the same copy of the variable, a static variable can be accessed directly by calling **“<<ClassName>>.<<VariableName>>”**without need to create instance for the class.

The **static** keyword can be applied to **variables, methods, blocks and nested class**.

<https://www.javainterviewpoint.com/use-of-static-keyword-in-java/>

## ****Java static block****

The static block, is a block of code inside a [**Java**](https://www.javainterviewpoint.com/category/core-java/) class that will be executed when a class is first loaded in to the JVM. Mostly the static block will be used for initializing the variables.

**How System.out.println() method works ?**

* **‘out’** is a static final reference of the **PrintStream** class declared in the **System** class.
* **‘out’** is initialized through the **setOut()** method which is called inisde **initializeSystemClass()** method of the **System** class.
* Finally about **println()**, it is the method which is declared inside the **PrintStream** class

**public** **class** PrintStream **extends** FilterOutputStream {

// out object is inherited from FilterOutputStream class

**public** **void** println() {

}

}

Overloading – Same Class, Same Method Name, Different Parameters

Overriding    – Different Class, Same Method Name, Same Parameters.

# Difference between equals() and ==

Both **equals()** and **‘==’** is used to check the equality of the objects but there is a significant amount of difference between **equals()** and **‘==’**.

The equals method is present in the **java.lang.Object** class and it is used to check the equivalence of the object (i.,e) to check if the content is equal whereas **‘==’** is used to check if the actual object instances are same or not.

<https://www.guru99.com/java-stack-heap.html>

**What is Stack Memory?**

Stack in java is a section of memory which contains methods, local variables, and reference variables. Stack memory is always referenced in Last-In-First-Out order. Local variables are created in the stack.

**What is Heap Memory?**

Heap is a section of memory which contains Objects and may also contain reference variables. Instance variables are created in the heap

**Summary:**

* When a method is called, a frame is created on the top of the stack.
* Once a method has completed execution, the flow of control returns to the calling method and its corresponding stack frame is flushed.
* Local variables are created in the stack
* Instance variables are created in the heap & are part of the object they belong to.
* Reference variables are created in the stack