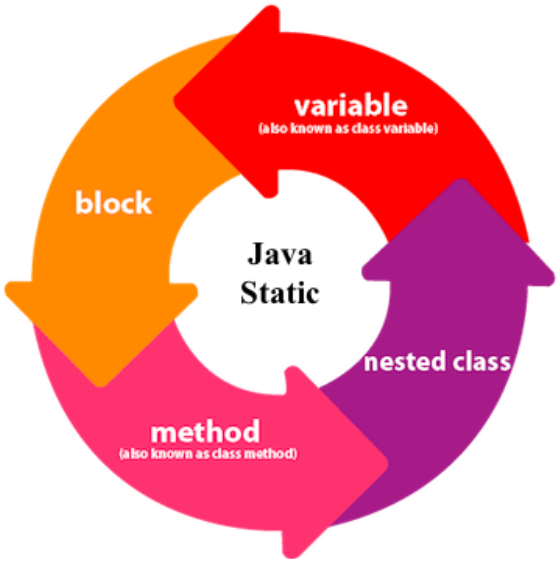
Static block

Static block is used for **initializing the static variables**. Although *static* variables **can be initialized directly during declaration**, there are situations when we're required to do the multiline processing.

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.



Java static variables

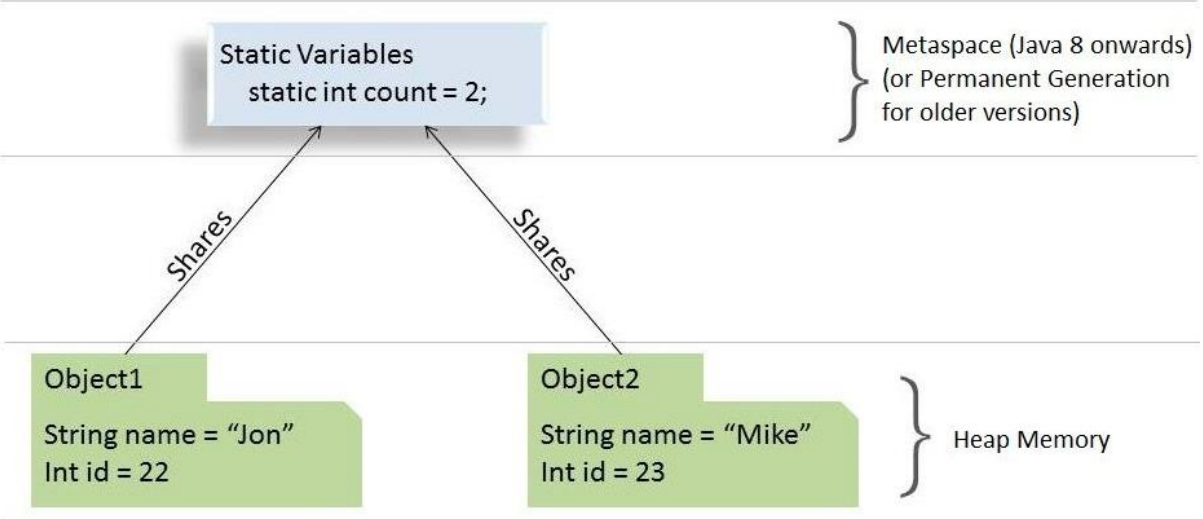
A static variable is common to all the instances (or objects) of the class because it is a class level variable. In other words you can say that only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded in the memory.
Few Important Points:

- Static variables are also known as Class Variables/static fields.
- Unlike **non-static variables**, such variables can be accessed directly in static and non-static methods.

Let's see what static variables can do...

1. Static variables can be **accessed directly in Static method**.
2. Static variables are **shared among all the instances of class.** Whereas **non-static variables cannot be accessed by static but can only be accessed by non-static**. That is because **non-static/instance variable would have not got allocated in the memory**

In Java, **if a field is declared *static*, then exactly a single copy of that field is created and shared among all instances of that class**. It doesn't matter how many times we initialize a class; there will always be only one copy of *static* field belonging to it. The value of this *static* field will be shared across all object of either same or any different class.



From the memory perspective, **static variables go in a particular pool in JVM memory called Metaspace** (before Java 8, this pool was called Permanent Generation or PermGen, which was completely removed and replaced with Metaspace).

Where to use static variables????

- When the value of variable is independent of objects.
- The static variable can be used to refer to the **common property of all objects** (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- If ***static*** variables require additional, multi-statement logic while initialization, then a ***static*** block can be used.

If you are thinking is there any **advantage** of using **static variables** then yes there is: It makes your program **memory efficient** (i.e., it saves memory).

How?? Well this is how…

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all objects. If we make it static, this field will get the memory only once.

Key points to remember..

- Since *static* variables belong to a class, they can be **accessed directly using class name** and **don't need any object reference.**
- *static* variables can only be **declared at the class level.**
- *static* fields can be **accessed without object initialization.**
- Although we can access *static* fields or the class variables using an object reference, we should refrain from using it as in this case it becomes difficult to figure whether it's an instance variable or a class variable; instead, we should always refer to *static* variables using class.
- If initialization of *static* variables requires some additional logic except the assignment
- If the initialization of static variables is error-prone and requires exception handling