Static keyword in Java

static keyword can be used with

- Variable (then called class variable)
- Methods (then called class method)
- Class (only in inner class)

static is a keyword which doesn't requires you to create an object because they are access before object creation.

You can access ***static variable*** **(class variable)** and ***static method*** **(class method)** without an object
e.g. - **A.c** ("class name dot operator variable name")

You can't use "***this***" and "***super***" keyword with static in same class.

```
class A
{
     int a=150,b;              //Instance Variable
     static int c=100;         // Class Variable

     void input()
     {
          System.out.println("This is the example of Static Method ");
     }
}

class B extends A
{
     static void disp()
     {
          super.input();       //can't use 'super' and 'this' keyword here
because this method is static method/class method

          System.out.println("Value of C = " +super.a);  //prohibited or
          illegal reason just above
     }
}
```

- **Can't declare constructor static**

Class Variable - A common value which is shared by all the instances of class and if one object changes the variable, it changes for all other object of that class.

Instance Variable - Variable which may have different value for each object of that class.

Gist/Summary

- ***static*** keyword converts a instance variable to class variable.
- ***static*** variable/method need not to be accessed by object because it is called before object creation. It means that it can be accessed without class object as well as with objects too.
- Accessing/calling ***static*** variable/method of class without object i.e. via "**class name dot operator variable name**" e.g.- **A.c or A.input()**
- Can't use '**super**' and '**this**' keyword in static method
- ***static*** keyword, if used with class then only in inner nested class. Outer class must not be static in any case.

Static Block Concept

What is Static Block in Java?

The static block is a block of statement inside a Java class that will be executed when a class is first loaded into the JVM.

```
class Test
{
	static
	{
		//Code goes here
	}
}
```

Why is it needed/used ?

A static block helps to initialize the static data members, just like constructors help to initialize instance members.

Important Points

- This code inside static block is executed only once: the first time the class is loaded into memory.
- ***static*** blocks are executed before constructors.
- A class can have any number of static initialization blocks, and they can appear anywhere in the class body.
- The runtime system guarantees that static initialization blocks are called in the order that they appear in the source code.

Reference Link

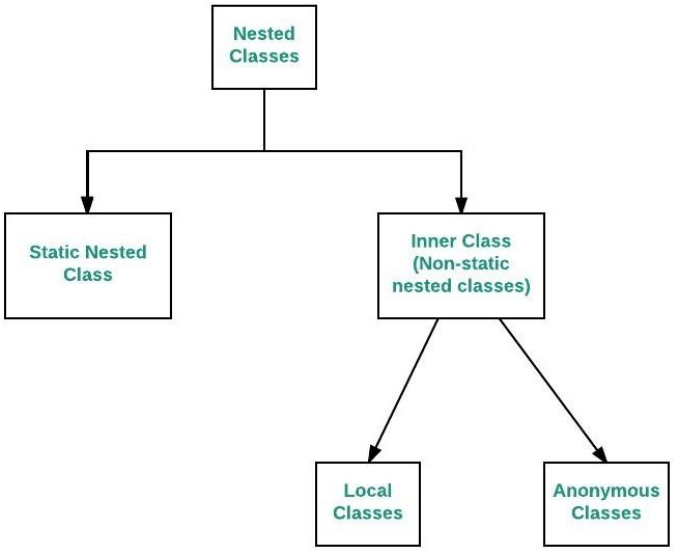
https://www.geeksforgeeks.org/g-fact-79/

https://www.guru99.com/java-static-variable-methods.html

NOTE : Actual java code (in **"PDF"** and **".java"** file) along with its output is in the same folder named **Static_Main**

Nested Class in Java

Nested class - Class within a class is called nested classes.
Outer Class - The class in which nested class exists called 'outer class'.
Inner class - The nested class present inside outer class called 'inner class'.



Example

```
class OuterClass
{
      class Inner1
      {
            class Inner2
            {
                  ...
            }
      }
}
```

NOTE

- Outer class must not be static.
- Inner class can be static i.e. known as static nested class.
- Method, interface and loops can also be nested.

Types of Nested Class (Refer Diagram) - Object Creation - Both have different way of object creation.

1) **Static Nested Class -** Class within a class with static keyword.

Example

```
class OuterClass
{
      static class InnerClass
      {

      }
}

class Static_Nested_class
{
      public static void main(String args[])
      {
            OuterClass.InnerClass obj= new OuterClass.InnerClass();
      }
}
```

2) Non-static Nested Class (Inner class) - Class within a class without static keyword

Example

```
class OuterClass
{
     class InnerClass
     {

     }
}

class Non_Static_Nested_Class
{
     public static void main(String args[])
     {
          OuterClass o=new OuterClass();
          OuterClass.InnerClass obj= o.new InnerClass();

          obj.input();
     }
}
```

Why we use nested classes?

- To group, to use it as a one place.
- To increase encapsulation.
- Increase readability and maintainability as it is closer to where it is used.
- Code optimization
- Less memory usage

NOTE - Actual java code (in **"PDF"** and **".java"** file) along with its output is in the same folder named

Static_Nested_Class

Non- Static_Nested_Class

Reference Links

https://www.geeksforgeeks.org/nested-classes-java/