Lesson:

Static Keyword

PW SKILLS

List of Concepts Involved:

- static keyword
- Class loading and How Java program actually executes
- Different Members in the java program
- Static variables
- Static Methods
- Static block
- Difference with respect to static and non static members of a class

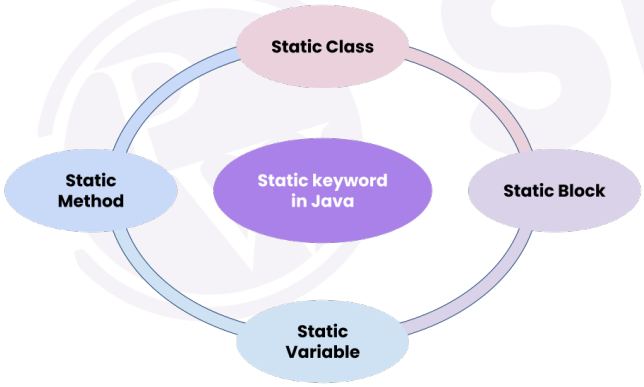
‘static’ keyword

The static keyword is mainly used for memory management in Java. A static keyword can be applied to variables, blocks, methods, and classes. The static keyword is a property of a class rather than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.

Where is the "static" keyword applicable?

The static keyword is a non-access modifier in Java and is applicable for the following:

1. Variables
2. Methods
3. Blocks
4. Class



Static Variables

If we declare any variable as static, then it is called a static variable. When a variable is declared as static, then a single copy of that variable is created and shared among all of the objects at the class level. Static variables are global variables. All instances of the class share the same static variable.
We can create static variables at the class level only.

Why static?

It makes our program more efficient, as every object doesn't allocate separate memory to a static variable.

Static Method

A static method is a method that belongs to a class rather than an instance of a class. This means you can call a static method without creating an object of the class. Static methods are sometimes called class methods.

There are a few other reasons why you might want to use static methods:

- You can access static methods from outside of the class in which they are defined. This is not possible with non-static methods.
- Subclasses can override static methods, but non-static methods cannot.
- Static methods are executed when an instance of the class is created, whereas non-static methods are not.
- Static methods can be used to create utility classes that contain general-purpose methods.

Static Blocks

It is used to initialize static data members. It is used to initialize before the main method at the time of class loading. It gets executed only once when the class gets loaded. It is not necessary to execute it again when creating different objects after the first time.

Static Class

In Java, a "static class" is a class that can be instantiated without having to create an instance of the containing class. A static class is defined as a member of another class and can only access static members of the containing class.

How Java Program Actually executes:

Class Loading

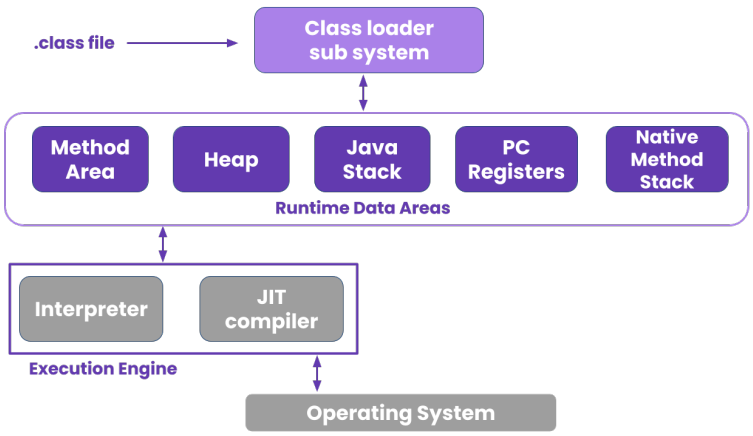
In Java, classloading is the process of loading class files into the JVM (Java Virtual Machine) at runtime. It is responsible for loading classes from various sources, such as the file system, network, and databases, and making them available to the JVM for execution.

The class loading process in Java is divided into three phases: loading, linking, and initialization.

1. Loading: In the loading phase, the classloader locates the class file using the fully qualified class name, reads the class file, and converts it into a Class object. The Class object contains the metadata of the class, such as the fields, methods, and constructors.

2. Linking: In the linking phase, the JVM performs several operations on the Class object, such as verifying the class file's integrity, resolving symbolic references, and allocating memory for the class variables.

3. Initialization: In the initialization phase, the JVM initializes the class variables with their default values, and runs the class's static initialization block (if any).



Principles of functionality of a Java classloader

A Java classloader's functionality is based on the following principles:

1. Delegation: A class loader delegates the responsibility of loading a class to its parent class loader before attempting to load it itself. This allows the class loader to take advantage of any classes that have already been loaded by the parent class loader.

2. Visibility: A class loaded by a class loader is only visible to that class loader and its child class loaders. This allows different class loaders to load different versions of the same class without interfering with each other.

3. Uniqueness: Each class is identified by its fully-qualified name (FQN) and is loaded by only one class loader. This ensures that the same class is not loaded multiple times by different class loaders.

4. Caching: A class loader caches the classes that it loads to improve performance. This allows the class loader to quickly return a class that has already been loaded, rather than having to load it again.

5. Security: A class loader enforces Java's security model by only allowing classes to access resources and execute code that is within their scope of permissions.

6. Extensibility: Java class loaders are extensible, allowing developers to create custom class loaders that can be used to load classes from specific locations or with specific behaviours.

7. Dynamic nature: Class loading is a dynamic process and can happen at runtime. Classes can be loaded, unloaded, and reloaded as the application runs.

8. Classpath: Java classloaders use the classpath to locate the classes that they need to load. The classpath is an ordered list of directories and JAR files that contain the class files.

9. Loading order: Java classloaders follow a specific order to load classes, first it checks the memory, if class is not found in memory it checks the cache, if class is not found in cache it checks the local file system, if class is not found in local file system it checks the network.

Different Members in the Java program

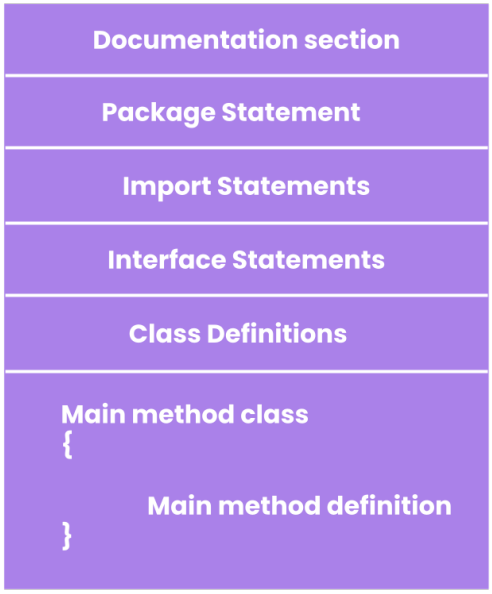
A Java program consists some of Members are:

1. Instance Member
2. Static Member

Instance Member: An instance member is essentially anything within a class that is not marked as static. That is, that it can only be used after an instance of the class has been made (with the new keyword). This is because instance members belong to the object, whereas static members belong to the class.

Static Member: Static members are those which belong to the class and you can access these members without instantiating the class. The static keyword can be used with methods, fields, classes (inner/nested), blocks.

Structure of Java Program:



Static variables

- If the value of a variable is not varied from object to object such type of variables is not recommended to be declared as instance variables.
- We have to declare such types of variables at class level by using static modifiers.
- In the case of instance variables for every object a separate copy will be created but in the case of static variables for the entire class only one copy will be created and shared by every object of that class.
- Static variables will be created at the time of class loading and destroyed at the time of class unloading hence the scope of the static variable is exactly the same as the scope of the .class file.
- Static variables will be stored in the method area. Static variables should be declared within the class directly but outside of any method or block or constructor.
- Static variables can be accessed from both instance and static areas directly. We can access static variables either by class name or by object reference but usage of class name is recommended.
- But within the same class it is not required to use class names we can access directly.

We can access static variables in 2 ways

1. Using className
2. Using reference variables

Note: Static variables also known as class level variables or fields.

Static methods

Methods which are available at the class level are referred to as “static methods”.
These methods are referred to as utility methods.
Inside the static methods we can access only static variables.
If we try to access the instance variables directly then it would result in **“CompileTimeError”.**

static block

- These are the blocks which gets executed automatically at the loading the .class files
- If we want to perform any activity at the time of loading a .class file we have to define that activity inside the static block.
- We can write any no of static blocks, those static blocks will be executed from top to bottom.
- Normally a static block is used to perform initialization of the static variables.

Difference with respect static and non static members of a class static

- These variables are called “class variables”.
- These variables will get memory in the method area.
- If the value does not change from object to object then we need to use static variables.
- Inside a static area we can access static variables only.
- Static variables are created using static keywords.

Non-static

- These variables are called “instance variables”.
- These variables will get memory in the heap area.
- If the value changes from object to object then we need to use “non-static” variables.
- Inside a nonstatic area we can access both static and non-static variables.
- Non-static variables are created without using the “static” keyword.