Nested Class

\*\*\*\*\*\*\*\*\*\*\*

Writing a class within another is allowed in Java.

-The class written within is called the nested class,

-and the class that holds the inner class is called the outer/enclosing class.

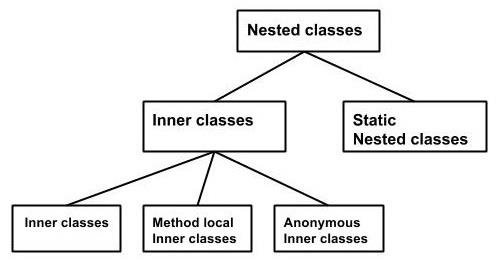
Why do we need it?

If a class is useful to only one other class then it seems logical to embed this class as a nested class. t is an way of logically grouping classes that are only used in one place

Nested classes are divided into two types −

1. Non-static nested classes (Inner Classes) − These are the non-static members of a class.
2. Static nested classes − These are the static members of a class.

Following are the types of Nested classes in Java –



**Non-static nested classes (Inner Classes)**

**Why do we need nested class:** Inner classes are a security mechanism in Java.

We know a class cannot be associated with the access modifier private, but if we have the class as a member of other class, then the inner class can be made private. **And this is also used to access the private members of a class.**

* An instance of inner class (non-static class) can exist only with in an instance of Outer class.
* We can get access private property through inner class

## Static Nested Classes

## is a nested class that which is declared as static that means it is a static member of the outer class.

## as it is static so It can be accessed without instantiating the outer class, like other static members. It can be accessed by outer class name.

## Just like static members, a static nested class does not have access to the instance variables and methods of the outer class (as we in static method cant use non-static variable)

* It can access static data members of the outer class, **including private**.
* If you have the static member inside the static nested class, you don't need to create an instance of the static nested class.

Java Anonymous inner class

-is an inner class without a name and for which only a single object is created for one anonymous class.

-if you have to override a method of class or interface. without having to actually subclass a class.

Java Anonymous inner class can be created in two ways:

1. Class (may be abstract or concrete).
2. Interface

* We can pass as an argument
* // interface
* interface Message {
* String greet();
* }
* public class My\_class {
* // method which accepts the object of interface Message
* public void displayMessage(Message m) {
* System.out.println(m.greet() +
* ", This is an example of anonymous inner class as an argument");
* }
* public static void main(String args[]) {
* // Instantiating the class
* My\_class obj = new My\_class();
* // Passing an anonymous inner class as an argument
* obj.displayMessage(new Message() {
* public String greet() {
* return "Hello";
* }
* });
* }
* }

Local inner class

A class which is created inside a method, is called local inner class in java.

* Local Inner Classes are the inner classes that are defined inside a block. Generally, this block is a **method body**.

Sometimes this block can be a **for loop**, or an **if clause**.

* Local Inner classes are not a member of any enclosing classes. They belong to the block they are defined within,

due to which local inner classes **cannot have any access modifiers** associated with them.

However, they can be marked as **final** or **abstract**. These classes have access to the fields of the class enclosing it.

* If you want to invoke the methods of the local inner class, you must instantiate this class inside the method

## Rules for Java Local Inner class

1. Local inner class cannot be invoked from outside the method.
2. Local inner class cannot access non-final local variable till JDK 1.7. Since JDK 1.8, it is possible to access the non-final local variable in the local inner class.