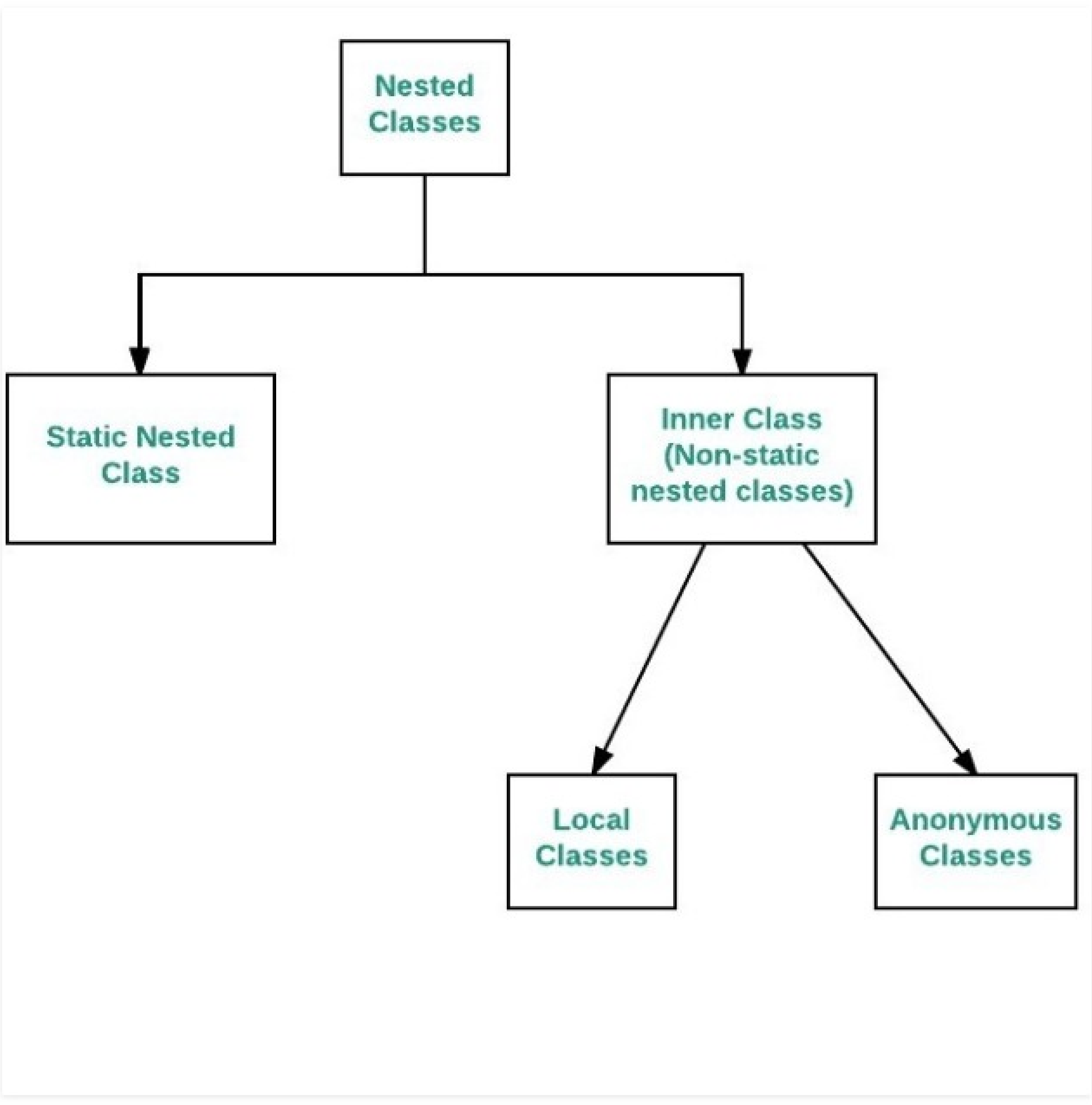
Nested Classes in Java

In java, it is possible to define a class within another class, such classes are known as *nested classes*. They enable you to logically group classes that are only used in one place, thus this increases the use of encapsulation, and create more readable and maintainable code.

- The scope of a nested class is bounded by the scope of its enclosing class. Thus in above example, class *NestedClass* does not exist independently of class *OuterClass*.
- A nested class has access to the members, including private members, of the class in which it is nested. However, reverse is not true i.e. the enclosing class does not have access to the members of the nested class.
- A nested class is also a member of its enclosing class.
- As a member of its enclosing class, a nested class can be declared *private*, *public*, *protected*, or *package private*(default).
- Nested classes are divided into two categories:
    1. **static nested class :** Nested classes that are declared *static* are called static nested classes.
    2. **inner class :** An inner class is a non-static nested class.

Syntax:

```
class OuterClass
{
...
    class NestedClass
    {
        ...
    }
}
```



Static nested classes

As with class methods and variables, a static nested class is associated with its outer class. And like static class methods, a static nested class cannot refer directly to instance variables or methods defined in its enclosing class: it can use them only through an object reference. They are accessed using the enclosing class name.

```
OuterClass.StaticNestedClass
```

For example, to create an object for the static nested class, use this syntax:

```
OuterClass.StaticNestedClass nestedObject =
     new OuterClass.StaticNestedClass();
```

```
// Java program to demonstrate accessing
// a static nested class

// outer class
class OuterClass
{
    // static member
    static int outer_x = 10;

    // instance(non-static) member
    int outer_y = 20;

    // private member
    private static int outer_private = 30;

    // static nested class
    static class StaticNestedClass
    {
        void display()
        {
            // can access static member of outer class
            System.out.println("outer_x = " + outer_x);

            // can access display private static member of outer class
            System.out.println("outer_private = " + outer_private);

            // The following statement will give compilation error
            // as static nested class cannot directly access non-static membera
            // System.out.println("outer_y = " + outer_y);

        }
    }
}

// Driver class
public class StaticNestedClassDemo
{
    public static void main(String[] args)
    {
        // accessing a static nested class
        OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

        nestedObject.display();

    }
}
```

Run on IDE

Output:

```
outer_x = 10
outer_private = 30
```

Inner classes

To instantiate an inner class, you must first instantiate the outer class. Then, create the inner object within the outer object with this syntax:

```
OuterClass.InnerClass innerObject = outerObject.new InnerClass();
```

There are two special kinds of inner classes :

1. Local inner classes
2. Anonymous inner classes

```
// Java program to demonstrate accessing
// a inner class

// outer class
class OuterClass
{
    // static member
    static int outer_x = 10;

    // instance(non-static) member
    int outer_y = 20;

    // private member
    private int outer_private = 30;

    // inner class
    class InnerClass
    {
        void display()
        {
            // can access static member of outer class
            System.out.println("outer_x = " + outer_x);

            // can also access non-static member of outer class
            System.out.println("outer_y = " + outer_y);

            // can also access private member of outer class
            System.out.println("outer_private = " + outer_private);

        }
    }
}

// Driver class
public class InnerClassDemo
{
    public static void main(String[] args)
    {
        // accessing an inner class
        OuterClass outerObject = new OuterClass();
        OuterClass.InnerClass innerObject = outerObject.new InnerClass();

        innerObject.display();

    }
}
```

Run on IDE

Output:

```
outer_x = 10
outer_y = 20
outer_private = 30
```

Difference between static and inner(non-static nested) classes

- Static nested classes do not directly have access to other members(non-static variables and methods) of the enclosing class because as it is static, it must access the non-static members of its enclosing class through an object. That is, it cannot refer to non-static members of its enclosing class directly. Because of this restriction, static nested classes are seldom used.
- Non-static nested classes (inner classes) has access to all members(static and non-static variables and methods, including private) of its outer class and may refer to them directly in the same way that other non-static members of the outer class do.