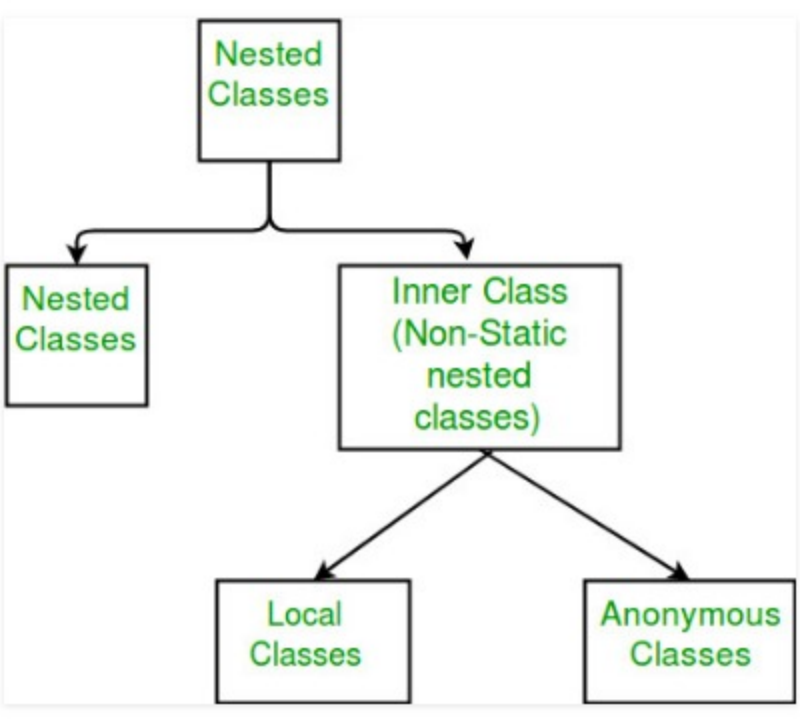
Local Inner Class in Java

Prerequisites :- Nested Classes 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 of which local inner classes cannot have any access modifiers associated with them. However, they can be marked as final or abstract. These class have access to the fields of the class enclosing it. Local inner class must be instantiated in the block they are defined in.

Rules of Local Inner Class:

1. The scope of local inner class is restricted to the *block* they are defined in.
2. Local inner class **cannot** be instantiated from outside the *block* where it is created in.
3. Till JDK 7,Local inner class can access only final local variable of the **enclosing block**. However From JDK 8, it is possible to access the non-final local variable of enclosing block in local inner class.
4. A local class has access to the members of its **enclosing class**.
5. Local inner classes can extend an abstract class or can also implement an interface.



Declaring a Local Inner class: A local inner class can be declared within a block. This block can be either a method body, initialization block, for loop or even an if statement.

Accessing Members: A local inner class has access to fields of the class enclosing it as well as the fields of the block that it is defined within. These classes, however, can access the variables or parameters of the block that encloses it only if they are declared as final or are effectively final. A variable whose value is not changed once initialized is called as effectively final variable. A local inner class defined inside a method body, have access to it's parameters.

What happens at compile time?

When a program containing a local inner class is compiled, the compiler generate two .class files, one for the outer class and the other for the inner class that has the reference to the outer class. The two files are named by compiler as:

- Outer.class
- Outer$1Inner.class

Declaration within a method body

```
// Java program to illustrate
// working of local inner classes
public class Outer
{
    private void getValue()
    {
        // Note that local variable(sum) must be final till JDK 7
        // hence this code will work only in JDK 8
        int sum = 20;

        // Local inner Class inside method
        class Inner
        {
            public int divisor;
            public int remainder;

            public Inner()
            {
                divisor = 4;
                remainder = sum%divisor;
            }
            private int getDivisor()
            {
                return divisor;
            }
            private int getRemainder()
            {
                return sum%divisor;
            }
            private int getQuotient()
            {
                System.out.println("Inside inner class");
                return sum / divisor;
            }
        }

        Inner inner = new Inner();
        System.out.println("Divisor = "+ inner.getDivisor());
        System.out.println("Remainder = " + inner.getRemainder());
        System.out.println("Quotient = " + inner.getQuotient());
    }

    public static void main(String[] args)
    {
        Outer outer = new Outer();
        outer.getValue();
    }
}
```

Run on IDE

Output:

```
Divisor = 4
Remainder = 0
Inside inner class
Quotient = 5
```

Note :- A local class can access local variables and parameters of the enclosing block that are *effectively final*. For example, if you add the highlighted assignment statement in the *Inner* class constructor or in any method of *Inner* class in above example :

```
public Inner()
{
        sum = 50;
        divisor = 4;
        remainder = sum%divisor;
}
```

Because of this assignment statement, the variable *sum* is not *effectively final* anymore. As a result, the Java compiler generates an error message similar to "local variables referenced from an inner class must be final or effectively final".

Declaration inside an if statement

```
// Java program to illustrate Declaration of
// local inner classes inside an if statement
public class Outer
{
    public int data = 10;
    public int getData()
    {
        return data;
    }
    public static void main(String[] args)
    {
        Outer outer = new Outer();

        if(outer.getData() < 20)
        {
            // Local inner class inside if clause
            class Inner
            {
                public int getValue()
                {
                    System.out.println("Inside Inner class");
                    return outer.data;
                }
            }

            Inner inner = new Inner();
            System.out.println(inner.getValue());
        }
        else
        {
            System.out.println("Inside Outer class");
        }
    }
}
```

Run on IDE

Output:

```
Inside Inner class
10
```

Demonstrating Erroneous codes for Inner class

```
// Java code to demonstrate that inner
// classes cannot be declared as static
public class Outer
{
    private int getValue(int data)
    {
        static class Inner
        {
            private int getData()
            {
                System.out.println("Inside inner class");
                if(data < 10)
                {
                    return 5;
                }
                else
                {
                    return 15;
                }
            }
        }
        Inner inner = new Inner();
        return inner.getData();
    }

    public static void main(String[] args)
    {
        Outer outer = new Outer();
        System.out.println(outer.getValue(10));
    }
}
```

Run on IDE

Output:

```
Compilation error
```

Explanation: The above program causes compilation error because the inner class cannot be declared
as static. Inner classes are associated with the block they are defined within and not with the external class(Outer in this case).

```
// Java code to demonstrate
// the scope of inner class
public class Outer
{
    private void myMethod()
    {
        class Inner
        {
            private void innerMethod()
            {
                System.out.println("Inside inner class");
            }
        }
    }

    public static void main(String[] args)
    {
        Outer outer = new Outer();
        Inner inner = new Inner();
        System.out.println(inner.innerMethod());
    }
}
```

Run on IDE

Output:

```
Compilation error
```

Explanation: The above program causes compilation error because the scope of inner classes are restricted to the block they are defined in.