NESTED CLASSES

In Java, **a class can have another class as its member**. The class written within another class is called the nested class, and the class that holds the inner class is called the outer class.

Java inner class is defined inside the body of another class. Java inner class can be declared private, public, protected, or with default access whereas an outer class can have only public or default access.

The syntax of nested class is shown below:

```
class Outer_Demo

{

class Nested_Demo

{

    }

  }
```

Types of Nested classes

There are two types of nested classes in java. They are non-static and static nested classes. The non-static nested classes are also known as inner classes.

- Non-static nested class (inner class)
  - Member inner class
  - Method Local inner class
  - Anonymous inner class
- Static nested class

Type	Description
Member Inner Class	A class created within class and outside method.
Anonymous Inner Class	A class created for implementing interface or extending class. Its name is decided by the java compiler.
Method Local Inner Class	A class created within method.
Static Nested Class	A static class created within class.
Nested Interface	An interface created within class or interface.

INNER CLASSES (NON-STATIC NESTED CLASSES)

Inner classes can be used as the security mechanism in Java. Normally, a class cannot be related with the access specifier **private.** However if a class is defined as a member of other class, then the inner class can be made private. This class can have access to the private members of a class.

The three types of inner classes are

- Member Inner Class
- Method-local Inner Class
- Anonymous Inner Class

Member Inner Class

The **Member inner class is a class written within another class**. Unlike a class, an inner class can be private and once you declare an inner class private, it cannot be accessed from an object outside the class.

The following program is an example for member inner class.

```
class Outer_class {
  int n=20;
  private class Inner_class {
  public void display() {
  System.out.println("This is an inner class");
  System.out.println("n:"+n);
    }
  }
   void print_inner() {
    Inner_class inn = new Inner_class();
    inn.display();
  }
}
  public class Myclass {
  public static void main(String args[]) {
   Outer_class out= new Outer_class();
   out.print_inner();
  }
}
```

Output:

This is an inner class

Method-local Inner Class

In Java, a class can be written within a method. Like local variables of the method, the scope of the inner class is restricted within the method. A method-local inner class can be instantiated only within the method where the inner class is defined. The following program shows how to use a method-local inner class. The following program is an example for Method-local Inner Class

```
public class Outer_class {
  void Method1() {
     int n = 100;
    class MethodInner_class {
    public void display() {
```

```
            System.out.println("This is method inner class ");
            System.out.println("n:"+n);
          }
        }
        MethodInner_class inn= new MethodInner_class();
        inn.display();
      }
      public static void main(String args[]) {
        Outer_class out = new Outer_class();
        out.Method1();
      }
    }
```

Output:

This is method inner class

n: 100

Anonymous Inner Class

An inner class declared without a class name is known as an anonymous inner class. The anonymous inner classes can be created and instantiated at the same time. Generally, they are used whenever you need to override the method of a class or an interface. The syntax of an anonymous inner class is as follows –

```
abstract class Anonymous_Inner {
  public abstract void Method1();
}
```

The following program is an example for anonymous inner class.

```
public class Outer_class {
  public static void main(String args[]) {
    Anonymous_Inner inn = new Anonymous_Inner() {
      public void Method1() {
        System.out.println("This is the anonymous inner class");
      }
    };
    inn.Method1();
  }
}
```

Output:

This is the anonymous inner class

Static Nested Class

A static inner class is a nested class which is a static member of the outer class. It canbe accessed without instantiating the outer class, using other static members. Just like static members, a static nested class does not have access to the instance variables and methods of the outer class. Instantiating a static nested class is different from instantiating an inner class. The following program shows how to use a static nested class.

```
public class Outer_class {
  static class inner_class{
    public void Method1() {
    System.out.println("This is the nested class");
    }
  }
    public static void main(String args[]) {
    Outer_class.inner_class obj = new Outer_class.inner_class();
    obj.Method1();
  }
}
```

Output:

This is the nested class

Advantage of java inner classes:

There are basically three advantages of inner classes in java. They are as follows:

- Nested classes represent a special type of relationship that is it can access all the members of outer class including private.
- Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.
- It provides code optimization. That is it requires less code to write.