Nested classes OR Inner classes :
----
* In Java, It is possible to assign a class (inner class) inside another class (outer class) is called Neasted class OR Inner class.

Example :
Outer.java

```
public class Outer   //Outer.class
{
     private class Inner //Outer$Inner.class
     {
     }
}
```

* An outer class (top level class) we can declare with public, abstract, final, sealed and non-sealed modifier.

* An inner class, We can declare with private, default (package-private), protected, public, **static,** abstract and final modifier.

* An inner class, .class file will be represented by $ symbol.

* Just like inner class, We can also write class/record/interface/enum and annotation inside another class/record/interface/enum and annotation.

* An inner class represents **HAS-A relation** and **strong encapsulation** with Outer class.

a) HAS-A Realtion :

```
public class Outer
{
   private class Inner  //Outer class HAS-A Inner class
   {
   }
}
```

b) Strong Encapsulation :

```
public class Trainer
{

}

public class Parent
{

}
```

```
public class Student
{
   private int totalMarks = 490;

   public int getTotalMarks()
   {
      return this.totalMarks;
   }
}
```

Trainer/Parents can access the Student marks with the help of getter (Encapsulation)

* An inner class can directly **access the private data of outer class**, Here no need to define any setter and getter because by default inner class provides Strong Encapsulation.

```
class Outer
{
    private int data = 100;

    public class Inner //non static inner class
    {
        public void access()
        {
            System.out.println("private variable data value is :"+data);
        }
    }
}
```

Direct access setter and getter is not reqd.

* As we know inner class provides HAS-A relation with outer class so It is similar to HAS-A relation concept, how we will decide when we should use HAS-A relation and when we should use inner class

a) HAS-A relation :
----
* If we are able to use **a class** as a property to multiple classes then We should use HAS-A relation, In this concept, to achieve encasulation setter and getter both are required.
Example :

```
public class Trainer
{
   private Address address;

}

public class Employee
{
   private Address address;

}

public class Student
{
   private Address address;

}
```

```
public class Address
{

}
```

b) If we have strong relation between two classes that means inner class cannot use by any other class It is only meant for Outer class (Tight coupling) then we should use inner class.

Example 1 :
```
public class Laptop
{
    private class Motherboard
    {

    }
}
```

Example 2 :
```
public class Person
{
  private class Heart
  {
  }

}
```

Example 3 :
```
public class University
{
  private class Department
  {
  }

}
```

Advantages of inner class :
----
1) Grouping the tightly coupled classes together:
Classes which are not reusable in multiple places can be defined as inner class.

For example : LinkedList class internally uses Node class so Node class must be inner class.

2) To achieve encapsulation (No setter and Getter):
Inner class can directly access the private data of Outer class, In case of HAS-A relation multile classes can access the private data with the help of getter but in case on inner class only inner class can access the private data outer class.

3) More Readable and Maintainable Code :
Rather than creating a new class we can create inner class so that it is easy to maintain.

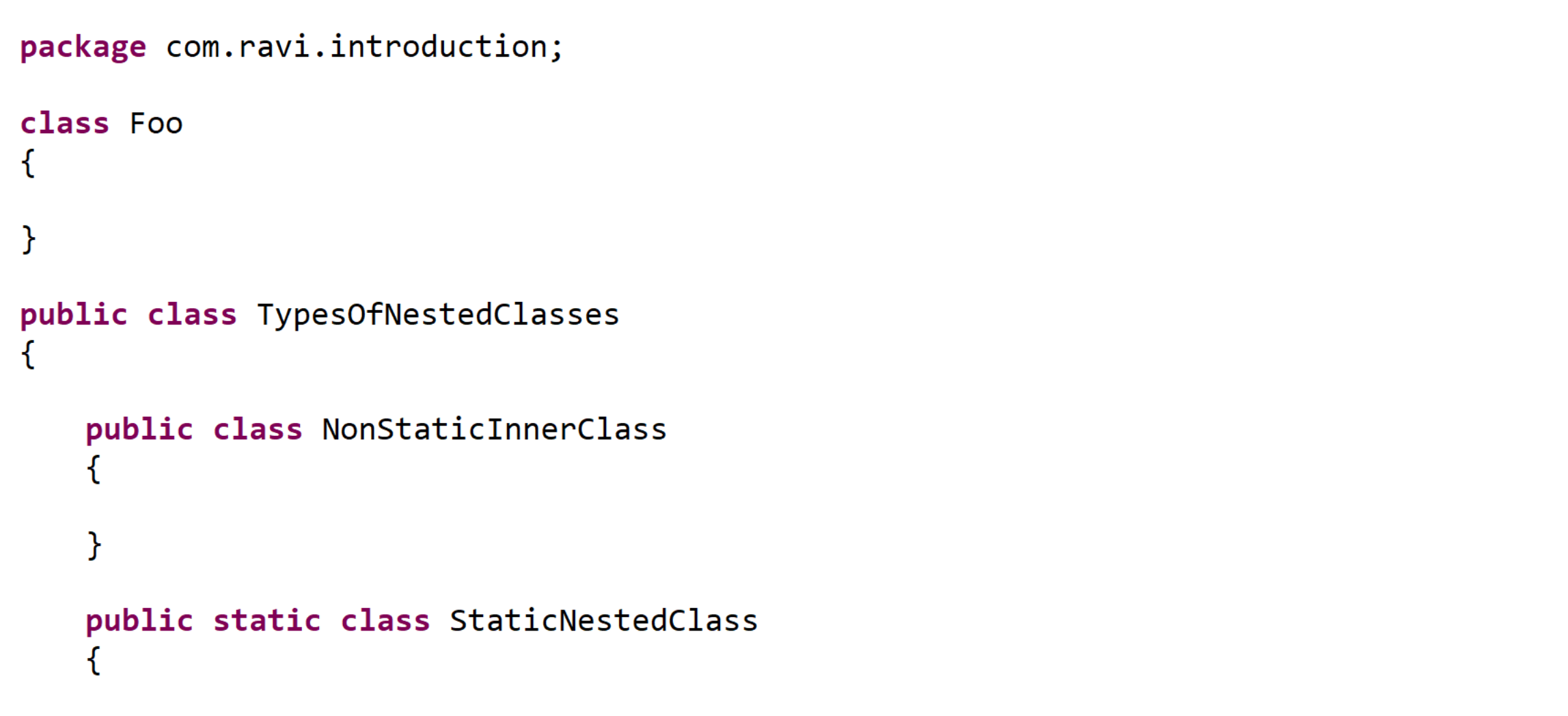
4) Hiding implementation :
Inner class helps us to hide implementation of class so only Outer class can use the inner class features.

Types of Inner classes in java :

There are 4 types of inner classes in java, 2 inner classes we can write at class level (static + non static) and two inner classes we can write a method level (local + anonymous)

1) At Class Level
----
a) Non Static Nested class OR Inner class
b) Static Nested class.

2) At Method Level
----
c) Method / Method local class (With class Name)
d) Anonymous inner class (Without class Name)



WAP to show all types of Nested classes in java ?

```
package com.ravi.introduction;

class Foo
{

}

public class TypesOfNestedClasses
{

    public class NonStaticInnerClass
    {

    }

    public static class StaticNestedClass
    {

    }

    public void localInnerClass()
    {
        final class MethodLocal
        {

        }
    }

    public static void main(String[] args)
    {
        //Anonymous Inner class
       Foo f1 = new Foo()
       {

       };
    }
}
```

Programs that describes all different possible combinations :

```
package com.ravi.introduction;

public class Demo1
{
   class A{}

   record B() {}

   interface C {}

   enum D{}

   @interface E{} //Annotation

}


package com.ravi.introduction;

public record Demo2()
{
       class A{}

       record B() {}

       interface C {}

       enum D{}

       @interface E{}  //Annotation

}


package com.ravi.introduction;

public interface Demo3
{
       class A{}

       record B() {}

       interface C {}

       enum D{}

       @interface E{}  //Annotation
}


package com.ravi.introduction;

public enum Demo4
{
       HR, SALES; //First line is reserved for enum constant

       class A{}

       record B() {}

       interface C {}

       enum D{}

       @interface E{}  //Annotation
}

package com.ravi.introduction;

public @interface Demo5
{
       class A{}

       record B() {}

       interface C {}

       enum D{}

       @interface E{}  //Annotation
}
```
