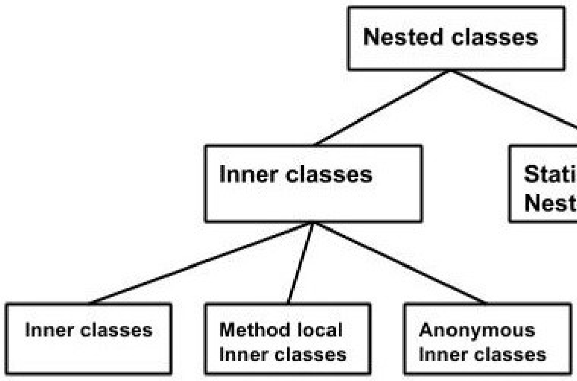
Inner / Nested Class - Inner class means one class which is a member of another class. There are basically four types of inner classes in java.



Inner Classes (Non-static Nested Classes) - Inner classes are a **security mechanism** in Java. Since, 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.

There are 3 types of Non-Static Nested Classes based on where we declare them -

1. Inner Class -

- ➢ We just need to write a class within a 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.

```
class Outer_Demo { // Private Inner Class (Example 1)
	int num;
	private class Inner_Demo {  // inner class
		public void print() {
			System.out.println("This is an inner class");
		}
	}
	void display_Inner() {   // Accessing he inner class from the method within
		Inner_Demo inner = new Inner_Demo();
		inner.print();
	}
}
public class My_class {
	public static void main(String args[]) {
		Outer_Demo outer = new Outer_Demo(); // Instantiating the outer class
		outer.display_Inner();  // Accessing the display_Inner() method.
  }
}
```

Output:

```
This is an inner class.
```

Example 2:

```
class Outer { // Public / Default Inner Class Example
  class Inner { // Simple nested inner class
    public void show() {
        System.out.println("In a nested class method");
    }
  }
}
class Main {
  public static void main(String[] args) {
      Outer.Inner in = new Outer().new Inner();
      in.show();
  }
}
```

Output:

```
In a nested class method
```

- **Inner** Class **can access private** members of it's **Outer** Class.
- We **can't** have **static method** in a **nested inner class** because an inner class is **implicitly associated** with an **object** of its **outer class** so it cannot define any static method for itself.

```
class Outer {
  void outerMethod() {
    System.out.println("inside outerMethod");
  }
  class Inner {
    public static void main(String[] args){
      System.out.println("inside inner class Method");
    }
  }
}
```

Output:

```
Error illegal static declaration in inner class Outer.Inner public static void main(String[] args) modifier
'static' is only allowed in constant variable declaration.
```

2. Method Local Inner Class -

- Inner class can be declared within a method of an outer class.
- Like local variables, 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.

```
public class Outerclass {
        void my_Method() {  // instance method of the outer class
                int num = 23;
                class MethodInner_Demo {   // method-local inner class
```

```
                        public void print() {
                                System.out.println("This is method inner class "+num);
                        }
                } // end of inner class
                MethodInner_Demo inner = new MethodInner_Demo();  // Accessing the inner class
                inner.print();
        }
        public static void main(String args[]) {
                Outerclass outer = new Outerclass();
                outer.my_Method();
        }
}
```
Output

```
This is method inner class 23
```

- Method local inner class **can't** be marked as **private**, **protected**, **static** and **transient** but **can** be marked as **abstract** and **final**, but not both at the same time.

3. Anonymous Inner Class -

- An inner class declared **without a class name** is known as an anonymous inner class.
- In case of anonymous inner classes, we **declare** and **instantiate** them at the **same** time.
- Generally, **used** whenever we need to **override** the **method** of a class or an interface.

```
abstract class AnonymousInner {
        public abstract void mymethod();
}
public class Outer_class {
   public static void main(String args[]) {
      AnonymousInner inner = new AnonymousInner() {
         public void mymethod() { // Overriding mymethod() of AnonymousInner Class
            System.out.println("This is an example of anonymous inner class");
         }
      };
      inner.mymethod();
   }
}
```
Output

```
This is an example of anonymous inner class
```

➢ Anonymous Inner Class as Argument -

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

Output

```
Hello, This is an example of anonymous inner class as an argument
```

Static Nested Class -

- ➢ A static inner class is a nested class which is a static member of the outer class.
- ➢ It can be 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.

```
public class Outer {
  static class Nested_Demo {
    public void my_method() {
      System.out.println("This is my nested class");
    }
  }
  public static void main(String args[]) {
    Outer.Nested_Demo nested = new Outer.Nested_Demo();
    nested.my_method();
  }
}
```

Output:

```
This is my nested class
```