10. Java Classes in Depth - Part 1

(Abstract Class)

(*) Class which can provide [0-100]% abstraction.

(*) It can have method implementations as well, along with abstract methods.

```
abstract class{

    abstract void method();
    public void method1(){ //do something }
}
```

Nested Class → A Class within another class is called Nested Class.

When will Nested Class be used?

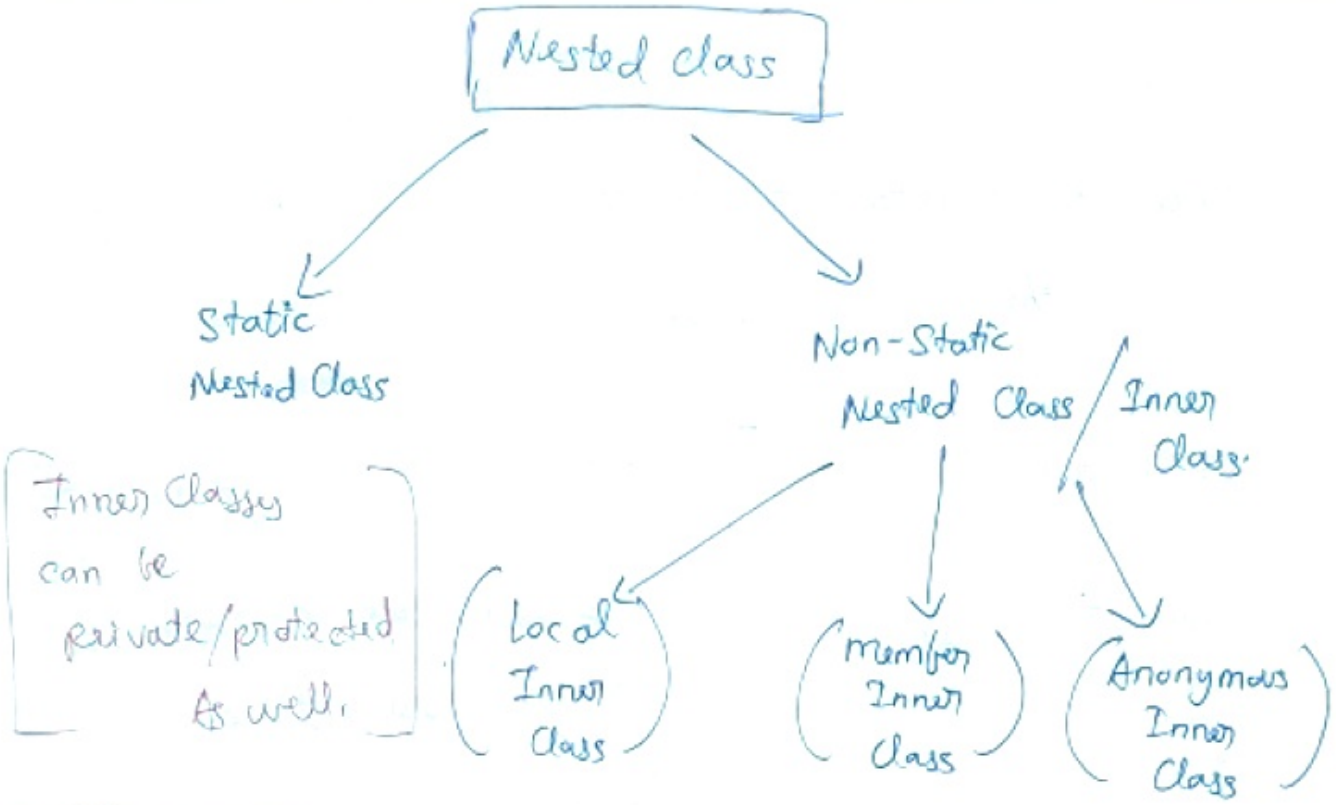
If you know that a Class(A) will be used only and only by one another class(B), then instead of creating a new file, why not create a class inside class(B) itself

Within one class you can group logically related files

```
class B{

    Class A{
    }
}
```

Nested class



static Nested Class

Example

```
class OuterClass{
  int a=10;
  static int b=20;

  static class NestedClass{

    public void print(){
      sout(a+b);
    }
}.
```

① Compile Error
Non static member variables cannot be accessed by static Inner class

(Imp)

- Static Nested Class can have private/protected as Access Specifier.
- Non-static instance variable cannot be accessed
- No need to create instance of outerclass

```
main(){

  OuterClass.NestedClass obj =
    new OuterClass.NestedClass();
```

② Since its static so we need not create an instance and directly access from class name.

Non-static Nested Class or Inner Class.

↳ For this. Since the class is not a static block.

Hence we have to initialize the Outer Class first and then initialize the Nested Inner Class.

```
class OuterClass{

    class InnerClass{

    }
}.
```

If you try

```
OuterClass.InnerClass obj
    = new OuterClass.InnerClass();
```

Will throw Compile ERROR.

because using ClassName only static methods or variables can be accessed.

Since it a non-static inner class.

```
OuterClass out = new OuterClass();
```

Syntax

```
OuterClass.InnerClass in = new out.new InnerClass();
```

(V.Imp)

Local Inner Class

↳ Inner Class is defined inside a

① method block ② Loop ③ IF Condition

Example

```
class OuterClass{
    public void display(){
        class LocalInnerClass{

        }
    }.
}
```

(Only default access specifier is allowed)

Anonymous Inner Class

An Inner class without any Name is called Anonymous Class.

Why is it used?

When we want to override behaviour of an method with creating any class.

Example

```
public abstract class Car{
    public abstract void pressbreak();
}.

main(){
    Car electric = new Car() {
        @Override
        public void pressbreak(){
            //-----
        }
    }
    electric.pressbreak();
}.
```