Inner Class

~~inner~~ class inside class ; Weird huh?

If a class is completely dependent on other class,

then we can create that class inside

Eg :-

```
class A {
    int age;

    public void show() {
            sout ("show");
    }

    class B {
        public void config() {
                sout ("in config");
        }
    }
}
public class Demo {

        main() {
```

→ If we want to print show :

```
A obj = new A();
A.show.
```

→ If we want to print config :

```
B obj = new B()
B.config()
```

Note :- When we compile this code, we get A.class, Demo.class & A$B.class

⚠ B cannot be resolved to a type

so,

```
A.B obj1 = A.new B(); ✗
         = new A.B() ✗
```

as, class B is non-static, we need a obj of A to access it so,

```
A.B obj1 = obj.new B();
obj1.config();
```

→ If inner class is static,

```
A.B obj1 = new A.B() ✓
```

> static is not allowed for outer class.

Anonymous class

→ Let we have a class A having show method

→ We want to have different implementation of show(), what we can do?

we can extends it & do method overriding.

→ But this will unnecessary create a extra class of that extended class.

→ So what we can do?

Usually in main method, we create an object of class A that uses the definition of class A defined already.

But if we define it during creation of object.
i.e. we will provide different implementation of show.

Let us :-

```
class A {

    public void show() {    }
}
class Demo {
    main() {

→       A obj = new A() {
            public void show() {
                sout("in new show");
            }
        };
    }
}
```

Printed output → In new show

This is the inner class that had created inside demo class having no name. That is why it is anonymous.

Evidence:- Look in explorer, a Demo $ 1.class had been created.

Lets combine abstract class, inner class, anonymous class

```
abstract class A
{   public abstract void show();  }

class B extends A {
    public void show() {
        System.out.println("in B show");
    }
}

public class Demo
{
    public static void main() {
        A obj = new B();
        obj.show();
    }
}
```

→ in B show

So, the entire purpose of B class is to implement the abs. class A, only.

So, why to create that? Remove it...
Use anonymous class implementation.

i.e

```
A obj = new A()
{
    public void show()
    {
        System.out.println("in new show");
    }
};
obj.show();
```

(8) Are we here creating object of A but A is abstract class.

we want create object of abstract class.

No!

The object is of the anonymous inner class.

Note :-
① If we want to ~~use object~~ implement the interface of abstract class only once, then we use anonymous inner class

② We can even implement multiple method (abstract