Inner Classes:

Inner class cant exist without outer class.

Like department class, if there, cant exist without university class. So its better to create department class inside university class which will infer that only when university class is created department class will be crested. That is why it is better to create department inside university.

e.g. When we have Map like:

|  |  |
| --- | --- |
| **Key** | **Value** |
| Name | Adarsh |
| Age | 24 |

So here, each row is an ***ENTRY*** into the map. For this purpose there is an ENTRY class created inside Map Class. So that only when tere is a map there will be an entry.

Similarly there is inner interface, which is actually defined for Maps like:

Interface Map{

Interface Entry{

}

}

Map is outer interface and Entry is inner interface.

Outer class and inner class has a HAS-A relationship i.e. conposotion and aggregation

## Types of Inner Class based on location an behavior:

* Normal/Regular Inner class
* Anonymous inner class
* Static nested class
* Method local inner class

### Normal/Regular Inner class

Class Outer

{

Class Inner{

}

}

This is called a regular inner class.

Here upon compilation, 2 class fileas will be generated for both class.

* Outer.class
* Outer$Inner.class

‘$’ is used for inner class representation

Now if we run either of class files , we will get NoSuchMethodExcetion:main , because no main method is there.

Class Outer

{

P s v m(String… a)

{

S.O.P(“outer”);

}

Class Inner{

}

}

If we run Outer.class file we will get Output:outer.

If we run Outer$Inner.class file we will get same NoSuchMethodException:main.

Class Outer

{

Class Inner{

P s v m(String… a) //Error

{

S.O.P(“outer”);

}

}

}

Now if we run Outer.class same exception.

For Inner class, though this timwe the code wont even compile.

Coz as we have read ***without outer class inner class doesn’t exist***, which means we, in this case JVM, cant directly call main method, it must be called from an object of Outer class. That means ***inner class cant have any static members.*** *Coz if* they do they will b eligible for a direct call.

## Calling Inner class methods from outer class:

#### CASE 1: Accessing Inner class member from Static Methods of Outer class

Steps:

* Create outer class object, coz ***without outer class inner class doesn’t exist***, so to access inner class object of outer class is must.
* Now create Inner class object, refereeing from outer class object. Coz we need the inner class object, obviously, to access any member of (inner) class.
* Now access the member from the inner class method.

Class Outer

{

Public static void main(String… a)  
 {  
 Outer o=new Outer();

Outer.Inner i=o.new Inner(); // This how inner class object is created

i.X();

}

Class Inner{

Public void X()

{

S.O.P(“outer”);

}

}

}

### Case 2: Accessing inner class code from non-static method/instance method of outer class:

Steps:

* Call non-static method of Outer class from main by creating the object of Outer and calling the method from this object.
* Now in an instace method or non-static method we will always have ***this*** reference, which is the object of current Outer class. So we don’t need to call Inner class from explicit Outer class object.
* Call the inner class method from inner class object.

Class Outer

{

public static void main(String… a)  
 {  
 Outer o=new Outer();

o.A();

}

public void A()

{

Inner i=new Inner();

i.X();

}

Class Inner{

Public void X()

{

S.O.P(“outer”);

}

}

}

### Case 3: Calling Inner class methods from class different from static method of Outer class i.e. from outside classes

Inner class method will get called the same way as we called them from static method of Outer class

Class Outer

{

Class Inner{

Public void X()

{

S.O.P(“outer”);

}

}

}

Class Main{

Public static void main(String… a)  
 {  
 Outer o=new Outer();

Outer.Inner i=o.new Inner(); // This how inner class object is created

i.X();

}

}

# Accessing of members of Outer class from Inner class:

We can access all the members of outer class be it public private static non static in Inner class directly.

**package** dom;

**import** dom.Outer.Inner;

**public** **class** Outer {

**int** a=1;

**static** **int** *b*=2;

**private** **int** c=3;

**public** **static** **void** main(String[] args) {

((**new** Outer()).**new** Inner()).execute();

}

**class** Inner {

**public** **void** execute()

{

System.***out***.println(a);

System.***out***.println(*b*);

System.***out***.println(c);

}

}

}

Output:

1

2

3

Accessing different level members from inner class:

**package** dom;

**import** dom.Outer.Inner;

**public** **class** Outer {

**static** **int** *a* = 1;

**public** **static** **void** main(String[] args) {

((**new** Outer()).**new** Inner()).execute();

}

**class** Inner {

**int** a = 2;

**public** **void** execute() {

**int** a = 3;

System.***out***.println("value from Inner class method variable:" + a);

System.***out***.println("value from Inner class instance variable:" + **this**.a);

System.***out***.println("value from Outer class Instance var:" + Outer.**this**.*a*);

}

}

}

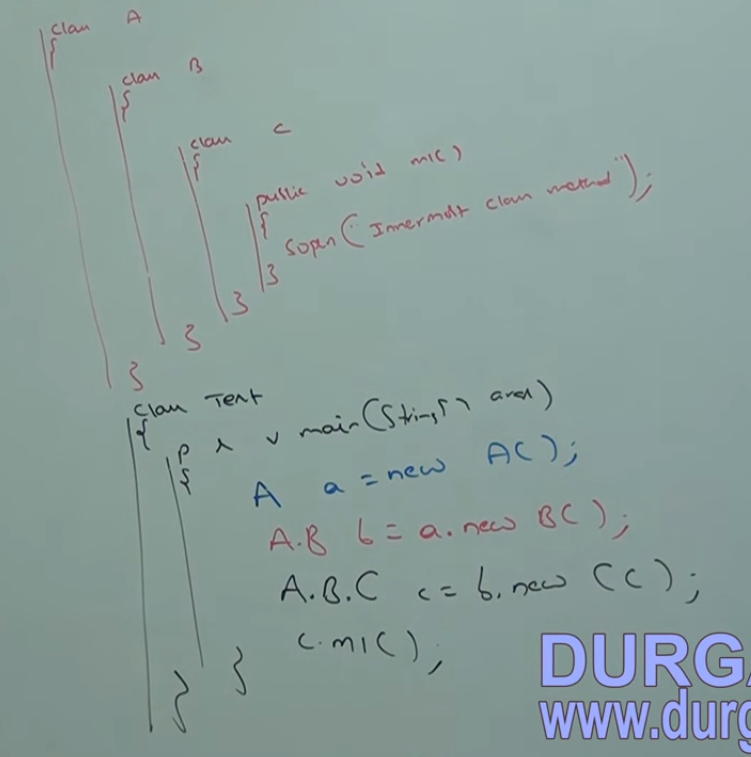
O/P:

value from Inner class method variable:3

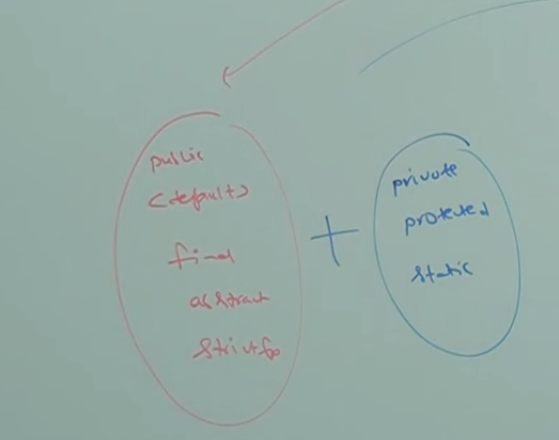
value from Inner class instance variable:2

value from Outer class Instance var:1

### Nested Inner Classes



### Modifiers Allowed in an inner class:



# Method Local Inner Classes

We can declare inner class inside a method. This inner class is called MLIC.

Main purpose is to define method specific repeated functions inside inner class which will be repeatedly accessed by method only and not ouside of that methpd

# Anonymous Inner class:

Inner Class without Name.

You want to go to building A from B, which is 10min walking distance. Now, while going from B to A, you thought u passed A . Now to confirm you ask someone on the road where is A. He told its 3min walk from here. Now you walkd for 3min and u got to A.

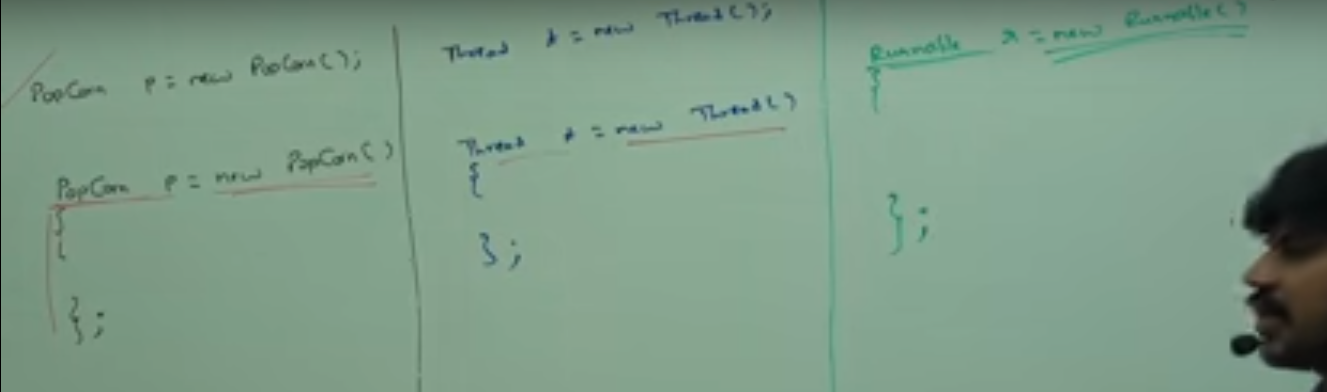
Now in this eg, u didn’t asked the person whi helped for his name or address or anything, He was anonymous and helped u. U needed help for that instant and u took help from that anonymous person.

Same is the case with anonymous class. Anonymous classes are create just for instance use or one time use.

Tere are 3 types of Anonymous inner class:

1. AIC that extends a class
2. AIC that implements an interface
3. AIC that is defined inside arguments

### AIC that extends a class



Thread and Runnable is an interface so cant make object like new Runnable(), therefore have to write a anonymous inner class under {}.

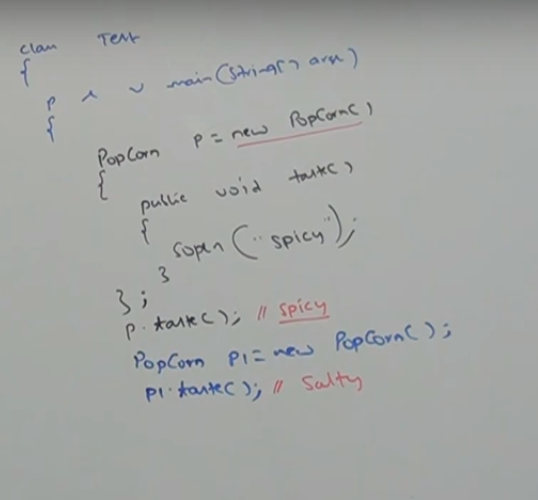
Popcorn P=new Popcorn()

{

};

Above syntax means we are creating a class (AIC) that extends Popcorn class. And the ***new Popcorn()*** is the child class object not Popcorn class object.

For one time requirement don’t go for a separate class extending the Parent class, instead use AIC.

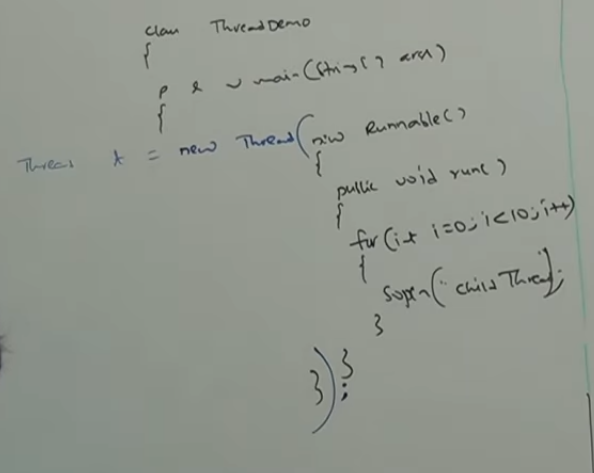


For above code 3class files will be generated.Test.class ,Test$1.class. First one is the main Test class’s class file and second one is for anonymous class. Now coz anonymous class don’t have name their class file name follows following syntax:

***MainOutercClassName$SequenceNumberOfAnonymousInnerClassInMainClass.class***

Which mean if there were more AIC here then their class files would be named Test$2.class, Test$3.class……..and so on…..

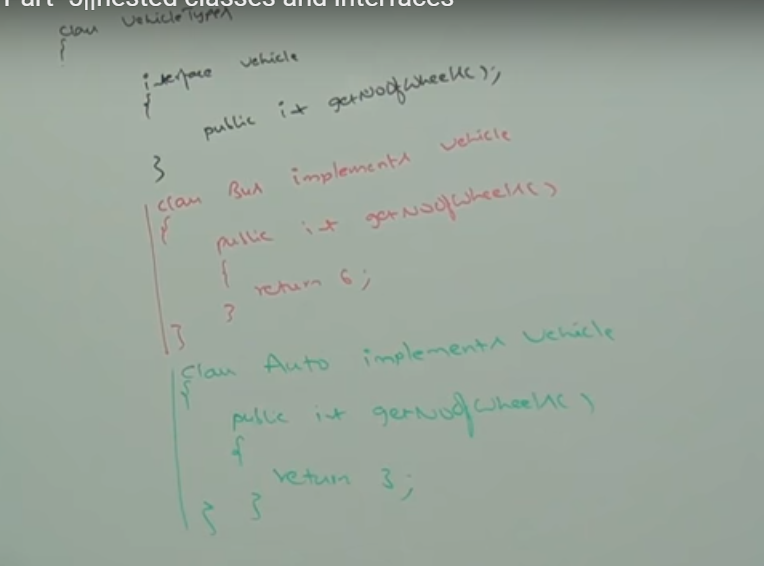
### AIC that is defined inside arguments

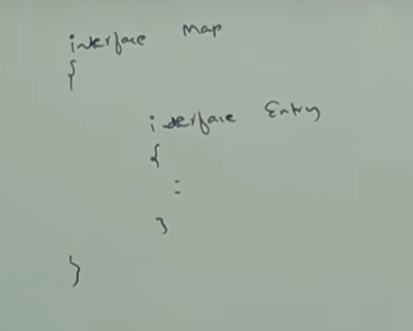


## Nested Classes and Interface:

W.R.T Classes and Interface, anything is possible inside anything

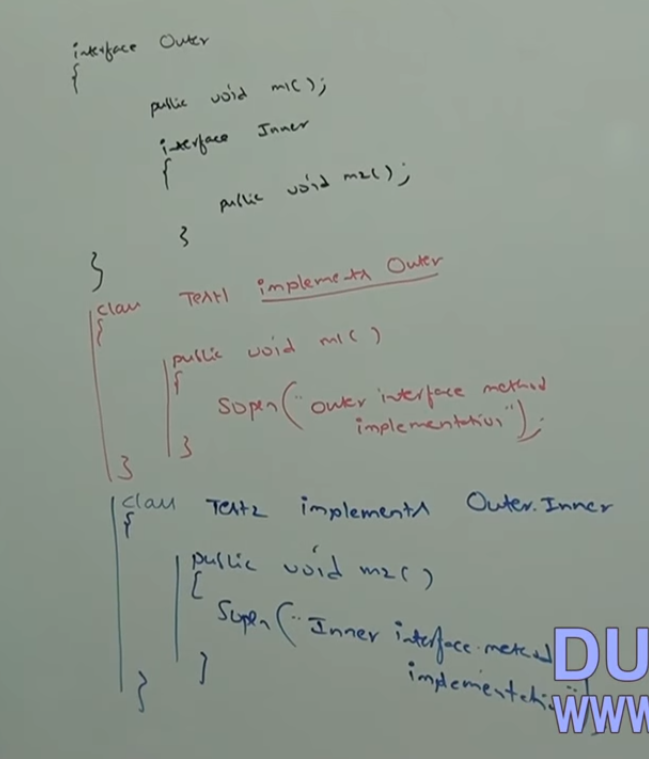
* Classes inside Interface
* Classes inside Classes
* Interface inside Interface
* Interface inside Classes





Inner Interface is always ***public static****.*

Hence We can touch inner interface without existence of Outer Interface, coz its static. So, whenever we implement outer interface, we dnt need to implement inner interface and whenever we implement iner interface we don’t need to implement outer interface either.

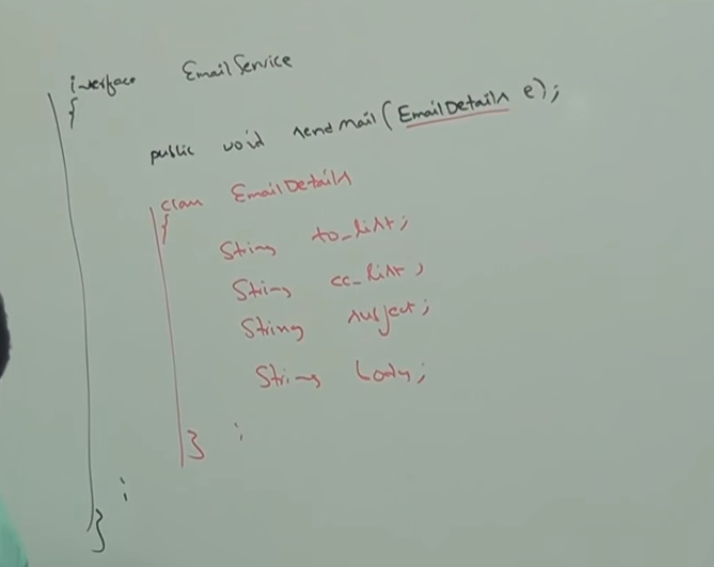


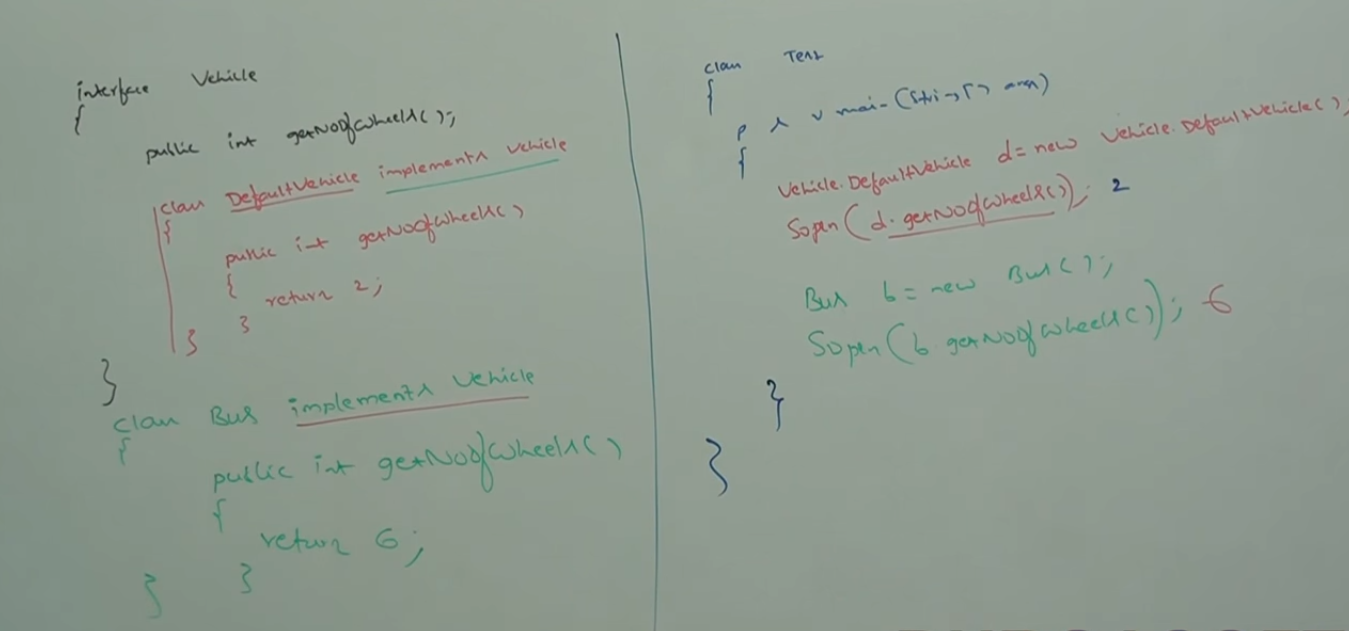
If a class is closely associated with an interface we can implement that class inside an interface.

We provide class inside interface to provide default implementation of the interface.

Hwre EmailDetails is only being used with senMail() so we define it here.

Class inside interface is by default ***public static.***





# Static Nested Class