**Inner Classes – Part-01**

* **Inner Classes:**

Sometimes we can declare a class inside another class, such type of classes is called inner classes.

Inner-classes concept introduced in Java1.1 version to fix GUI (AWT) bugs as part of event handling. But, because of powerful features and benefits of inner-classes slowly programmers started using in regular coding also.

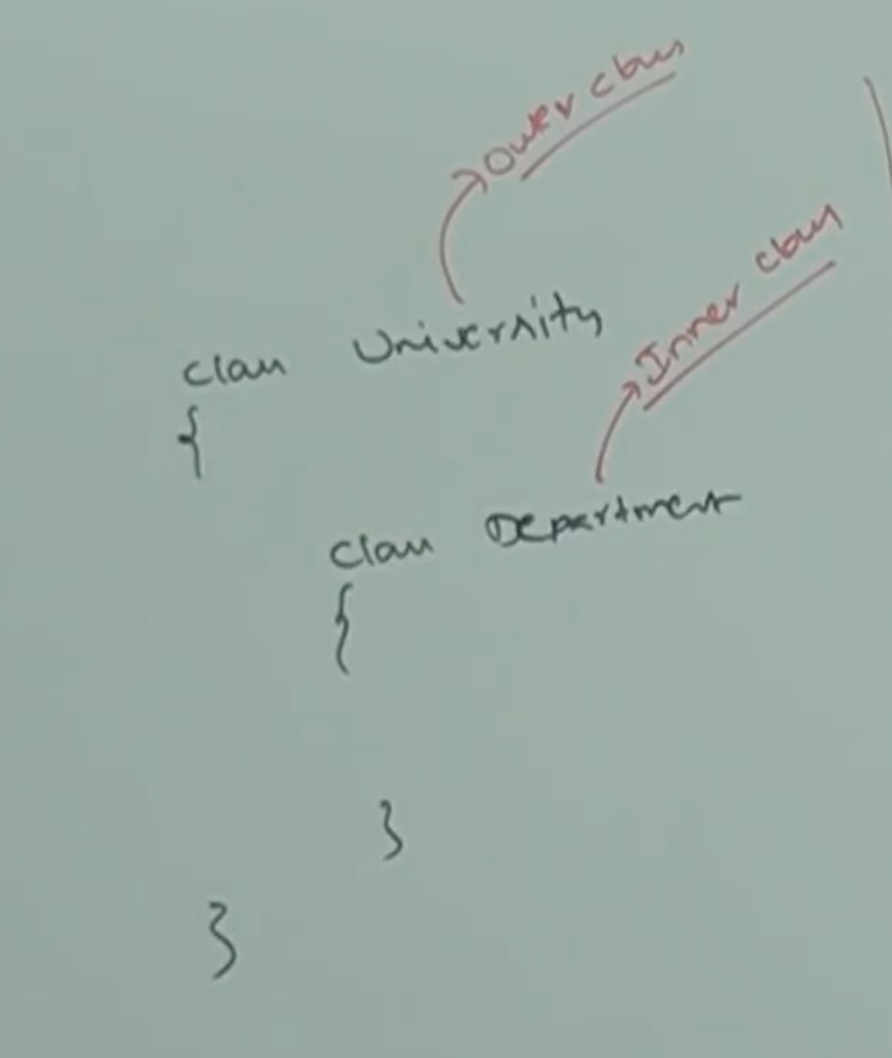
Note: Java1.0 came in 1995, with in 3 months span they released the next version to improve the performance and GUI issues.

**When to go for Inner-Classes?**

Without existing one type of object, if there is no chance of existing another type of object, then we should go for inner-classes.

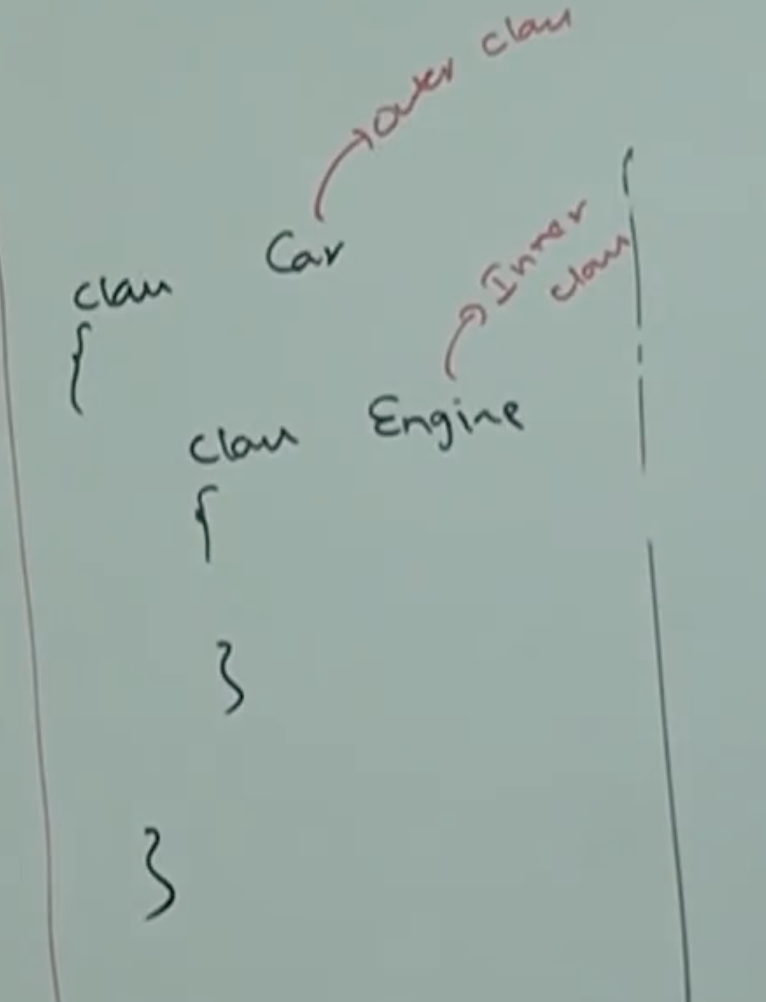
* **Example\_1:**

University consists of several departments, without existing university there is no chance of existing department. Hence we have to declare department class inside university class.



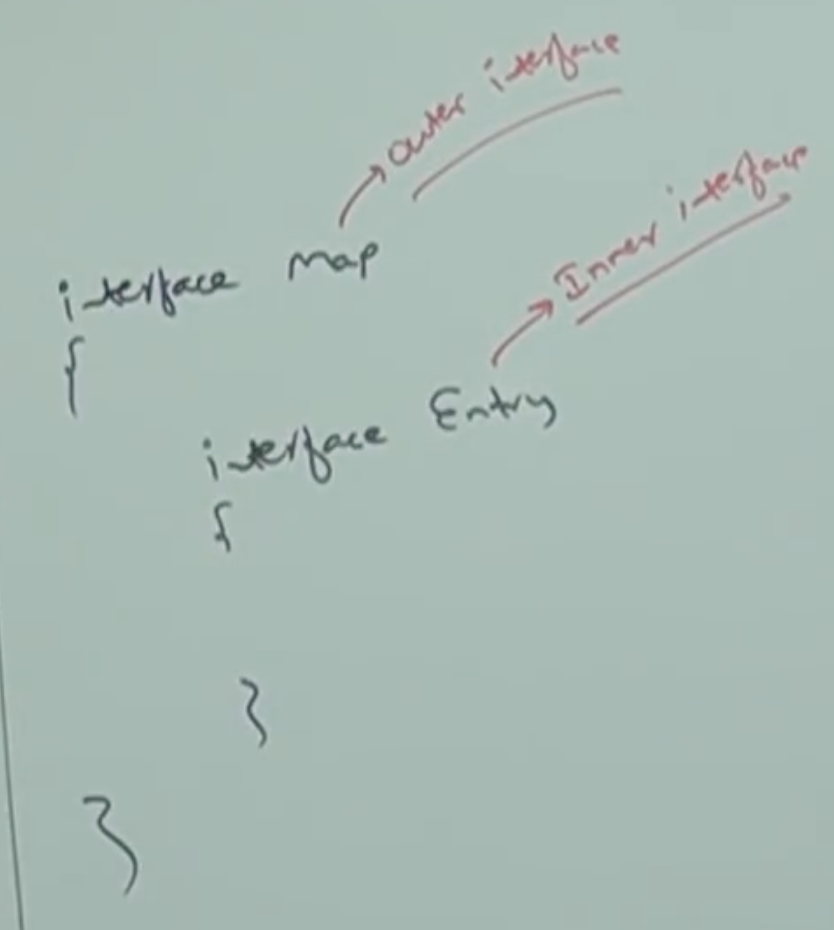
* **Example\_2:**

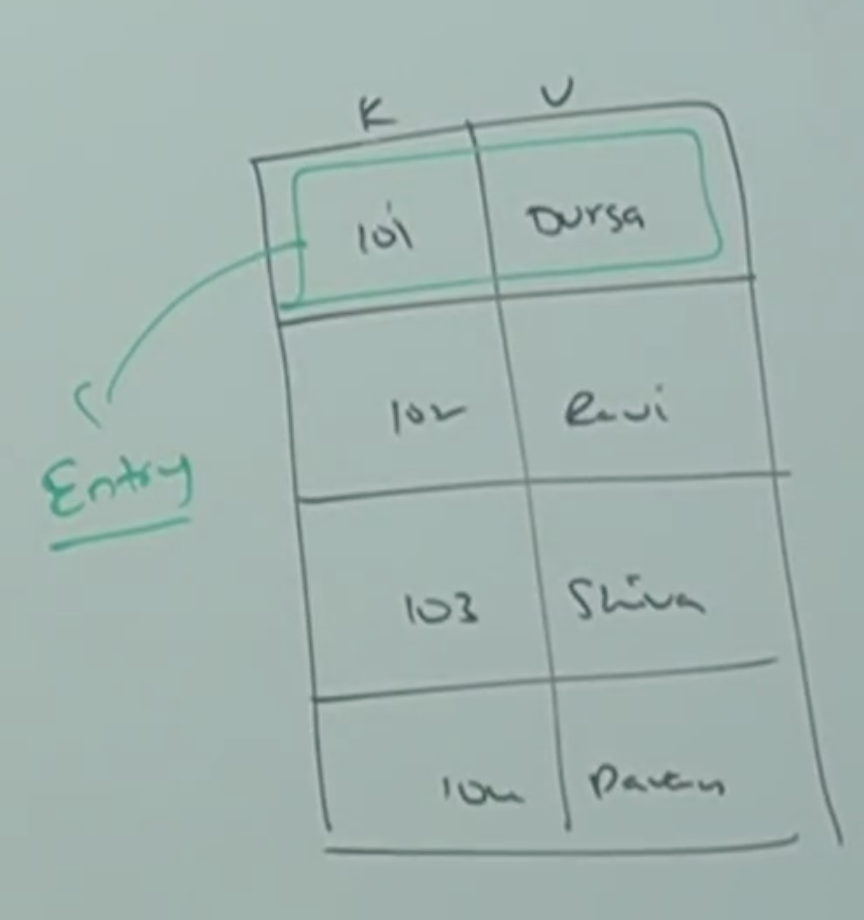
Without existing car object there is no chance of existing Engine object. Hence, we have to declare Engine class inside Car class.



* **Example\_3:**

Map is a group of Key Value pairs. And each key value pair is called an Entry. Without existing Map object, there is no chance of existing Entry object. Hence, interface Entry is defined inside map interface.





* **Note:**

1. Without existing outer class object there is no chance of existing inner class object.
2. The relation between outer class and inner class is not IS-A relation and it is HAS-A (Composition or Aggregation) relationship.

* **Types:**

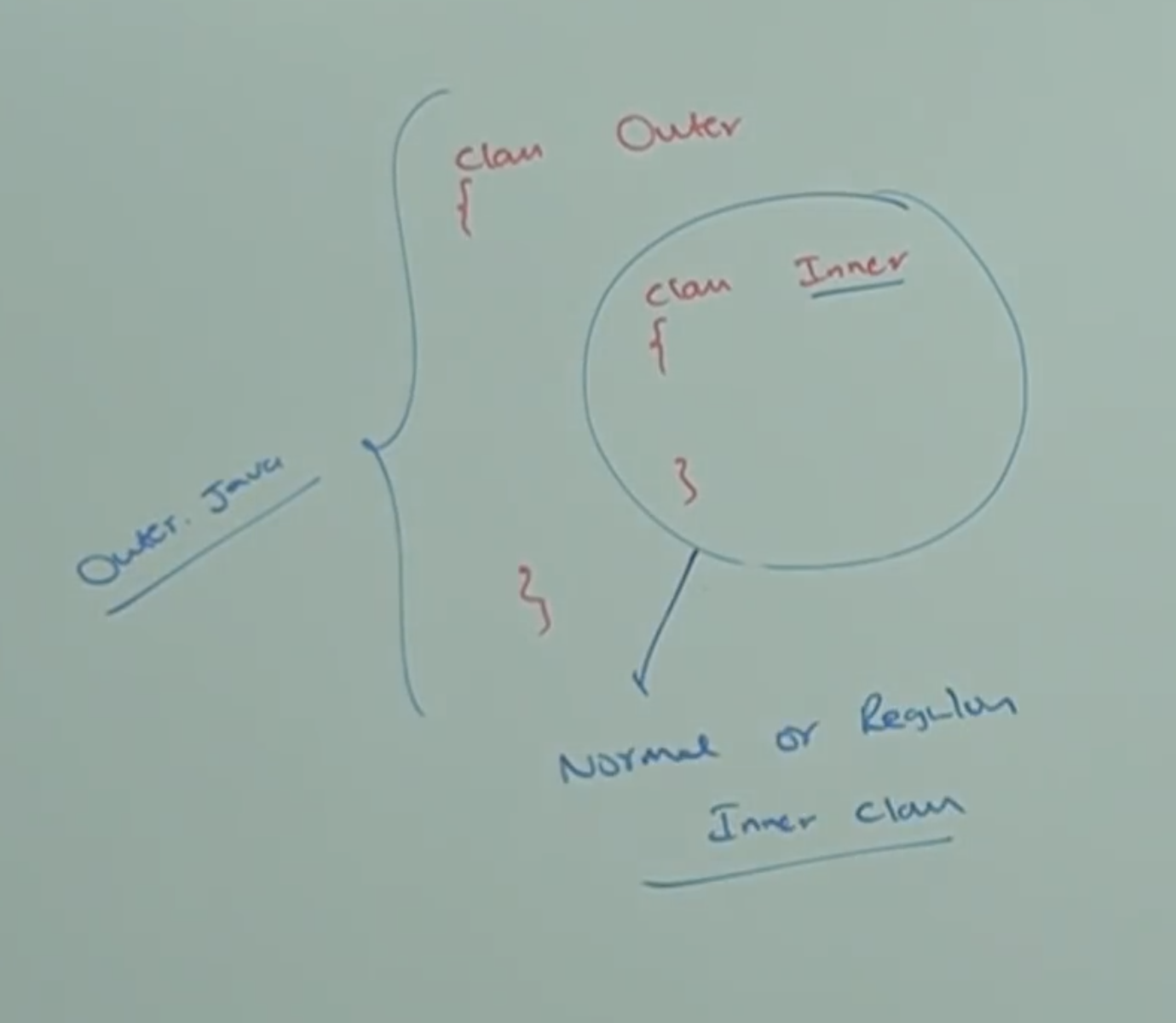
Based on position of declaration and behavior all inner-classes are divided into four types.

1. Normal or regular inner-classes
2. Method Local inner-classes
3. Anonymous inner-classes
4. Static nested classes

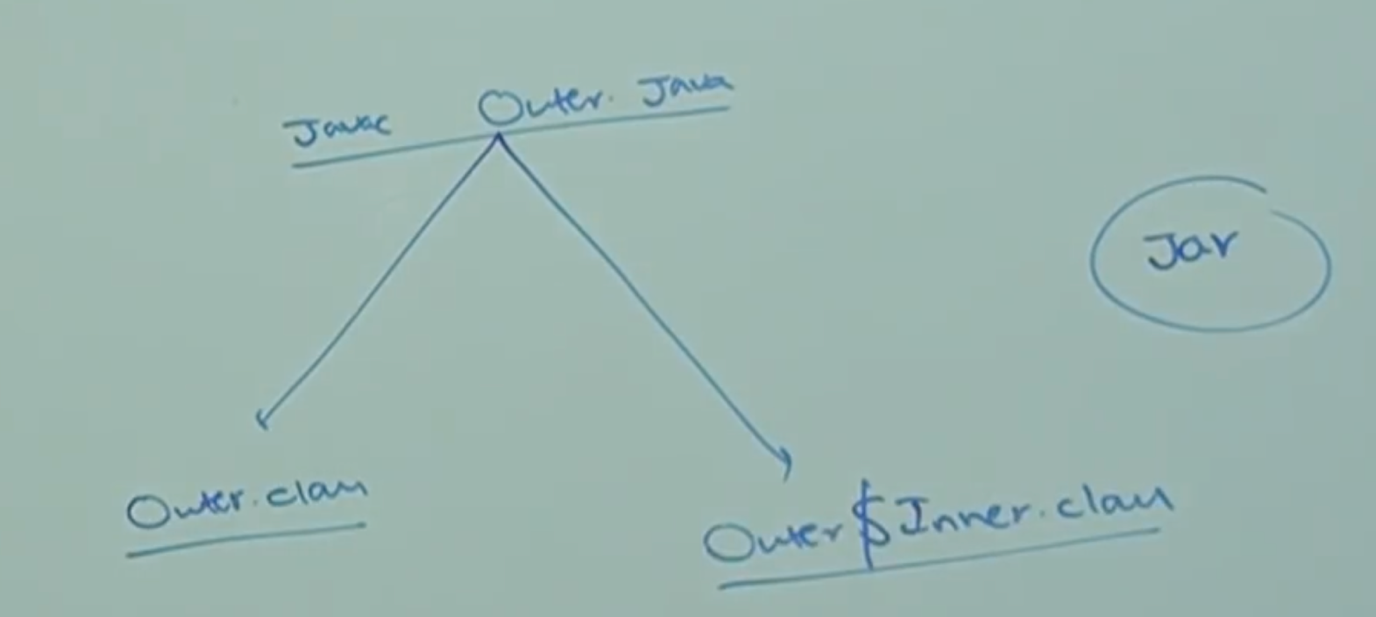
* **Normal or Regular Inner-Classes:**

If we are declaring any named class directly inside a class without static modifier such type of inner class is called normal or regular inner class.

Example\_1:



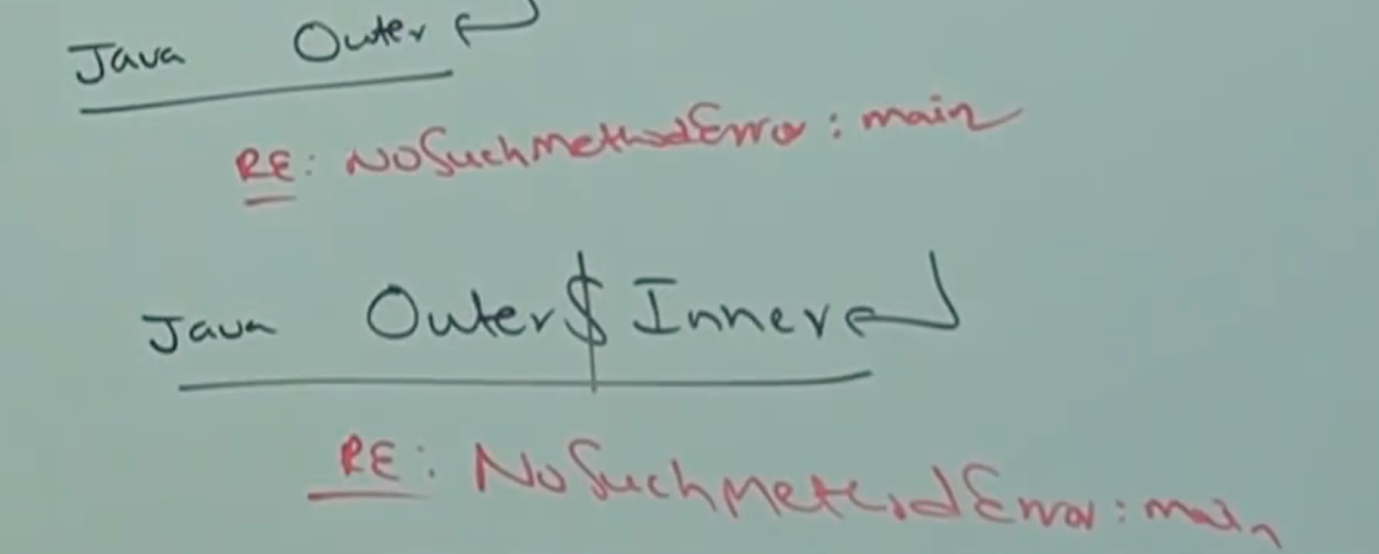
**Inner class file names after compilation:**

****

**Both the inner class doesn’t contain main method:**

So, if we run any of the class we will get the below error:

**NoSuchMethodError**

****

**Example\_2:**

class Outer{

class Inner{

}

public static void main(String[] args){

System.out.println(“Outer class main method”);

}

}

java Outer

Output: Outer class main method

java Outer$Inner

RE: NoSuchMethodError

**Example\_3:**

Inside inner-class, we can’t declare any static members hence we can’t declare main method and we can’t run inner class directly from command prompt.

Note: Inner class talks about object, without main object I can’t access inner object, so within inner class static is not allowed.

class Outer{

class Inner{

public static void main(String[] args){

System.out.println(“Inner class main method”);

}

}

}

javac Outer.java

CE: Inner classes cannot have static declartions

* **Case\_01: Accessing Inner class code from static area of outer class:**

class Outer{

class Inner{

System.out.println(“Inner class method”);

}

public static void main(String[] args){

Outer o = new Outer(); // Line-01

Outer.Inner i = o.new Inner(); //Line-02

i.m1(); // Line-03

}

}

Combining Line-01 & 02:

Outer.Inner i = new Outer().new Inner();

Combining Line-01 & 02:

new Outer().new Inner().m1();

* **Case\_02: Accessing inner-class code from instance area of outer class:**

class Outer{

class Inner{

public void m1(){

System.out.println(“Inner class method”);

}

}

public void m2(){

Inner i = new Inner();

i.m1();

}

public static void main(String[] args){

Outer o = new Outer();

o.m2();

}

}

* **Case\_03: Accessing inner class code from outside of Outer class:**

class Outer{

class Inner{

public void m1(){

System.out.println(“Inner class method”);

}

}

}

class Test{

public static void main(String[] args){

Outer o = new Outer();

Outer.Inner i = o.new Inner();

i.m1();

}

}

* **Summary of all the above cases:**

