DAY-50

------

INNER CLASSES

------------------------------

eg:1

-----

class A

{

}

class B

{

}

eg:2

----

class A

{

class B

{

}

}

eg:3

----

class A

{

class B

{

class C

{

}

}

}

Inner class : The class declared inside a another class is called as INNER CLASS.

eg:1

----

class Bike

{

class Engine

{

}

}

eg:2

----

class Mobile

{

class Battery

{

}

}

NOTE: Outer class and inner class is not [is-a] relationship.It is always [has - a ] relationship.

Types of INNER CLASSES based on positional declaration.

--------------------------------------------------------

1. Normal or Regular inner classes

2. Method local inner classes.

3. Ananymous inner classes.

4. Static nested classes.

NORMAL INNER CLASSES:

----------------------

A class declared within a another class directly without the static modifier is called as regular inner class/normal inner class.

eg:1

----

class Outer

{

class Inner

{

}

}

HOW THE CLASS FILE IS GENERATED IN CASE OF INNER CLASSES?

--> javac Outer.java

Outer.class

Outer$Inner.class

EXAMPLE:

---------

class Outer

{

class Inner

{

}

public static void main(String[] args)

{

System.out.println("Outer class main method");

}

}

OUTPUT:

--------

D:\sep\_batch\javaPrograms\DAY-50>javac Outer.java

D:\sep\_batch\javaPrograms\DAY-50>java Outer

Outer class main method

EXAMPLE:2

---------

class Outer1

{

class Inner1

{

public static void main(String[] args)

{

System.out.println("inner class main method");

}

}

}

OUTPUT:

-------

COMPILATION ERROR

NOTE: Inside inner class we can not declare static members hance main method can not be decalred inside inner class.

EXAMPLE:3

---------

class Outer2

{

class Inner2

{

public void fun1()

{

System.out.println("inside the inner fun1");

}

}

public static void main(String[] args)

{

/\*Outer2 o = new Outer2();

Outer2.Inner2 i = o.new Inner2();

i.fun1();\*/

/\*Outer2.Inner2 i = new Outer2().new Inner2();

i.fun1();\*/

Outer2.Inner2 i = new Outer2().new Inner2().fun1();

}

}

OUTPUT:

-------

inside the inner fun1

CASE 1:

-------

Accessing inner class code from static area of outer class []

[refer above example]

CASE 2:

-------

Accessing the inner class code from the instance area of outer class.

EXAMPLE:

--------

class Outer3

{

class Inner3

{

public void fun1()

{

System.out.println("inside inner class fun1 method");

}

}

public void fun2()

{

Inner3 i = new Inner3();

i.fun1();

}

public static void main(String[] args)

{

Outer3 o = new Outer3();

o.fun2();

}

}

OUTPUT:

--------

inside inner class fun1 method

CASE 3: Accessing the inner class code from Outside the Outer class.

EXAMPLE:

--------

class Outer4

{

class Inner4

{

public void fun1()

{

System.out.println("inside inner class fun1 method");

}

}

}

class Demo

{

public static void main(String[] args)

{

Outer4 o = new Outer4();

Outer4.Inner4 i = o.new Inner4();

i.fun1();

}

}

OUTPUT:

-------

inside inner class fun1 method

conclusion for cases:

----------------------

1. we can access inner class code from static area of outer class

2. we can access inner class code from instance area of outer class

3. we can access inner class code from outside the outer class

EXAMPLE:

--------

class Outer5

{

int x = 10;

static int y = 20;

class Inner5

{

public void fun1()

{

System.out.println(x);

System.out.println(y);

}

}

public static void main(String[] args)

{

Outer5.Inner5 i = new Outer5().new Inner5();

i.fun1();

}

}

OUTPUT:

-------

10

20

NOTE: From the regular inner class we can access both static and non-static members of outer class directly.

[we cant decalare static members but we can access them]

EXAMPLE:

--------

class Outer6

{

int x = 10;

class Inner6

{

int x = 20;

public void fun1()

{

int x = 30;

System.out.println(x);//30

System.out.println(this.x);//20

System.out.println(Inner6.this.x);//20

System.out.println(Outer6.this.x);//10

}

}

public static void main(String[] args)

{

Outer6.Inner6 i = new Outer6().new Inner6();

i.fun1();

}

}

OUTPUT:

------

30

20

20

10

NOTE: Within the inner class 'this ' keyword always refer current inner class object if we wnat to refer cuurent outer class object we have to use

outerClassName.this

NOTE: For outer class the access modifiers applicable are --> public,default,final,abstract,strictfp

For the inner classes the access modifers which are applicable are --> private,protected,static

NESTED CLASSES:

---------------

Class declared within inner class is called as 'NESTED CLASSES'

eg:

---

class A

{

class B

{

class C

{

public void fun1()

{

S.o.p("inside fun1");

}

}

}

}

class Demo

{

p s v m(---)

{

A a = new A();

A.B b = a.new B();

A.B.C c = b.new C();

}

}

refer dia:1 for inner and nested classes