DAY-51 [INNER CLASSES]

------

2. METHOD LOCAL INNER CLASSES:

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

--> Declaring a class inside a method is called as 'Method local inner classes'.

eg:

---

class Demo

{

public void fun1()

{

public void fun2()

{

s.o.p("inside fun2");

}

}

}

output:

-------

compilation error

NOTE: Nested methods is not permitted in java.

why method local inner classes are used?

--> To define method specific repeatedly required functionalities.

EXAMPLE:

--------

class Demo

{

public void fun1()

{

class Inner

{

public void sum(int x, int y)

{

System.out.println("the sum of x and y is : "+(x+y));

}

}

Inner i = new Inner();

i.sum(10,20);

}

public static void main(String[] args)

{

Demo d = new Demo();

d.fun1();

}

}

OUTPUT:

-------

the sum of x and y is : 30

NOTE: Scope of MLIC is lesser when compare with another inner classes.Hence it is rarely used.

NOTE: We can decalre MLIC inside both instance and static method.

EXAMPLE:

--------

class Demo1

{

int x = 10;

static int y = 20;

public void fun1()

{

class Inner1

{

public void fun2()

{

System.out.println(x);

System.out.println(y);

}

}

Inner1 i = new Inner1();

i.fun2();

}

public static void main(String[] args)

{

Demo1 d = new Demo1();

d.fun1();

}

}

OUTPUT:

-------

10

20

NOTE: If we declare inner class directly inisde instance method from then that MLIC we can access both static and instance members of outer class

directly.

EXAMPLE:

--------

class Demo1

{

int x = 10;

static int y = 20;

public static void fun1()

{

class Inner1

{

public void fun2()

{

//System.out.println(x);

System.out.println(y);

}

}

Inner1 i = new Inner1();

i.fun2();

}

public static void main(String[] args)

{

Demo1 d = new Demo1();

d.fun1();

}

}

OUTPUT:

-------

20

NOTE: If we declare inner class directly inisde static method from then that MLIC we can access only static members of outer class

directly.

NOTE: From MLIC we cant access local variables of the method in which we declare inner class.If the local variable is declared as 'final' then we can

access (till 1.6 version)

EXAMPLE: [WRT 1.8 VERSION]

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

class Demo2

{

public void fun1()

{

int x = 10;

class Inner2

{

public void fun2()

{

System.out.println(x);

}

}

Inner2 i = new Inner2();

i.fun2();

}

public static void main(String[] args)

{

Demo2 d = new Demo2();

d.fun1();

}

}

OUTPUT:

-------

10

3. ANONYMOUS INNER CLASSES:

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

Declaring the inner classes without name is called as ANONYMOUS INNER CLASS.

There are 3 types of AIC based on declaration and behaviour:

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

1. AIC that extends a class

2. AIC implements an interface.

3. AIC that defiend inside arguments

SYANTAX :

--------

class\_Name c = new class\_Name(); --> obejct creation

class\_name c = new class\_name() --> anonymous inner class creation

{

};

AIC that extends a class

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

eg:

---

class Biryani

{

public void taste()

{

s.o.p("hyderabad biryani spicy");

}

;;;;;;//// 100 methods

}

class MughalBiryani extends Briyani

{

public void taste

{

s.o.p("mughal biryani normal")

}

}

note: in the above example we have to override the methods multiple times instead of that we can make use of AIC.

EXAMPLE:

---------

class Biryani

{

public void taste()

{

System.out.println("hyderabad biryani spicy");

}

//;;;;;;//// 100 methods

}

class Demo3

{

public static void main(String[] args)

{

Biryani b = new Biryani()

{

public void taste()

{

System.out.println("less spicy");

}

};

b.taste();

Biryani b1= new Biryani();

b1.taste();

}

}

OUTPUT:

-------

less spicy

hyderabad biryani spicy

where is the AIC class file is generted?

--> always outerclass name $ AIC name .class file is generated.

in the above program the AIC class file is:

Demo.class

Biryani.class

Demo$1.class

AIC implements an interface.

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

defining Thread by extending thread class --> multithread concept

AIC that defiend inside arguments

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

defining Thread by implementing runnable interface --> multithread concept

STATIC NESTED CLASSES:

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

Declaring the inner classes with static modifier is called as static nested classes

eg:

---

class Demo

{

int x = 10;

static y = 20;

}

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

class Outer

{

class Inner --> instance

{

}

static class nested --> static

}

NOTE: for regular class, inner classes can not exist without outer classs beacuse inner class object is strongly associated with outer class.

In static nested class it can exist without outer class because static members exist without object.

EXAMPLE:

--------

class Outer

{

static class Nested

{

public void fun1()

{

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

}

}

public static void main(String[] args)

{

Nested n = new Nested();

n.fun1();

}

}

OUTPUT:

-------

Error: Could not find or load main class Nested

NOTE: To create nested class object outside the outer class it can be created as shown below:

--> Outer.Nested n = new Outer.Nested();

EXAMPLE:

--------

class Outer

{

static class Nested

{

public void fun1()

{

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

}

}

public static void main(String[] args)

{

Outer.Nested n = new Outer.Nested();

n.fun1();

}

}

OUTPUT:

-------

inside fun1

NOTE: In normal inner classes we cant declare any static members but we can declare static members in static nested classes

EXAMPLE:

-------

class Outer1

{

static class Nested

{

public static void main(String[] args)

{

System.out.println("hi");

}

}

public static void main(String[] args)

{

System.out.println("welocme to study online");

}

}

EXAMPLE:

--------

class Demo4

{

int x =10;

static int y = 20;

static class Nested

{

public void fun1()

{

System.out.println(x);

System.out.println(y);

}

}

}

OUTPUT:

------

error: non-static variable x cannot be referenced from a static context

System.out.println(x);

NOTE: from regular inner classes we can access both static and instance members of outer class directly.

BUT,

from the static nested classes we can access only the static members of outer class and we cant access the instance members outer class.

DIFFERENCE B/W regular inner classes and static nested classes:

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

REGUALR INNER CLASSES:

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

1. without existing outer class object inner class object can not be created.

2. we cant declare static members

3. we cant declare main method

4. we can access both static and instance members of outer class directly

STATIC NESTED CLASSES:

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

1. without existing outer class object inner class object can be created.

2. we can declare static members

3. we can declare main method

4. we can access only static members of outer class directly

possible combinations w.r.t inner classes:

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

1. class within a class

2. interface within a class

3. interface within a another interface

4. class inside a interface