22/02/11

Inner Classes

→ Sometimes we can declare a class inside another class, such type of classes are called "Innerclasses".

difficult or doubtful situation.

→ Innerclasses concept introduced in Java1.1 version to fix GUI bugs as the part of Eventhandling.

→ But because of powerfull features & benefits of Innerclasses slowlly programmers started using even in regular coding also.

→ without existing one type of object if there is no chance of existing another type object, then we should go for Innerclass concept.

Ex(1):-

① without existing car object, if there is no chance of existing wheel object then we should go for Inner classes.

② we have to declare wheel class with in the car class.

```
class Car
{
 class Wheel
 {
 }
}
```

②:- without existing Bank object there is no chance of existing account object, Hence we have to define account class inside Bank class.

```
class Bank
{
  class Account
  {
```

(3) —A map is a Collection of Key value pairs and each Key-value pair is called Entry. without existing map object there is no chance of existing Entry object. hence interface Entry is define inside Map interface.

```
interface Map
{
  interface Entry
   {
   }
}
```

Note:-

→ The Relationship b/w Outer & inner classes is not parent tochild Relationship. It is has-A Relationship.

→ Based on the purpose & position of declaration all innerclasses are divided into 4 types

1) Normal (or) Regular Inner classes.
2) Method Local Inner classes
3) Annoymous Inner classes (without class name)
4) Static Nested Classes

Note:-

From Static Nested Class we can access only static members of outer Class directly. But in normal Inner Classes we can access both static & Non-static members of outer class directly.

Normal or Regular Inner class :-

→ If we declare any named class directly Inside a class without Static modifier, Such type of class is Called "Normal or regular Inner class"

Ex(1):-

```
Class Outer
{
  Class Inner
  {

  }
}
```

$$\begin{tikzcd}
 & \text{Javac Outer.Java} \arrow[dl] \arrow[dr] & \\
\text{Outer.class} & & \text{Outer\$Inner.class}
\end{tikzcd}$$

Java Outer ↵

R.E:- NoSuchMethodError : main

Java Outer$Inner

R.E:- NoSuchMethodError : main

Ex(2):-

```
Class Outer
{
  Class Inner
  {
  }
  public static void main(String [] args)
  {
    S.o.pln(" Outer class main method");
  }
}
```

O/P. Javac Outer.java

Java Outer ↵

O/P!. Outer Class main method.

Java Outer$Inner ↵

O/P!- NoSuchMethod Error : main.

Ex(3):

→ Inside Inner classes we can't declare static members hence it is not possible to declare main method & hence we can't invoke inner class directly from Command prompt.

Ex:-
```
class Outer
{
 class Inner
 {
  p.s.v.m(String [] args)
  {
   S.o.pln("inner class method");
  }
 }
}
```
✗

Javac Outer.java

C.E:- Inner Classes can't have static declarations

23/02/11:-

Accessing Inner class Code from Static area of Outer Class:-

Ex:-
```
class Outer
{
  class Inner
  {
   p.v.m1()
   {
    S.o.pln(" Inner class method");
   }
  }
 p.s.v.m (String [] args)
 {
  Outer o = new Outer();
  Outer.Inner i = O.new Inner();
```

```
            i.m1();
        }
    }
```

O/P:- Javac Outer.java ↵

java Outer ←

Inner class method.

```
Outer o = new Outer();                      ─→ Outer.Inner i = new Outer().new Inner();
Outer.Inner i = o.new Inner();
i.m1();                                     ─→ new Outer().new Inner.m1();
```

Accessing Inner class Code from Instance Area of outer Class:-

Eg.

```
Class Outer
{
  class Inner
  {
    p.v.m1()
    {
      S.o.pln("Inner class method");
    }
  }
  p.v.m2()
  {
    Inner i = new Inner();
    i.m1();
  }
  p.s.v.m (String [] args)
  {
    Outer o = new Outer();
    o.m2();
} }
```

Accessing Inner class Code from Outside of outer Class :

Eg :

```
Class Outer
{
  Class Inner
  {
   p.v.m1()
   {
    S.o.pln("Inner Class method");
   }
  }
}
```

```
Class Test
{
 p.s.v.m(String [] args)
 {
  Outer o = new Outer();
  Outer.Inner i = o.new
                 Inner();
  i.m1()
 }
}
```

According Inner Class Code

from static area of Outer class
(or)
from outside of outer class

```
Outer o = new Outer();
Outer.Inner i = o.new Inner();
i.m1();
```

from Instance are of
outer class

```
Inner i = new Inner();
i.m1();
```

```
outer o = new Outer();
```

→ from the Inner class we can access all members of outer class (both static & non-static) directly.

Ex:-
```
class Outer
{
    static int x=10;
    int y = 20;
    class Inner
    {
        public void m1()
        {
            S.o.pln(x);      10
            S.o.pln(y);      20
        }
    }
    p.s.v.m (String [] args)
    {
        new Outer().new Inner().m1();
    }
}
```
✓

o/p:-
```
10
20
```

→ With in the Inner class this always pointing to current Inner class object.

→ To refer current outer class object we have to use " Outerclassname.this "

" Outerclassname.this ".

Ex:-

Eg:-

```
Class Outer
{
  int x=10;
  class Inner
  {
    int x=100;
    public void m1()
    {
      int x=1000;
      S.o.pln(x) ; 1000
      S.o.pln(this.x) ; 100
      S.o.pln(outer.this.x) ; 10
    }
  }
  p.s.v.m(String [] args)
  {
    new Outer().new Inner().m1();
  }
}
```

→ for the Outer classes (Top-level classes) the applicable modifiers are public, <default>, final, abstract, strictfp.

But for the Inner classes In addition to above The following modifiers are also applicable.

only for outer classes

public default final abstract strictfp	+	private protected static	=	Inner classes

2) Method Local Inner classes :-

→ Some-times we can declare a class inside a method such type of classes are called "Method Local Inner classes".

→ The main purpose of method local Inner class is to define method specific functionality.

→ The scope of method Local Inner class is the method in which we declared it. That is from outside of the method we can't access method Local Inner classes.

→ As the scope is very less, this type of Inner classes are most rarely used Inner classes.

Ex:
```
    class Test
    {
      public void m1()
      {
        class Inner
        {
          public void sum( int x, int y)
          {
            S.o.pln ("sum is :"+ (x+y));
          }
        }
        Inner i = new Inner();
        i. sum(10, 20);
        i. sum(100, 200);
        i. sum (1000,2000);
        i. sum(10000, 20000);
      }
```

```
p.s.v.m(String [] args)
{
  new Test().m1();
}
}
```

O/P!- Sum is 30
Sum is 300
Sum is 3000
Sum is 30000

→ We Can declare Inner class either in Instance method or in Static-method.

→ If we declare Inner class inside Instance method then we Can access Both Static & NON-Static variables of outerclass directly from that Inner class.

→ If we declare Inner class inside Static method then we Can access only Static members of OuterClass directly from that Inner class.

Ex!-
```
class Test
{
 int x=10;
 Static int y=20;
 public void m1()          if Static is there
 {
   Class Inner
   {                               o/p!- 20
     p. void m2()
     {
       S.o.pln(x);  10
       S.o.pln(y);  20
   }}
   Inner i = new Inner();
   i.m2();
 }}
```

```
P.S.V.m (String [] args)
{
 new Test().m1();
}
}
```

O/P:- 10, 20

→ from method Local Inner Class we Can't access Local variables of the method in which we declared it. But if That local variable declared as the final Then we can access.

Ex:-

```
class Test
{
 int x=10;
 public void m1()
 {
  int y=20;          → if we declare final
                       ( final int y=20;)
                       O/P:- x=10
                             y=20
  class Inner
  {
   public void m2()
   {
    System.out.println(x);
    System.out.println(y);
   }
  }
  Inner i = new Inner();
  i.m2();
 }
P.S.V.m (String [] args)
{
 new Test().m1();
}
}
```

O/P:
C.E:- Local variable y is accessed from with inner class; needs to be declared final.

→ If we declare y as final Then we won't get any Compiletime Error.

O/P:- x= 10
y= 20

24/02/11:

Q) Consider The following Code

```
class Test
{
  int x=10;
  Static int y=20;
  Public void m1()
  {
    int i =30;
    final int j=40;
  Class Inner
  {
   public void m2()
   {
   ————→ Line①
   }
  }
 }
}
```

→ At line① which Variables we Can access ① x ✓
② y ✓
③ i ✗
④ j ✓

Note:- If declare m1() as Static Then at Line① which Variables we Can access ~~as~~ y, j.

② If we declare m2() as static, then which variable we can access Line①

we will get C.E, because inside inner classes we can't have static declarations.

→ The only applicable modifiers for method local inner classes are final, abstract, strictfp,

(3) Annonymus Inner Class :-

→ Sometimes we can declare a class without name also. Such type of nameless inner classes are called Annoymus inner classes.

→ This type of inner classes are most commonly used type of inner classes.

→ There are 3 types of Annonymus inner classes.

1. Annonymus Inner class that extends a class.
2. " " " implements an interface.
3. " " " defined inside method arguements.

① Annonymus Inner class that extends a class :-

Ex:-
```
Class popCorn
{
  Public void taste()
  {
    S.o.pln("Salty");
  }
  //100 more methods
}

Class Test
{
 P.S.V.m(String []args)
 {
```

```
PopCorn p = new PopCorn
{
  public void taste()
  {
   S.o.pln("Sweety");
  }
};
P.taste(); Sweety
PopCorn P1 = new PopCorn();
P1.taste(); Salty
} }
```

Note:-

① The internal class name generated for Annoymus Inner class is "Test$1.class".

② parent class reference Can be used to hold child class object but by using that reference we Can Call only methods available in the parent class & we Can't Call child Specific methods. In the annoymus inner classes also we Can define new methods but we Can't call these method from outside of the class because, ~~these are~~ We are depending on parent reference. this methods Just for internal purpose only.

Analysis:-

```
popcorn p = new Popcorn();
```

→ Just we are Creating an object of PopCorn class.

→ ~~Pop~~

```
PopCorn p = new popCorn()
{

};
```

→ We are Creating child class for the popCorn & for that child class we are Creating an object with parent reference.

Eg:

```
Class Test
{
 p.s.v.m (String [] args)
 {
  Thread t = new Thread()
  {
   p.v. run()
   {
    for(int i=0 ; i<10 ; i++)
    {
     S.o.pln ("child Thread");
    }
   }
  };
  t.start();
  for(int i=0 ; i<10; i++)
  {
   S.o.pln ("main Thread");
  }
 }
}
```

→ In the above Example both main & child threads will be Executed Simultaneously & Hence we can't ~~Exe~~ exact output.

(b) Anonymous Inner class that implements an Interface :-

Eg :-

```
Class Test
{
 p.s.v.m(String [] args)
 {
  Runnable r =new Runnable()
  {
   public void run()
   {
    for (int i=0 ; i<10 ; i++)
    {
     S.o.pln("child thread");
    }
   }
  };
 Thread t = new Thread (r);
 t.start();
 for (int i=0 ; i<10; i++)
 {
  S.o.pln(" main Thread");
 }
}
}.
```

It is an object of Runnable

(c) Anonymous Inner class that define Inside method argument:-

Ex:-
```
Class Test
 {
  Public static void main(String [] args)
   {
    New Thread(new Runnable()
                {
                 public void run()
                 {
                  for (int i=0 ; i<10 ; i++)
                  {
                   S.o.pln ("child thread-i");
                  }
                 }
               }).start();

    for (int i=0 ; i<10 ; i++)
     {
      S.o.pln (" main thread -i");
     }
   }
 }
```

General class Vs Anonymus Inner class:-

→ A General class can extend only one class at a time. where as Annonymos Innerclass also can extend only one class at a time.

→ A General class can implement any no. of Interfaces where as Annoymus Innerclass can implement only one interface at a time.

→ A General class can Extend another class & can implement an interface simultaneously. where as Annonymus Inner class can extend another or can implement an interface but not both simultaneously.

) Static Nested classes:-

→ Some times we can declare Inner class with static modifier Such type of Inner classes are called "Static Nested classes".

→ In the normal Inner class, Inner class object always associated with outer class object.

→ ie, with out existing outer class object, There is no chance of existing Inner class object.

→ But Static Nested class object is not associated with Outerclass object i.e with out existing outer class object There may be a chance of existing Static Nested class object.

Ex:-
```
class Outer
{
 static class Nested
 {
  public void m1()
  {
   S.o.pln("static Nested class method");
  }
 }
public static void main(String[] args)
 {
  Outer.Nested   n  = new  Outer.Nested();
   n.m1()
 }
}
```

→ With in the Static Nested Class we Can declare Static members including main() also. Hence it is possible to invoke Nested Class directly from Command prompt.

Ex:-

```
Class Outer
 {
  Static class Nested
  {
    public static void main(String[] args)
     {
      S.o.pln(" Static Nested class main method");
     }
  }
  public ~~static~~ static void main(String[] args)
  {
   S.o.pln(" Outer class main method");
  }
 }
```

```
o/p: Javac Outer.java ↵
     Java Outer ↵
        Outer class main method
     Java Outer$Nested ↵
        Static Nested Class main method
```

→ from the Normal Inner class both Static & Non-Static members directly.
but from, Static Nested class we Can access only Static members of outer class directly.

Ex:-
```
class Outer
{
 int x=10;
 Static int y=20;

 Static class Nested          ×
 {
   p.v.m()
   {
     S.o.pln(x); ×  ——→ C.E:- Non-Static variable x can't be
                        referenced from Static ~~class~~ ~~Nested~~ Content
     S.o.pln(y); ✓
   }
 }
}
```

diff b/w Normal Innerclass & Static Nested Class ?

Innerclass	Static Nested class
1) Inner class object is always associated with Outerclass object. i.e without existing outerclass object there is no chance of existing Innerclass object	1) Static Nested Class object is not associated with Outer class object, i.e, without existing Outer class object there may be a chance of existing Static Nested class object.
2) Inside Normal Inner class we can't declare Static members	2) Inside Static Nested class we Can declare Static members.
3) Inside normal Inner class we can't declare main() and hence it is not Possible to invoke inner class directly from Commandprompt	3) Inside Static Nested class we Can declare main() & hence we Can invoke Static Nested Class directly from Command Prompt.