Inner Classes

Sometimes we can declare a class inside another class, such type of classes are called Inner Classes.

Inner Classes concept was introduced in java 1.1 version to fix GUI bugs as the part of Event handling.

->But because of powerful features and benefits of inner classes slowly 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 Inner class concept.

Ex. Without existing car object if there is no chance of existing wheel object then we should go for Inner Classes.

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 pass and each key-value pass is called Entry. Without existing map object there is no chance of existing entry object. Hence interface entry is defined inside map interface.

interface Map

{

interface Entry

{

}

}

Note:-

->The relationship between outer and inner classes is not parent to child relationship. It is has-A relationship.

->Based on the purpose and position of declaration all inner classes are divided into 4 types.

1)Normal or regular Inner Classes

2)Methods 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 and 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.class outer

{

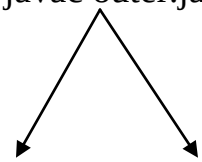
Class Inner

{

}

}

javac outer.java



outer class outer/Inner class

Java outer

R.E:-NosuchmethodError=main

java outer/Inner

R.E.Nosuchmethod error=main

Ex.

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:-NoSuchMethodError=main

ex(3).

->Inside Inner classes we can’t declare static members hence it is not possible to declare main method and hence we can’t invoke inner class directly from command prompt.

Ex.

class outer

{

class Inner

{

p.s.v.m1()

{

s.o.pln(“Inner class Method”);

}

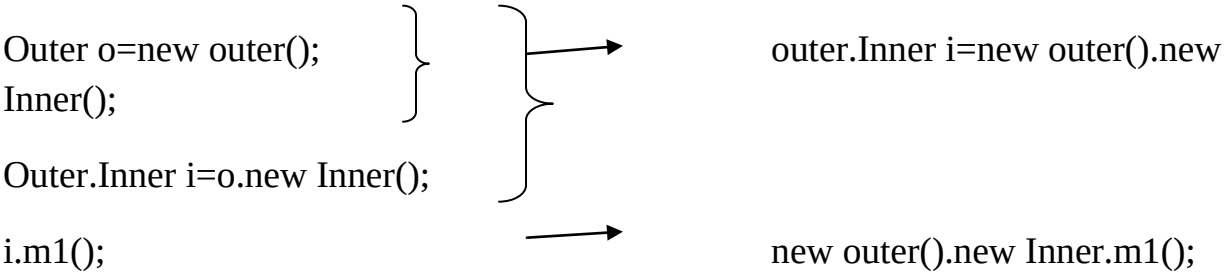
}

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

o/pJavac outer.java

java outer

Inner class Method



Accessing Inner class code from Instance area of outer class:-

Ex.

```
 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:-

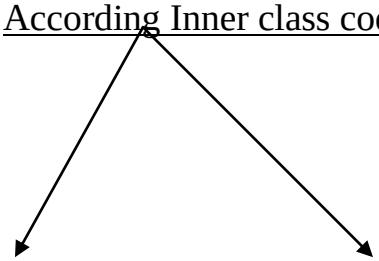
Ex1.

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

Ex2.

```
class Test
{
p.s.v.main(Sring [] args)
{
Outer o=new outer();
Outer.inner i=o.new Inner();
i.m1();
}
}
```

According Inner class code



From static area of outer class

from Instance area of outer class

From outside of outer class

Outer o=new outer();

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

i.m1();

Inner i=new Inner();

i.m1();

outer o=new outer();

->from The inner class we can access all member of outer class (both static and non-static) directly

Ex.

```
class outer
{
Static int x=10;
int y=20;
class Inner
{
```

```
public void m1()
{
s.o.pln(r);
s.o.pln(y);
}
}
p.s.vmain(String args[])
{
New outer().new Inner().m1();
}
}
```

o/p:-

10

20

->With in the Inner class this always pointing to convert to current Inner class object.

->To refer current outer class object we have to use “Outerclassname”**.**

“outer classname.this”

Ex.class outer

{

```
int x=10;
class Inner
{
int x=100;
Public void m1()
{
int x=1000;
S.o.pln(x);   //1000
So.pln(this.x);   //100
S.o.pln(outer.this.x);        //10
}
}
p.s.v.main(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.

Public	private
Default	+ protected=Inner classes
Final	static
Abstract	
Strictfp	

2)Method Local Inner Classes:

->Some times we can declare classes inside a method such type of class are called "method local Inner classes".

->The main purpose of method local Inner classes is to define method specific functionality.

->The scope of the methods local Inner class is the method in which we declared it that is from outside of the methods we can't access methods 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.main(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 and non-static variable of outerclass directly from that Inner class.

->If we declare Innerclass inside the static method then we canaccess only static members of outerclass directly from that innerclass

Ex.

```
class Test
{
int x=10;
Static int y=20;
public void m1()
{
class Inner
{
p.v.m2()
{
S.o.pln(x);      //10
S.o.pln(y)//20
}
}
Inner i=new inner();
i.m2();
```

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

o/p:

10,20

->From method local Inner class we can't access local variable and 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;
class Inner
{
public void m2()
{
S.o.pln(x);
S.o.pln(y);
}
}
Inner i=new Inner();
```

```
i.m2();
}
p.s.v.main(String args[])
{
New Test().m1();
}
}
```

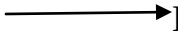
C.E:- local variable y is accessed from with inner class needs to be declared final.

->If we declare y as final then we wan't get any compile time error.

o/p:
x=10;
y=20;

Q)consider the fallowing code

```
Class Test
{
int x=10;
Static int y=20;
Public void m1()
{
int i=80;
Final int i=40;
class Inner
{
public void m2()
{
```



```
Line(1)
}
}
}
}
```

->At line(1) which variable we can access (1) x //write
(2) y //write
(3) I //wrong
(4) j //write

Note:-If declare m1() as static then at line(1) which variables
We can access y,j;

(3)If we declare m2() as static ,then which variable we can access line(1) we will get C.E. because Inside Inner classes we can't have static declarations.
->The only applicable modifiers for methods local Inner classes are final,abstract,strictfp.

(3) Anonymous Inner class:-
->Sometimes we can declare a class without name also such type of normal Inner classes are called Anonymous Inner classes
->This type of Inner classes are most commonly used type of Inner classes
->There are 3 types of Anonymous Inner classes…
1. Anonymous Inner class that extends a class
2. Anonymous Inner class that implements an interface
3. Anonymous Inner class that defined inside methods arguments

(a) Anonymous Inner class that extends a class:-

```
Class popcorn
{
Public void tester()
{
S.o.pln("salty");
}
//100 more methods
}
class Test
```

```
{
p.s.v.main(String[] args)
{
Popcorn p=new popcorn
{
public void tester()
{
s.o.pln("swalty");
}
};
p.tester();              //swalty
popcorn p1=new popcorn();
p1.tester;                    //salty
```

Note:-

(1)The internal class name generated for anonymous Inner class is "Test t1.class".

(2)Parent class reference can be used to hold child class object but by using reference we can call only methods available in the parent class and we can't call child specified methods. In the anonymous inner classes also we can define new methods but we can't call these method form outside of the class because, we are depending on parent reference. This methods for internal purpose only.

Analysis:-

Popcorn p=new popcorn();

->Just we are creating an object of popcorn class

Popcorn p=new popcorn()

{

};

->We are creating child class for the popcorn and for that child class we are creating an object with parent reference.

```
class Test
{
p.s.v.main(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 and child threads will be executed simultaneously and hence we can't exact output

(b)Annonymous Inner Class That implements an Interface:-

```
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); //It is an object and runnable
t.start();
for(int i=0;i<10;i++)
{
```

```
s.o.pln("main Thread");
}
}
}
```

(b) Annonymous Inner class that define Inside the method arguments:-

```
class test
{
p.s.v.main(String[] args)
{
New Thread(new Runnable()
{
public void run()
{
For(int i=0;i<10;i++)
{
S.o.pln("child Thread-1");
}
}
} ).start();
For(int i=0;i<10;i++)
{
s.o.pln("main thread");
}
}
```

- Genaral class Vs.Anonymous Inner class.

1) A general class can extend only one class.

2) Ageneral class can implement no of interface where as anonymous inner class can implements only one interface at a time.
3) A general class can extends another clas and can implement an interface simulteneosly,Where as anonymous inner class can extend another or implemets interface but not both simultaneously.

Static nested classes

- Sometimes we can declare inner class with static modifier such type of inner class called "static nested classes".
- In the normal inner class,inner class object always associated with outer class object.
- That is without existing outer class object there no chance of existing inner class object.But static nested class object is not associated with outer class object there may be a chance of existing static nested class object

  Example:

```
class Outer
{
static class Nested
{
public void m1()
{
System.out.println("static nested class method");
}
public static void main(String args[])
{
   Outer.Nested n=new Outer.Nested();
    n.m1();
 } }
```

- Within the static nested class we can declare static member including main() also. Hence it is possible to invoke nested class directly from command prompt.

  Example:

```
 class  Outer
  {
static class Nested
    {
public static void main(String argd[])
       {
         System.out.println("Static nested class main method");
       }
    }
public static void main(String argd[])
    {
System.out.println("outer class main method");
     }
  }
```

  O/P:Javac Outer.java

```
      Inner Outer
      Outer class main method
      Java OuterNested
      Static nested class main method
```

- From the normal inner class both static and non-static member directly but from static nested class we can access only static members of outer class directly.

  Example:

```
class outer
{
int x=10;
static int y=20;
static class nested
{
public void m1()
{
```

```
System.out.println(x);   ///C.E. non-static variable x can't be
referenced from static content.
System.out.println(y);
}
}
}
```

Difference between Normal inner class and static Nested class:

Normal inner class	Static Nested class
1)Inner class object is always associated with outer class object.That is without existing outer class object there is no chance of existing inner class object	1) static nested class object is not associated with outer class object.That is without existing outer class object there may be chance of existing static nested class object
2)Inside Normal inner class we can't declare static members.	2)Inside static nested class we candeclare static members.
3)Inside Normal inner class we can't declare main() and hence it is not possible to invoke inner class directly from command prompt.	3)Inside static nested class we can declare main() and hence we can invoke static nested class directly from command prompt.