Thursday, June 16, 2022 9:33 AM

Class ---- independent structure

```
class A
{
}//end of class A

class B
{
}//end of class B

class C
{
}
```

X = outer class
Y = inner class

```
class  X
{
        class Y
        {
        }//end of class Y

}//end of class X
```

Advantage of inner class = it automatically gets the "this" of outer class and it can access All(private , default , protected, public) properties and methods of the outer class with any restriction.

Exercise = study.withoutinner

```
class Test
        private int data=100;


class User
{
        Private int data2=101

        Public void show()
        {
                Sysout  data1 and data2
        }

        Main
                Call the show()
```

}

study.usinginner

1. inner class gets the "this" of outer class IMPLICITLY
2. To distinguish between "this" of inner and "this" of outer in the inner class
   this.data1 and Test.this.data1
3. Outer class DOES NOT GET the "this" of inner class IMPLICITLY , so to use inner class properties or methods the outer class must EXPLICITLY create the object of inner class
4. To get the object of inner class in main
   First create object of outer class , then using it create object of inner class

---

Static Inner class

Exercise = study.useinner

```
Class Test2
{

    //does not get the "this" of Test2 but it can access static properties and
    methods of Test2
    static class Inner
    {
    }


}//end of class
```

Anonymous Inner Class =

We can declare these classes inside a method !!!

This is the traditional way to do it ----------

```
interface I1
{
    void doJob();
}


class Abc implements I1
{
    @Override
    void doJob()
    {
     ……..
    }
}

Class User
{
    p.s.v.main(String[] args)
```

```
        {
                I1  obj = new Abc();
                 obj.doJob();
        }
}
```

Using Anonymous inner class we can skip class creation part ----------

```
interface  I1
 {
        void doJob();
 }


Class User
{
        p.s.v.main(String[] args)
        {
                I1  obj = new I1() { //anonymous inner class that implements interface I1
                        @Override
                         void doJob()
                         {
                           ....
                         }
                };

                obj.doJob();
        }
}
```

Anonymous inner class ----

The class implements the interface . The class has no name , so the compiler internally gives it some name for example ---- User$1.class

We can create only one object of a given anonymous class!!!!!!

I want a thread that prints hello 10 times!!!!
I want another thread that prints Hi 20 times !!!

Use Anonymous inner class

Method Local Inner class - this is a named inner class within a method !!!

4.30 pm to 6.30pm !!!

Lambda Expression ------ Arrow Function ---- FAT Arrow Notation ---

Shorthand notation !!!

1. Similar to Anonymous inner class

2. But it Implements a special interface called as FUNCTIONAL interface only.
   ---- the interface having exactly ONE method !!

3. when there are short implementations of the interface , with one time usage then go for lambda
   When u have complex long implementations you can write in traditional way.

Collections ---- Stream API----using Lambda expression is convenient !!!

1. First we create ArrayList <String> al

Stream obj = al.stream();

Stream interface has a method forEach

obj.forEach(object of Consumer Interface)

forEach API
Calls consumer interface method accept for each value in al

```
class XYZ implements Consumer
{
    @Override
    Void accept( String s)
    {
        //process the element of the AL
        Sysout ( s)
    }
}
```

obj.forEach(new XYZ());

//if al has 15 elements then forEach will call accept method 15 times.

Reflection ---------- using class Class to get info about the object / to create object without constructor call or to call methods without knowing method name while writing code / access private fields of objects

PEEPING inside an object and finding details .

Special class is used ----- name of the class is **Class**
it is java.lang.Class

_____
When our program runs
all .class files are loaded in the class area in JVM RAM !!!!
|
After loading them , the **JVM will create** a class Class object for each .class !!!
|
This class Class object has all METADATA of the class

---

class Class is HEAVILY used by IDEs like eclipse , Debuggers, Servers like tomcat , Containers like Spring , Hibernate

---

HW --

1. Write a simple inner class

```
Outer class User
        Non static String name
         non static int day , int month , int year

         class AgeCal
         {
                  Void showAge()
                  {
                        Use GregorianCaledar and calculate age and show it
                  }
         }

         Main
         {
                Call the showAge()

         }
```

2. Write a class that has main

```
create arraylist of Student
        Using forEach show the name of each student in uppercase

        Using filter --show all student roll, name having roll > 10
```

3. Type the Reflection code Again as done in class!!!