Dt : 8/11/2022

(i)private static reference variable:

=>private static reference variable will hold the reference of object created inside the class.

(ii)private Constructor:

=>private constructor will restrict the object creation from externally.

(iii)static method:

=>static method is used to access the object reference outside the class

---

=>Based on Object creation process the "SingleTon class design pattern" is categorized into two types:

(a)Early Instantiation process

(b)Late Instantiation process

(a)Early Instantiation process:

=>In Early Instantiation process the object is created using static block.

Ex:

PDisplay.java

```
package test;
public class PDisplay {
    private int k=200;
    private PDisplay() {}
    private static PDisplay ob = null;
    static
```

```
    {
     ob = new PDisplay();//Con_Call
    }
    public static PDisplay getReference() {
     return ob;
    }
    private void dis() {
     System.out.println("***private method dis()****");
     System.out.println("The value k:"+k);
    }
    public void access() {
     System.out.println("***private Variable***");
     System.out.println("The value k:"+k);
     this.dis();
    }
}
```

DemoPoly1.java(MainClass)

```
package maccess;
import test.PDisplay;
public class DemoPoly1 {
     public static void main(String[] args) {
       //PDisplay ob = new PDisplay();//Error
       //System.out.println("The value k:"+ob.k);//Error
       //ob.dis();//Error
          PDisplay ob = PDisplay.getReference();
                                   //Accessing the Object
reference
       ob.access();
     }
}
```

o/p:

****private Variable****

The value k:200

****private method dis()*****

The value k:200

(b)Late Instantiation process:

=>In Late Instantiation process the object is created using method.

(Lazy Instantiation process)

Ex:

PDisplay2.java

```
package test;
public class PDisplay2 {
    private int k=200;
    private PDisplay2() {}
    private static PDisplay2 ob = null;
    public static PDisplay2 getReference() {
     if(ob==null) {
     ob = new PDisplay2();//Con_Call
     }
     return ob;
    }
    private void dis() {
     System.out.println("***private method dis()****");
     System.out.println("The value k:"+k);
    }
    public void access() {
     System.out.println("***private Variable***");
     System.out.println("The value k:"+k);
     this.dis();
    }
}
```

DemoPoly2.java(MainClass)

```
package maccess;
import test.PDisplay2;
public class DemoPoly2 {
     public static void main(String[] args) {
       //PDisplay2 ob = new PDisplay2();//Error
       //System.out.println("The value k:"+ob.k);//Error
       //ob.dis();//Error
          PDisplay2 ob = PDisplay2.getReference();
                                        //Accessing the Object
reference
```

```
        ob.access();
    }
}
```

o/p:

******private Variable******

The value k:200

******private method dis()*******

The value k:200

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

Note:

=>In realtime "SingleTon class design pattern" is used to hold DB Connection code part of DAO(Data Access Object) layer in MVC(Model View Controller).

==========

(d)private Classes:

=>The classes which are declared with private keyword are known as private classes.

Coding Rules:

(i)private classes can be declared only as InnerClasses.

(ii)These private InnerClass objects are created inside the NonPrivate methods of same class.

Ex:

SubClass1.java

```
package test;
```

```
public class SubClass1 {
      private class SubClass2{
       public void m2(int x) {
            System.out.println("====InnerClass m2(x)====");
            System.out.println("The value x:"+x);
       }
      }//Private InnerClasss
      public void access()
      {
       SubClass2 ob2 = new SubClass2();
       ob2.m2(123);
      }
}//OuterClass
```

DemoPoly3.java(MainClass)

```
package maccess;
import test.SubClass1;
public class DemoPoly3 {
      public static void main(String[] args) {
         SubClass1 ob1 = new SubClass1();//OuterClass object
         //SubClass1.SubClass2 ob2 = ob1.new SubClass2();//Error
         ob1.access();
      }
}
```

o/p:

====InnerClass m2(x)====

The value x:123

==========

**imp*

3.final:

=>The following are the final programming components:

(a)final variables

(b)final methods

(c)final classes

=>There is no concept of final blocks,final constructors,final Interfaces and final abstract classes

(a)final variables:

=>The variables in classes which are declared with "final" keyword are known as final variables.

Coding Rule:

=>final variables must be initialized with values and once initialized cannot be modified(Secured Variables)

Note:

=>final variables in classes can be initialized using blocks or constructors.

(b)final methods:

=>The methods which are declared with "final" keyword are known as final methods.

Coding rule:

=>final methods cannot be Overrided,which means final methods cannot be replaced.

(c)final classes:

=>The classes which are declared with "final" keyword are known as final classes

Coding rule:

=>final classes cannot be extended,which means final classes cannot be inherited.

Ex:

Test.java

```
package test;
public final class Test {
	public static final int k;
	public final int z;
	public final int p;
	static
	{
	 k=200;
	}

	{
	 z=300;
	}

	public Test(int p)
	{
	 this.p=p;
	}
	public final void getData()
	{
	 System.out.println("***Display Data****");
	 System.out.println("The value k:"+k);
	 System.out.println("The value z:"+z);
	 System.out.println("The value p:"+p);
	}
}
```

DemoPoly4.java(MainClass)

```
package maccess;
import test.*;
public class DemoPoly4 {
	public static void main(String[] args) {
	  Test ob = new Test(12);//CClas_Con_Call
	  ob.getData();
	}
```

}

o/p:

****Display Data*****

The value k:200

The value z:300

The value p:12

==========

Note:

=>In realtime using final programming components we can construct "Immutable Classes".

faq:

define Immutable classes?

=>The classes which are constructed using the following rules are known as Immutable Classes.

Rule-1 : The class must be final class

Rule-2 : Variables which are declared in classes must be "private and final".

Rule-3 : The methods which are declared in classes must be only "getter methods"

Rule-4 : These "getter methods" final methods.

Note:

=>The objects which are generated from Immutable classes are known as "Immutable Objects"

```
import java.io.Serializable;
import java.util.Date;
@SuppressWarnings("serial")
public final class TransLog implements Serializable
{
   private final long hAccNo,bAccNo;
   private final double amt;
   private final Date dateTime;
   public TransLog(long hAccNo,long bAccNo,double amt,Date dateTime)
   {
         this.hAccNo=hAccNo;
         this.bAccNo=bAccNo;
         this.amt=amt;
         this.dateTime=dateTime;
   }
         public final long gethAccNo() {
               return hAccNo;
         }
         public final long getbAccNo() {
               return bAccNo;
         }
         public final double getAmt() {
               return amt;
```

```
        }
        public final Date getDateTime() {
                return dateTime;
        }

}
```

==========

=>Based on Security the Objects in Java are categorized into two types:

1.Mutable Objects

2.Immutable Objects

1.Mutable Objects:

=>The Objects once created can be modified are known as Mutable Objects.

(UnSecured Objects)

2.Immutable Objects:

=>The Objects once created cannot be modified are known as Immutable Objects.

(Secured Objects)

==========

faq:

wt is the diff b/w

(i)static constructor

(ii)private constructor

(iii)final constructor

(i)static constructor:

=>There is no concept of static constructor in Java.

(ii)private constructor:

=>private construct will restrict object creation from externally.

(iii)final constructor:

=>There is no concept of final Constructor in Java

==========

faq:

wt is the diff b/w

(i)final

(ii)finally

(iii)finalize

(i)final:

=>"final" is a keyword used to declare variables,methods and classes.

(ii)finally:

=>"finally" is a block part of exception handling process and which hold resource closing operations.

(iii)finalize:

=>finalize() is a method from "java.lang.Object" class used part of Garbage Collection process.

==========

faq:

wt is the diff b/w

(i)static method Overriding

(ii)private method Overriding

(iii)final method Overriding

=>These Overriding processes not available in Java.

==========