Dt : 16/9/2023

faq:

define SingleTon class?

=>The classes which generate only one object is known as SingleTon class.

faq:

define SingleTon class design pattern?

=>The structure which is used to construct SingleTon class is known as SingleTon class design pattern.

=>SingleTon class design pattern will use the following Components:

1.private static reference variable

2.private Constructor

3.static method

1.private static reference variable:

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

2.private Constructor:

=>private Constructor is executed when the object is created inside the class and which also restrict object creation from externally.

3.static method:

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

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

(i)Early Instantiation process

(ii)Late Instantiation process

(i)Early Instantiation process:

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

Note:

=>Early Instantiation process is used in DAO(Data Access Object) layer in MVC.

Ex:

p1 : Test1.java

```
package p1;
public class Test1
{
    private static Test1 ob=null;
    private Test1() {}
    static
    {
     ob = new Test1();
    }
    public static Test1 getRef()
    {
     return ob;
    }
    public void dis(int k)
    {
    System.out.println("****Instance-dis(k)****");
    System.out.println("The value k:"+k);
    }
}
```

p2 : DemoPoly4.java(MainClass)

```
package p2;
import p1.Test1;
public class DemoPOly4
{
    public static void main(String[] args)
    {
       Test1 ob = Test1.getRef();
       ob.dis(123);
    }

}
```

o/p:

```
****Instance-dis(k)****

The value k:123
```

(ii)Late Instantiation process:

=>In Late Instantiation process the Object is created while static method execution and which is also known as Lazy Instantiation Process.

Ex:

p1 : Test2.java

```
package p1;
public class Test2
{
    private static Test2 ob=null;
    private Test2() {}
    public static Test2 getRef()
    {
    if(ob==null)
    {
    ob = new Test2();
    }
    return ob;
    }
    public void dis(int k)
    {
    System.out.println("****Instance-dis(k)****");
    System.out.println("The value k:"+k);
    }
}
```

p2 : DemoPoly5.java(MainClass)

```
package p2;
import p1.Test2;
public class DemoPOly5
{
    public static void main(String[] args)
    {
        Test2 ob = Test2.getRef();
        ob.dis(123);
    }

}
```

o/p:

```
****Instance-dis(k)****
```

The value k:123

***==
=======***

****imp***

3.final:

=>"final" keyword in Java specifies the components must not be modified, which means restricted from modifications.

=>The following are some important 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 AbstractClasses.

(a)final variables:

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

Coding Rule:

=>final variables must be initialized with values and once initialized cannot be modified(These final variables are also known as Secured variables or Constant variables)

Note:

=>final variables in Classes can be initialized using 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 method-bodies cannot be replaced.

Note:

=>we can perform final method Overloading process.

(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 cannot be inherited.

--

Note:

=>In realtime,using final programming components we construct Immutable Classes

==