SINGLETON WITH CLONING SOLUTION

To perform cloning **clone() method** has given in object class so that all object which are clonable can inherit and use that method

Q:- why clone method is there in object class?

Ans:- clonable interface is not giving any method but there should be method to perfoem cloning in one or other place so that any object become cloneable then for all object the direct and indirectly the super class is java.lang.Ojbect class if it is there in object class so I can class in any class .

Q:- why clone method is protected method?

Ans:- because it wll visible direct sub class of the object class

“by seeing interface implementation interface will not do anything to make object as cloning ..by seeing interface implementation jvm will do this normal object as clonable object ”

“clone() is protected method of java.lang.Object It can be invoked only from direct sub classes of java.lang.Ojbect class.”

Q;- Whenever object is created through with cloning process object is crated but constructor not executed why?

Ans:- since the new object/duplicate created through cloning process will have invoking existing object state object as initial state so there is not need to for separate initialization for the object that is created through cloning so JVM does not invoke constructor.

Solving cloning problem with singleton:-

We can solve cloning problem in two ways

1. Throwing clone not supported exception
2. Return existing object at the time of creation clone of object

SINGLETON WITH CLONING SOLUTION

1. Ex:-

```java
package com.nit.cloneble;

import com.nit.printer.Printer;

public class UtilsClass implements Cloneable  {

    @Override
    public Object clone() throws CloneNotSupportedException {
        // TODO Auto-generated method stub
            return (Printer) super.clone();
        }
    }
```

```java
import com.nit.cloneble.UtilsClass;

public class Printer extends UtilsClass {

    private static Printer instence;

    //private constructor
    private Printer() {
        System.out.println("0 param constructor ");
        }

        //static factory method
        public static Printer getInstence() {
            if(instence==null) {
                instence=new Printer();
            }
            return instence;
        }
        public void printData(String data) {
            System.out.println(data);
        }

        @Override
        public Object clone() throws CloneNotSupportedException {
        throw new CloneNotSupportedException("cloning is not     allowed");
        }
    }
```

SINGLETON WITH CLONING SOLUTION

```java
package com.nit.test;

import com.nit.printer.Printer;

public class CloningTest {
    public static void main(String[] args) throws CloneNotSupportedException {
            Printer p=null, p1=null, p3=null,p4=null;
            p=Printer.getInstence();
            p1=Printer.getInstence();
            System.out.println("hashcode of first Printer Class="+p.hashCode());
            System.out.println("hashcode of second Printer Class="+p1.hashCode());

            System.out.println();
            p3=(Printer) p.clone();
            p4=(Printer) p3.clone();
            System.out.println("hashcode of Clonable Printer Class="+p3.hashCode());
            System.out.println("hashcode of Clonable Printer Class="+p4.hashCode());
    }
}
```

Output:-

```
Exception in thread "main" 0 param constructor
java.lang.CloneNotSupportedException: cloning is not allowed
	at com.nit.printer.Printer.clone(Printer.java:27)
	at com.nit.test.CloningTest.main(CloningTest.java:14)
hashcode of first Printer Class=366712642
hashcode of second Printer Class=366712642
```

===

SINGLETON WITH CLONING SOLUTION

2. Returning existing object :- this approach is not recommended

Ex: -

```java
package com.nit.printer;

import com.nit.cloneble.UtilsClass;

public class Printer extends UtilsClass {

    private static Printer instence;

    //private constructor
    private Printer() {
            System.out.println("0 param constructor ");
    }

    //static factory method
    public static Printer getInstence() {
            if(instence==null) {
                    instence=new Printer();
            }
            return instence;
    }
    public void printData(String data) {
            System.out.println(data);
    }

    //solving cloning problem
    //.by throwing exception (this is the best practice )

    /*@Override
    public Object clone() throws CloneNotSupportedException {
            throw new CloneNotSupportedException("cloning is not allowed");
    }*/

    //2.by returning existing object
    @Override
    public Object clone() throws CloneNotSupportedException {

            return instence;
    }
}
```

```java
package com.nit.cloneble;
import com.nit.printer.Printer;
public class UtilsClass implements Cloneable  {

    @Override
    public Object clone() throws CloneNotSupportedException {
            // TODO Auto-generated method stub
            return (Printer) super.clone();
    }
    }
```

SINGLETON WITH CLONING SOLUTION

```java
package com.nit.test;

import com.nit.printer.Printer;

public class CloningTest {
    public static void main(String[] args) throws CloneNotSupportedException {
            Printer p=null, p1=null, p3=null,p4=null;
            p=Printer.getInstence();
            p1=Printer.getInstence();
            System.out.println("hashcode of first Printer Class="+p.hashCode());
            System.out.println("hashcode of second Printer Class="+p1.hashCode());

            System.out.println();
            p3=(Printer) p.clone();
            p4=(Printer) p3.clone();
            System.out.println("hashcode of Clonable Printer Class="+p3.hashCode());
            System.out.println("hashcode of Clonable Printer Class="+p4.hashCode());
            } }
```

Output:-

```
0 param constructor
hashcode of first Printer Class=366712642
hashcode of second Printer Class=366712642

hashcode of Clonable Printer Class=366712642
hashcode of Clonable Printer Class=366712642
```

===

“here we stopping to create object with new keyword then multithreading then cloning but here also we can create object with deserialization for this we have to consontreat also serialization ”