SINGLETON WITH CLONING

Breaking singleton class pattern using cloning concept

Cloning :-

- Creating new object having existing object state as the initial state of new object is called cloning.

How cloning is possible :-

- In java cloning is possible only on cloneable object… object become cloneable only when their classes implements java.lang.Cloneable(I) interface ..This is an empty interface and marker interface.

Marker interface :-

- The interface that makes underlying JVM/container/framework to provide special runtime capabilities is called marker interface.
  Ex:- CLONEABLE, SERIALIZABLE and etc..
- By seeing marker interface implementation the underlying JVM/container/Framework starts providing special capabilities to the object of class….. Not by using methods of market interface.

To do cloning:-

1. Make the object as cloneable object (the class of the object must implement java.lang.Cloenable(I))
2. Call clone() method of java.lang.Object class on existing cloneable object to get new object

Ex:-

UtilsClass:-

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

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

}
```

SINGLETON WITH CLONING

Printer Class:-

```
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);
    }
}
```

```
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=p.clone();
        p4=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=1829164700
hashcode of Clonable Printer Class=2018699554
```

SINGLETON WITH CLONING

“here indirectly **printer class** implementing cloneable interface.”

Q: **-** why constructor is not executing when object is creating through cloning?

Ans: **-** constructor is there to initialize the object with initial value ..When object is created through cloning already existing object data assign to new initial data.