Dt : 8/11/2023

****imp***

"java.lang.Object" class:

=>"java.lang.Object" class is the SuperClass/ParentClass of all the classes declared in the application.

=>The following are some important methods of "java.lang.Object" class:

1.hashCode()

2.toString()

3.equals()

4.wait()

5.notify()

6.notifyAll()

7.getClass()

8.finalize()

9.clone()

1.hashCode():

=>The unique numeric number which is created while object creation process is known as hashCode.

=>we use hashCode() method to display the hashCode of an object.

syntax:

int hc = obj.hashCode();

2.toString():

=>toString() method is used to display the data from the object.

=>This toString() method is executed automatically when we display Object reference variable.

synatx:

```
String var = obj.toString();
```

3.equals():

=>equals() method is used to compare two objects and generate boolean result.

syntax:

```
boolean b = obj1.equals(obj2);
```

4.wait()

5.notify()

6.notifyAll():

=>These methods are used to perform Inter-thread-Communication process.

7.getClass():

=>getClass() method is used to know the class-name of an Object.

syntax:

Class<?> c = obj.getClass();

8.finalize():

=>finalize() method is part of garbage collection process to check the anonymous objects eligible for detstroying or not.

****imp***

9.clone():

=>Th process of creating the duplicate copy of an Object is known as Cloning process.

=>we use clone() method to perform Cloning process.

syntax:

Object o = obj.clone();

=>we use the following steps to perform Cloning process:

step-1 : The user defined class must be implemented from 'java.lang.Cloneable' interface.

step-2 : The User defined class must be declared with User defined Object return_type method

step-3 : This User defined Object retrun_type method must call pre-defined 'clone()' method to perform Cloning

step-4 : we must execute user defined Object return_type method to

start cloning process.

--

=>This cloning process can be perfomed in two ways:

(a)Shallow Cloning process

(b)Deep Cloning process

(a)Shallow Cloning process:

=>In Shallow Cloning process,only OuterClass-objects are cloned and reffered class objects are not cloned.

Ex:

p1 : Address1.java

```
package p1;
public class Address1 extends Object
{
    public String city,state;
    public int pinCode;
    @Override
    public String toString()
    {
        return "City:"+city+"\nState:"+state+
                  "\nPinCode:"+pinCode;
    }
}
```

p1 : Employee1.java

```
package p1;
public class Employee1 extends Object implements
Cloneable
{
   public String id,name,desg;
   public Address1 ad = new
Address1();//reffered_Object
   @Override
   public String toString()
   {
        return "Id:"+id+"\nName:"+name+"\nDesg:"+desg;
   }
   public Object getClone1()throws
CloneNotSupportedException
   {
        Object o = super.clone();
        return o;
   }
}
```

p2 : DemoClone1.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoClone1

{

        public static void main(String[] args)

        {

     Scanner s = new Scanner(System.in);

     try(s;){

         try {
```

```
        //Original Object
Employee1 ob1 = new Employee1();//Outer Object
System.out.println("Enter the EmpId:");
ob1.id = s.nextLine();
System.out.println("Enter the EmpName:");
ob1.name = s.nextLine();
System.out.println("Enter the EmpDesg:");
ob1.desg = s.nextLine();
System.out.println("Enter the EmpCity:");
ob1.ad.city = s.nextLine();
System.out.println("Enter the EmpState:");
ob1.ad.state = s.nextLine();
System.out.println("Enter the EmpPinCode:");
ob1.ad.pinCode = s.nextInt();
System.out.println("***Original-ob1***");
System.out.println(ob1);
System.out.println(ob1.ad);
System.out.println("----hashCodes----");
System.out.println("hashCode of Employee : "+ob1.hashCode());
System.out.println("hashCode of Address : "+ob1.ad.hashCode());
  //Cloned Object
Employee1 ob2 = (Employee1)ob1.getClone1();
```

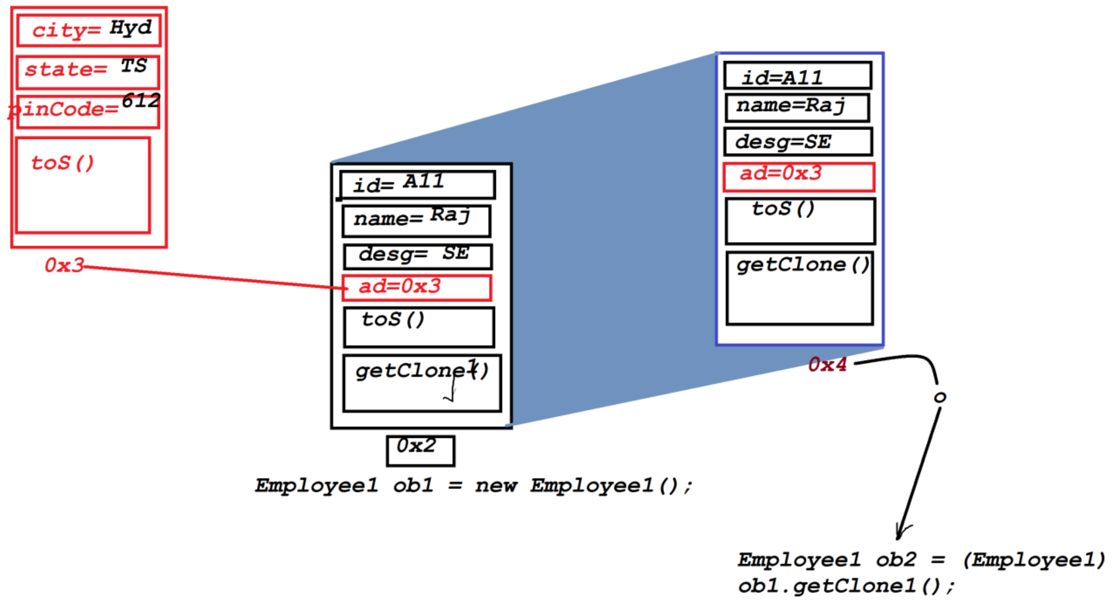
```
            System.out.println("***Cloned-ob2***");
            System.out.println(ob2);
            System.out.println(ob2.ad);
            System.out.println("----hashCodes----");
            System.out.println("hashCode of Employee : "+ob2.hashCode());
            System.out.println("hashCode of Address : "+ob2.ad.hashCode());
            }catch(Exception e) {e.printStackTrace();}
        }//end of try with resource
          }

}
```

o/p:

```
Enter the EmpId:
A11
Enter the EmpName:
Ram
Enter the EmpDesg:
TE
Enter the EmpCity:
Hyd
Enter the EmpState:
Ts
```

```
Enter the EmpPinCode:

6124

***Original-ob1***

Id:A11

Name:Ram

Desg:TE

City:Hyd

State:Ts

PinCode:6124

----hashCodes----

hashCode of Employee : 2074407503

hashCode of Address : 999966131

***Cloned-ob2***

Id:A11

Name:Ram

Desg:TE

City:Hyd

State:Ts

PinCode:6124

----hashCodes----

hashCode of Employee : 1989780873

hashCode of Address : 999966131
```

Diagram:



==
=

(b)Deep Cloning process:

=>In deep cloning process the both objects are cloned,which means

OuterClass objects and reffered class objects of duplicated.

=>To perform Deep Cloning process we have to perform cloning process

on reffered classes also

Ex:

p1 : Address2.java

```
package p1;
public class Address2 extends Object implements
Cloneable
{
   public String city,state;
   public int pinCode;
   @Override
   public String toString()
   {
        return "City:"+city+"\nState:"+state+
                "\nPinCode:"+pinCode;
   }
   public Object getClone2()throws
CloneNotSupportedException
   {
        Object o = super.clone();
        return o;
   }
}
```

p1 : Employee2.java

```
package p1;
public class Employee2 extends Object implements
Cloneable
{
   public String id,name,desg;
   public Address2 ad = new
Address2();//reffered_Object
   @Override
   public String toString()
   {
        return "Id:"+id+"\nName:"+name+"\nDesg:"+desg;
   }
   public Object getClone1()throws
CloneNotSupportedException
   {
        Employee2 o =
(Employee2)super.clone();//Creating Object for Employee
```

```
        o.ad = (Address2)o.ad.getClone2();//Creating
Object for Address
        return o;
    }
}
```

p2 : DemoClone2.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoClone2

{

        public static void main(String[] args)

        {

    Scanner s = new Scanner(System.in);

    try(s;){

        try {

                //Original Object

        Employee2 ob1 = new Employee2();//Outer Object

        System.out.println("Enter the EmpId:");

        ob1.id = s.nextLine();

        System.out.println("Enter the EmpName:");

        ob1.name = s.nextLine();

        System.out.println("Enter the EmpDesg:");
        ob1.desg = s.nextLine();

        System.out.println("Enter the EmpCity:");

        ob1.ad.city = s.nextLine();

        System.out.println("Enter the EmpState:");

        ob1.ad.state = s.nextLine();

        System.out.println("Enter the EmpPinCode:");

        ob1.ad.pinCode = s.nextInt();

        System.out.println("***Original-ob1***");

        System.out.println(ob1);

        System.out.println(ob1.ad);

        System.out.println("----hashCodes----");

        System.out.println("hashCode of Employee : "+ob1.hashCode());

        System.out.println("hashCode of Address : "+ob1.ad.hashCode());

         //Cloned Object

        Employee2 ob2 = (Employee2)ob1.getClone1();

        System.out.println("***Cloned-ob2***");

        System.out.println(ob2);

        System.out.println(ob2.ad);

        System.out.println("----hashCodes----");

        System.out.println("hashCode of Employee : "+ob2.hashCode());

        System.out.println("hashCode of Address : "+ob2.ad.hashCode());

        }catch(Exception e) {e.printStackTrace();}
```

```
    }//end of try with resource
      }


}
```

o/p:

```
Enter the EmpId:
R12
Enter the EmpName:
Raj
Enter the EmpDesg:
SE
Enter the EmpCity:
Hyd
Enter the EmpState:
TS
Enter the EmpPinCode:
6123
***Original-ob1***
Id:R12
Name:Raj
Desg:SE
City:Hyd
```

```
State:TS

PinCode:6123

----hashCodes----

hashCode of Employee : 2074407503

hashCode of Address : 999966131

***Cloned-ob2***

Id:R12

Name:Raj

Desg:SE

City:Hyd

State:TS

PinCode:6123

----hashCodes----

hashCode of Employee : 1989780873

hashCode of Address : 1480010240
```

==