2.1 OBJECT CLONING

Object cloning refers to creation of **exact copy of an object**. It creates a new instance of the class of current object and initializes all its fields with exactly the contents of the corresponding fields of this object. **In Java, there is no operator to create copy of an object.** Unlike C++, in Java, if we use assignment operator then it will create a copy of reference variable and not the object. This can be explained by taking an example. Following program demonstrates the same.

// Java program to demonstrate that assignment operator creates a new reference to same object.

```
import
java.io.*;class
sample
{
    int a;
    float b;
    sample()
    {
        a = 10;
        b = 20;
    }
}
class Mainclass
{
    public static void main(String[] args)
    {
        sample ob1 = new sample();
        System.out.println(ob1.a + " " +
        ob1.b);sample ob2 = ob1;
        ob2.a = 100;
        System.out.println(ob1.a+"
        "+ob1.b);System.out.println(ob2.a+"
        "+ob2.b);
    }
}
```

Output:

```
10 20.0
100 20.0
100 20.0
```

Creating a copy using clone() method

The class whose object's copy is to be made must have a public clone method in it or inone of its parent class.

- Every class that implements clone() should call **super.clone()** to obtain the cloned
  object reference.
- The class must also implement **java.lang.Cloneable** interface whose object clone we want to create otherwise it will throw **CloneNotSupportedException** when clone method is called on that class's object.

  Syntax:

  protected Object clone() throws CloneNotSupportedException

```
import  ava.util.ArrayList;

class sample1
{
    int a, b;
}
class sample2 implements Cloneable
{
    int
    c;int
    d;
    sample1 s = new sample1();
    public Object clone() throws CloneNotSupportedException
    {
            return super.clone();
    }
}
public class Mainclass
{
    public static void main(String args[]) throws CloneNotSupportedException
    {
    sample2 ob1 = new
    sample2();ob1.c = 10;
    ob1.d = 20;
    ob1.s.a = 30;
    ob1.s.b = 40;
    sample2 ob2 = (sample2)ob1.clone();
    ob2.d = 100; //Change in primitive type of ob2 will not be reflected in ob1 field
```

```
        ob2.s.a = 300; //Change in object type field will be reflected in both ob2
        and ob1(shallow copy)
        System.out.println(ob1.c + " " + ob1.d + " " +ob1.s.a + " " + ob1.s.b);
        System.out.println(ob2.c + " " + ob2.d + " " +ob2.s.a + " " + ob2.s.b);
        }
    }
```

Types of Object cloning

1. Deep Copy
2. Shallow Copy

Shallow copy

Shallow copy is method of copying an object. It is the default in cloning. In this method the **fields of an old object ob1 are copied to the new object ob2**. While copying the object type field the reference is copied to ob2 i.e. object ob2 will point to same location as pointed out by ob1. If the field value is a primitive type it copies the value of the primitive type. So, any changes made in referenced objects will be reflected in other object.

Note:

Shallow copies are cheap and simple to make.

Deep Copy

To create **a deep copy of object ob1 and place it in a new object ob2 then new copy of any referenced objects fields are created and these references are placed in object ob2.** This means any changes made in referenced object fields in object ob1 or ob2 will be reflected only in that object and not in the other. A deep copy copies all fields, and makes copies of dynamically allocated memory pointed to by the fields. A deep copy occurs when an object is copied along with the objects to which it refers.

```
//Java program for deep copy using
clone()

import java.util.ArrayList;
    class Test
    {
        int a, b;
    }
    class Test2 implements Cloneable
    {
        int c, d;
        Test ob1 = new Test();
        public Object clone() throws CloneNotSupportedException
        {
                // Assign the shallow copy to new refernce variable t
                Test2 t1 = (Test2)super.clone();
                t1.ob1 = new Test();
                // Create a new object for the field c
                // and assign it to shallow copy obtained,
                // to make it a deep copy
                return t1;
        }
}
public class Main
{
        public static void main(String args[]) throws CloneNotSupportedException
        {
        Test2 t2 = new Test2();
        t2.c = 10;
        t2.d = 20;
        t2.ob1.a = 30;
        t2.ob1.b = 40;
        Test2 t3 =
        (Test2)t2.clone();t3.c =
        100;
        t3.ob1.a = 300;
        System.out.println (t2.c + " " + t2.d + " " + t2.ob1.a + " " + t2.ob1.b);
        System.out.println (t3.c + " " + t3.d + " " + t3.ob1.a + " " + t3.ob1.b);
}       }
```

Output

```
10 20 30 40
100 20 300 0
```

Advantages of clone method:

- If we use assignment operator to assign an object reference to another reference variable then it will point to same address location of the old object and no new copyof the object will be created. Due to this any changes in reference variable will be reflected in original object.
- If we use copy constructor, then we have to copy all of the data over explicitly i.e. we have to reassign all the fields of the class in constructor explicitly. But in clone method this work of creating a new copy is done by the method itself. So to avoid extra processing we use object cloning.