**Final Variable**

final int x= 5;

x= 6//error

final int[] arr= new int[5];

arr= new int[5]//error

arr[1]= 2;

arr[1]= 3;

int[] arr= {1,2,3}

int[] arr1= arr; //No. of objects : 1, references: 2

arr= null;

arr1=? {1,2,3}

Sysout(arr1); //J@635bd7

**Cloning in Java:**

Class A implements Cloneable{

Int x;

Float y;

Int[] arr= new int[5];

A(int x, float y, int[] arr){

This.x= x;

This.y= y;

This.arr= arr;

}

Public A clone(){

Return (A) super.clone();

}}

Class B implements Cloneable{

//main (String args[]) throws CloneNotSupportedException{

A obj= new A( 22,3.4f, new int[]{2,3,4,6,7});

//A obj1= obj;//this is not creating another object but just adding a reference to it

//A obj1=(A) obj.clone();//error

B obj2= new B();

B obj3=obj2.clone();

Sysout(obj3==obj2);

}//Runtime error if not implements Cloneable.

//Protected: visible to all in the same package but also to child classes in a diff package using child ref

**Object class:**

protected Object clone() throws CloneNotSupportedException{

//returns shallow copy of the object;

}

//Method Overriding: U cannot throw a new CheckedException

Interface **Cloneable**: Marker interface: (Interface with no methods.): If someone calls a clone method on an object, the class should implement Cloneable interface. Else it will throw CloneNotSupportedException.

== : It checks whether the two references are pointing to the same object.

**Shallow Cloning:**

Though it creates a new copy for the object which is cloned, for the associated objects, it just adds a new reference.

Class A implements Cloneable{

Object o= new Object();

//When u override a method, the overridden method cannot throw a new CheckedException.

Protected A clone() throws CloneNotSupportedException {

Return (A) super.clone();

}

}

A obj= new A();

A obj1= Obj.clone();

Sysout(obj==obj1) //false Sysout(obj.o== obj1.o) //true [false]

**Deep Cloning: Creates a new object for the class and also for the associated objects.**

Class A implements Cloneable{

Object o= new Object();

Public A clone(){  
A obj1= new A();

Obj1.o= new Object();

Return obj1;

}

}

Class A implements Cloneable{

Int[] arr= {1,2,3};

Public A clone() throws CloneNotSupportedException{

Return (A) super.clone();

}

A obj1= new A();

A obj2 = obj1.clone();

Q) Modify this class for enabling deep cloning

Class A{

Object[] arr= {new Object(), new Object()};

}