How to implement clonable:

When one object is copied to another using assignment operator, only the reference of the object is copied. So a later change in one object is reflected in the other.

Java uses the clone() method of Object class to copy the contents of one object to the other. Classes can implement the Cloneable interface and override the clone() method of the Object class.

The following sample code will show the procedure for implementing cloneable interface.

A problem will occur if the Class that needs to be copied also contains reference to another object.

|  |
| --- |
| **public class** CloneExp **implements** Cloneable {  **private** String name; **private** String address; **private int** age; **private** Department depart; **public** CloneExp(){  } **public** CloneExp(String aName, **int** aAge, Department aDepart) {  **this**.name = aName; **this**.age = aAge; **this**.depart = aDepart; }  **protected** Object clone() **throws** CloneNotSupportedException {  CloneExp clone=(CloneExp)**super**.clone();  // make the copy of the object of type Department  // is it deep or shallow? Write a program to test it  clone.depart=(Department)depart.clone(); **return** clone;  } **public static void** main(String[] args) {  CloneExp ce=**new** CloneExp();  **try** { // make deep copy of the object of type CloneExp CloneExp cloned=(CloneExp)ce.clone();  // Now see if a change to the department object of ce changes the other  // one ie cloned } **catch** (CloneNotSupportedException e) { e.printStackTrace(); }  } } |

Write the class department and create CloneExp objects.

Clone the cloneExp object then change depart in the cloned one and see if the depart changes in the original one.

public Class Department {

private String name;

private int noofEmployees;

public Department ( string name ; int no)

{

this.name = name;

this.no=noofEmployees;

}

}