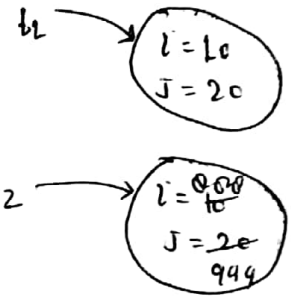
Clone

- The process of creating exactly duplicate object is called cloning.
- The main purpose of cloning is to maintain backup copy and to preserved state of an object.
- We can perform cloning by using clone method of object class.

protected native object clone() throws CloneNotSupportedException

Example:-

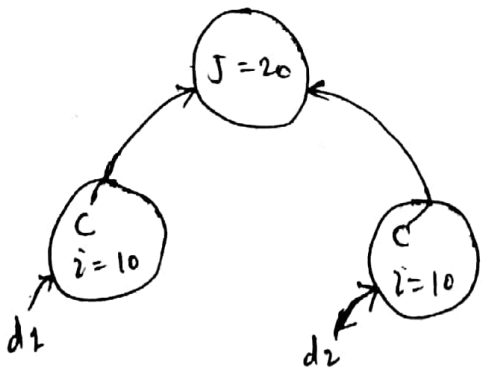
```
class Test implements Clonable
{                    ③ RE
 int i = 10;
 int j = 20;
 public static void main(String[] args) throws CloneNotSupportedException
 {                                              ② CE
    Test t1 = new Test();
    Test t2 = (Test) t1.clone();
     t2.i = 888;   ① CE
     t2.j = 999;
    Sopln(t1.i + "...." + t1.j);
 }                    10        20
}
```



We can perform cloning only for clonable objects. An object is said to be clonable if and only if corresponding class implements clonable interface. It is present in java.lang package and it does not contain any methods. It is marker interface. If we trying to perform cloning for non clonable objects then we will get RE exception saying CloneNotSupported exception.

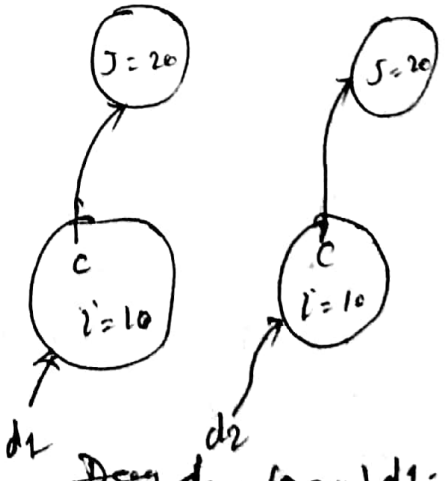
Shallow cloning And Deep Cloning

Shallow cloning



Dog d2 = (Dog)d1.clone();

Deep Cloning



Dog d2 = (Dog)d1.clone();

- The process of creating Bitwise copy of an Object is called shallow cloning
- If the main object contain primitive variables then exactly duplicate copy will be created in the cloned object.
- If the main object contain any reference variable then corresponding object will not be created just duplicate reference variable will be created pointing to old contain object.
- Object class clone method meant for shallow cloning.

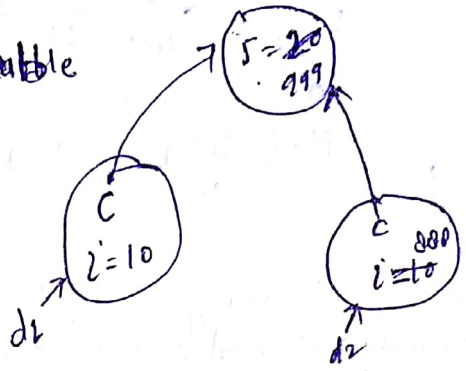
Example

```
class Cat
{
  int j;
  Cat(int j) {
    this.j = j;
  }
}

class Dog implements Cloneable
{
  Cat c;
  int i;
  Dog (Cat c; int i)
  {
    this.c = c;
    this.i = i;
  }

  Public Object clone() throws CloneNotSupperedException
  {
    return super.clone();
  }
}

class ShallowCloning
{
Public Static void main(String[] args) throws CloneNotSupperedException
{
    Cat c = new Cat(20);
    Dog d1 = new Dog(c, 10);
    System.out.println(d1.i + "....." + d1.c.j); // 10 ..... 20
    Dog d2 = (Dog) d1.clone();
    d2.i = 888;
    d2.c.j = 999;
    System.out.println(d1.i + "....." + d1.c.j); // 10 ..... 999
```



In shallow cloning by using cloned object reference if we perform any changes to the contain object then those changes will be reflected to the main object to overcome this problem we should go for deep cloning.

Deep Cloning :-

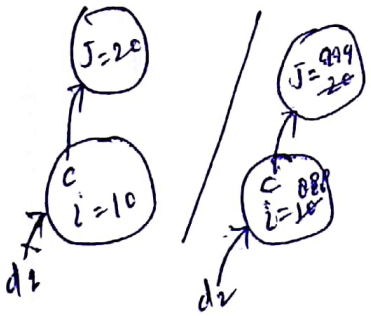
The process of creating exactly duplicate independent copy including contained object is called Deep Cloning.

- In Deep cloning if the main object contain any primitive variables then in the cloned object duplicate copy will be created
- if the main object contain any reference variable then the corresponding contain object also will be created in the cloned copy.
- By default object class clone method meant for shallow cloning but we can implement Deep Cloning explicitly by overriding clone method in our class.

Example:-

```
class Cat {
  int j;
  Cat(int j) {
    this.j = j;
  }
}
```

```
class Dog implements Cloneable {
  Cat c;
  int i;
  Dog(Cat c, int i) {
    this.c = c;
    this.i = i;
  }
  public Object clone() throws CloneNotSupportedException
  {
    Cat c1 = new Cat(c.j);
    Dog d = new Dog(c1, i);
    return d;
  }
}
```




```
class DeepCloning {
  public static void main(String[] args) throws CloneNotSupportedException
  {
    Cat c = new Cat(20);
    Dog d1 = new Dog(c, 10);
    System.out.println(d1.i + "....." d1.c.j); // 10 ..... 20
    Dog d2 = (Dog) d1.clone();
      d2.i = 888
      d2.c.j = 999
    System.out.println(d1.i + "....." + d1.c.j); // 10 ..... 20
  }
}
```

By using cloned object reference if we performed any change to the contained object then those changes will not be reflected to the main object.

* Which cloning is Best?

- If object contain only primitive variables then shallow cloning is the best choice.
- If object contain reference variables then Deep cloning is the best choice.