**Cloneable Interface in Java – Object Cloning**

In this post we are going to discuss about Object cloning with the help of examples. As the name suggests, object cloning is a process of generating the exact copy of object with the different name. Lets see how this can be done.

**A Simple example to understand Object cloning**

public class DogName implements Cloneable {

private String dname;

public DogName(String dname) {

this.dname = dname;

}

public String getName() {

return dname;

}

// Overriding clone() method of Object class

public Object clone()throws CloneNotSupportedException{

return (DogName)super.clone();

}

public static void main(String[] args) {

DogName obj1 = new DogName("Tommy");

try {

DogName obj2 = (DogName) obj1.clone();

System.out.println(obj2.getName());

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

}

}

Output:

Tommy

As you can see we have clonned the object obj1 into the object obj2. After cloning we got the same Dog name in object obj2, which we have set in the object obj1.

**Important Points to Note in the above example**:  
1) The class “DogName” implements Cloneable interface.  
2) Our class “DogName” overrides clone() method of Object class.  
3) During clone() method call we handle CloneNotSupportedException using try catch blocks.

**What if we don’t implement Cloneable interface?**

The program would throw CloneNotSupportedException if we don’t implement the Cloneable interface. A class implements the Cloneable interface to indicate to the Object.clone() method that it is legal for that method to make a field-for-field copy of instances of that class.

**Does clone object and original object point to the same location in memory**

The answer is no. The clone object has its own space in the memory where it copies the content of the original object. That’s why when we change the content of original object after cloning, the changes does not reflect in the clone object. Lets see this with the help of an example.

public class DogName implements Cloneable {

private String dname;

public String getDname() {

return dname;

}

public void setDname(String dname) {

this.dname = dname;

}

// Overriding clone() method of Object class

public Object clone()throws CloneNotSupportedException{

return (DogName)super.clone();

}

public static void main(String[] args) {

DogName obj1 = new DogName();

try {

obj1.setDname("Dog1");

//Cloning obj1 into obj2

DogName obj2 = (DogName) obj1.clone();

//Displaying both the objects content

System.out.println(obj1.getDname());

System.out.println(obj2.getDname());

//Setting up name in obj2

obj1.setDname("Dog2");

//Displaying both the objects content

System.out.println(obj1.getDname());

System.out.println(obj2.getDname());

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

}

}

**Output:**

Dog1

Dog1

Dog2

Dog1

As you can see that when we changed the content of object obj1 it does not get changed in the the clone object obj2.