# Implementing the Cloneable Interface

The java.lang.Object class contains a clone() method that returns a bitwise copy of the current object.

protected native Object clone() throws CloneNotSupportedException

Not all objects are cloneable. It particular only instances of classes that implement the Cloneable interface can be cloned. Trying to clone an object that does not implement the Cloneable interface throws a CloneNotSupportedException.

For example, to make the Car class cloneable, you simply declare that it implements the Cloneable interface. Since this is only a marker interface, you do not need to add any methods to the class.

public class Car extends MotorVehicle implements Cloneable {  
  
 // ...  
  
}

For example

Car c1 = new Car("New York A12 345", 150.0);  
Car c2 = (Car) c1.clone();

Most classes in the class library do not implement Cloneable so their instances are not cloneable.

Most of the time, clones are shallow copies. In other words if the object being cloned contains a reference to another object A, then the clone contains a reference to the same object A, not to a clone of A. If this isn't the behavior you want, you can override clone() yourself.

You may also override clone() if you want to make a subclass uncloneable, when one of its superclasses does implement Cloneable. In this case simply use a clone() method that throws a CloneNotSupportedException. For example,

public Object clone() throws CloneNotSupportedException {  
 throw new CloneNotSupportedException("Can't clone a SlowCar");  
 // never get here  
 return this;  
 }

You may also want to override clone() to make it public instead of protected. In this case, you can simply fall back on the superclass implementation. For example,

public Object clone() throws CloneNotSupportedException {  
 return super.clone();  
 }

[Previous](http://docs.google.com/45.html) | [Next](http://docs.google.com/47.html) | [Top](http://docs.google.com/index.html) | [Cafe au Lait](http://www.cafeaulait.org/)

Copyright 1997-1999, 2003 Elliotte Rusty Harold

[elharo@metalab.unc.edu](mailto:elharo@metalab.unc.edu)

Last Modified March 5, 2003