# LA INTERFAZ CLONEABLE

La interfaz **Cloneable** especifica que un objeto puede ser clonado.

* A menudo se desea crear una copia de un objeto. Para hacer esto necesitamos usar el método **clone** y entender la interfaz **Cloneable**.
* Una interfaz contiene constantes y métodos abstractos, pero la interfaz **Cloneable** es un caso especial. En el paquete **java.lang** de java la interfaz **Cloneable** está definida de la siguiente manera:

|  |
| --- |
| **package** java.lang;  **public interface** Cloneable {  } |

* Esta interfaz está vacía. Una interfaz vacía se conoce como ***interfaz de marcado***. Dichas interfaces no contienen constantes o métodos. Se usa para indicar que una clase posee ciertas propiedades deseables. Una clase que implementa la interfaz **Cloneable** es marcada como Cloneable y sus objetos pueden ser clonados usando el método **clone** definido en la clase **Objetc**.
* Varias clases de la librería java (ejemplos: **Date, Calendar** y **ArrayList**) implementan **Cloneable**. Por ende, las instancias de esas clases pueden ser clonadas. Por ejemplo, el siguiente código:

|  |
| --- |
| 1 import java.util.Calendar;  2 import java.util.GregorianCalendar;  3  4 public class **ClasesCloneables** {  5 public static void ***main***(String[] args) {  6  7 Calendar **calendario** = new GregorianCalendar(2015, 1, 12);  8 Calendar **calendarioA** = calendario;  9 Calendar **calendarioB** = (Calendar) calendario.clone();  10  11 boolean comparaA = calendario == calendarioA;  12 System.*out*.println(“calendario == calendarioA es: “ + comparaA);  13  14 boolean comparaB = calendario == calendarioB;  15 System.*out*.println(“calendario == calendarioB es: “ + comparaB);  16  17 boolean comparaC = calendario.equals(calendarioB);  18 System.*out*.println(“calendario.equals(calendarioB) es: “ + comparaC);  19 }  20 }  **Salida:**  calendario == calendarioA es: true  calendario == calendarioB es: false  calendario.equals(calendarioB) es: true |

* En el código anterior, en la línea **8** copia la referencia de **calendario** a **calendarioA**, entonces **calendario** y **calendarioA** apuntan al mismo objeto **Calendar**. La línea **9** crea un nuevo objeto que es el clon de **calendario** y asigna el nuevo objeto referencia a **calendarioB**. **calendarioB** y **calendario** son objetos diferentes con el mismo contenido. Cuando se clona, se crea otro objeto diferente con el mismo contenido.
* El siguiente código:

|  |
| --- |
| 1 import java.util.ArrayList;  2  3 public class **ListasCloneables** {  4 public static void ***main***(String[] args) {  5 ArrayList<Double> listaA = new ArrayList<>();  6  7 listaA.add(1.5);  8 listaA.add(2.5);  9 listaA.add(3.5);  10  11 ArrayList<Double> listaB = (ArrayList<Double>)listaA.clone();  12 ArrayList<Double> listaC = listaA;  13  14 listaB.add(4.5);  15 listaC.remove(1.5);  16  17 System.*out*.println("listaA : " + listaA);  18 System.*out*.println("listaB : " + listaB);  19 System.*out*.println("listaC : " + listaC);  20 }  21 }  Salida:  listaA : [2.5, 3.5]  listaB : [1.5, 2.5, 3.5, 4.5]  listaC : [2.5, 3.5] |