Interfaces

CC10A - 2003

INTERFACES

En Java no está soportada la herencia múltiple, esto es, no está permitido que una misma clase pueda heredar las propiedades de varias clases padres. En principio esto pudiera parecer una propiedad interesante que le daría una mayor potencia al lenguaje de programación, sin embargo los creadores de Java decidieron no implementar la herencia múltiple por considerar que esta añade al código una gran complejidad (lo que hace que muchas veces los programadores que emplean programas que sí la soportan no lleguen a usarla).
Sin embargo para no privar a Java de la potencia de la herencia múltiple sus creadores introdujeron un nuevo concepto: el de interface. Una interface es formalmente como una clase, con dos diferencias: sus métodos están vacíos, no hacen nada, y a la hora de definirla en vez de emplear la palabra clave "class" se emplea "inteface". Veámoslo con un ejemplo:

```
interface animal{
        public int edad = 10;
        public String nombre = "Bob";
        public void nace();
        public void get_nombre();
        void get_nombre(int i);
}
```

Cabe preguntarnos cual es el uso de una interface si sus métodos están vacíos. Bien, cuando una clase implementa una interface lo que estamos haciendo es una promesa de que esa clase va a implementar todos los métodos de la interface en cuestión. Si la clase que implementa la interface no "sobrescribiera" alguno de los métodos de la interface automáticamente esta clase se convertiría en abstracta y no podríamos crear ningún objeto de ella. Para que un método sobrescriba a otro ha de tener el mismo nombre, se le han de pasar los mismos datos, ha de devolver el mismo tipo de dato y ha de tener el mismo modificador que el método al que sobrescribe. Si no tuviera el mismo modificador el compilador nos daría un error y no nos dejaría seguir adelante. Veámoslo con un ejemplo de una clase que implementa la anterior interface:

```
interface animal{
        public int edad = 10;
        public String nombre = "Bob";
        public void nace();
        public void get_nombre();
        void get_nombre(int i);
}
public class perro3 implements animal{
        perro3(){
            get_nombre();
            get_nombre(8);
        }
//Compruévese como si cambiamos el nombre del método a nac()
//no compila ya que no henos sobreescrito todos los métodos
//de la interfaz.
        public void nace(){
            System.out.println("hola mundo");
        }
        public void get_nombre(){
```

```
            System.out.println(nombre );
        }
        public void get_nombre(int i){
            System.out.println(nombre +" " +i);
        }
        public static void main (String[] args){
                perro3 dog = new perro3();
//Compruevese como esta línea da un error al compilar debido
//a intentar asignar un valor a una variable final
// dog.edad = 8;
        }
}
```

Las variables que se definen en una interface llevan todas ellas el atributo final, y es obligatorio darles un valor dentro del cuerpo de la interface. Además no pueden llevar modificadores private ni protected, sólo public. Su función es la de ser una especie de constantes para todos los objetos que implementen dicha interface.
Por último decir que aunque una clase sólo puede heredar propiedades de otra clase puede implementar cuantas interfaces se desee, recuperándose así en buena parte la potencia de la herencia múltiple.

```
interface animal1{
        public int edad = 10;
        public String nombre = "Bob";
        public void nace();
}
interface animal2{
        public void get_nombre();
}
interface animal3{
        void get_nombre(int i);
}
public class perro4 implements animal1,animal2,animal3{
        perro4(){
           get_nombre();
           get_nombre(8);
           //edad = 10; no podemos cambiar este valor
        }
        public void nace(){
           System.out.println("hola mundo");
        }
        public void get_nombre(){
           System.out.println(nombre );
        }
        public void get_nombre(int i){
          System.out.println(nombre +" " +i);
        }
        public static void main (String[] args){
          perro4 dog = new perro4();
        }
}
```