Clase 8 - Interfaces

martes, 18 de abril de 2023 16:41

Interfaces → dice qué tiene q hacer clase q' la implementa pero no cómo.

↳ Define métodos abstractos y constantes

No puede haber nada de clase abstracta pero sí upcasting a una

Clase puede implementar + de una interfaz.

Permite métodos + genéricos

No hay herencia múltiple pero con interfaces se soluciona. Al no poder implementar interfaz, no pasa nada con combinarlas

Interfaz define tipo de dato

```
package taller;
public class PruebaInterfaces {

  public static void partida(Volador v){
    v.despegar();
  }
        El binding dinámico resuelve a que método invocar.
  public static void main(String[] args) {
    Volador[] m = new Volador[3];
    m[0]= new Avion();
    m[1]= new Helicóptero();
    m[2]= new Pajaro();
    for (int j=0; j<m.length; j++)
       partida(m[j]);
  }
```

Upcasting castea al tipo de la interface

- Las interfaces definen un nuevo **tipo de dato** entonces, podemos definir:

Volador[] m = new Volador[]

- El mecanismo de **upcasting** no tiene en cuenta si **Volador** es una clase concreta, abstracta o una interface. Funciona de la misma manera.
- **Polimorfismo**: el método **despegar()** es polimórfico, se comportará de acuerdo al tipo del objeto receptor, esto es, el **despegar()** de Avion es diferente al **despegar()** de Pajaro.

El principal objetivo de las interfaces es permitir el "upcasting" a otros tipos, además del upcasting al tipo base. Un mecanismo similar al que provee la herencia múltiple.

Interfaz vs clase abstracta.

Si quiero atributos o heredar métodos, interfaz

Siempre q se pueda usar interfaz, solo abstracta. Si necesito implementar o definir atributos