Apuntes/Interfaces

Interface

Colección de definiciones de métodos *abstractos*, métodos *default* y de declaraciones de variables de clases constantes, agrupadas bajo un nombre. Las interfaces proporcionan un mecanismo para que una clase defina comportamiento de un tipo de datos diferente al de sus superclases.

```
package nomPaquete;
public interface UnaInter extends SuperInter1, SuperInter2, .
    (declaración de métodos - public abstract)
    (declaración de métodos default - public)
    (declaración de constantes - public static final)
}
```

- Su principal objetivo es permitir el *upcasting* a otros tipos de datos, además del upcasting al tipo base.
- **public →** establece que la interface puede ser usada por cualquier clase o interface de cualquier paquete. Si se omite el especificador de acceso, la interface *solamente podría ser usada por las clases e interfaces contenidas en el mismo paquete que la interface declarada*.
- Una interface puede tener múltiples interfaces. Una interface de la cual otras heredan comportamiento son **SuperInterfaces** (misma relación que clase y superclase)
- Cualquier clase que implemente una interface deberá implementar cada uno de los métodos abstractos de la interface
- Para especificar que una clase implementa una interface se usa la palabra clave **implements**

Implementación de Interfaces

Cuando una clase implementa una interface se establece un contrato entre la interface y la clase que la implementa. El compilador chequea que la clase

implemente todos los métodos abstractos de la interface (y si es necesario algún método default para evitar ambiguedad)

- **Si una clase implementa dos interfaces que tienen el mismo método abstracto**, la clase está obligada a implementar cada uno de los métodos abstractos de sus interfaces (o lo implementa una vez y cumple con ambos contratos)
- **Si una clase implementa dos interfaces que tienen el mismo método declarado como default,** se produce una ambigüedad. La clase debe implementar (sobrescribir) el método con algún comportamiento para quitar esa ambiguedad.

Herencia múltiple

Java no soporta *herencia múltiple*, pero provee interfaces para lograr un comportamiento "similar" mediante la combinación de varias interfaces (no surjen complicaciones gracias a que las interfaces no tienen variables de instancia)

Diferencias con las clases abstractas

- Una clase solo puede extender otra clase sola. Pero **las interfaces pueden ser múltiples**.
- **No poseen variables de instancia**, tampoco suelen tener métodos propios que no sean abstractos.
- **Permite** trabajar con **la herencia múltiple**.

Algunas interfaces relevantes

- **java.util.Comparable →** Permite implementar un método de comparación entre instancias de una misma clase, con un criterio numérico ordinal (0<, 0, >0)

```
public int compareTo(T o); {
    //Implementación
}
```

- j**ava.util.Comparator** → Permite implementar un método de comparación entre instancias de una misma clase, con distintos criterios (uno debe implementar la lógica de comparación)

```
public int compare(T o1, T o2); {
    //Implementación (0 si o1 = o2 / 0< si o1 < o2 / >0 si
}
```