



UNIVERSIDAD TÉCNICA  
FEDERICO SANTA MARÍA

# Interfaces y Clases Internas

---

ELO-329: Diseño y programación  
orientados a objetos  
Agustín J. González



# ¿Qué queremos decir con interfaces y clases internas?

---

- NOTA: El término interfaz aquí NO se refiere a las interfaces gráficas.
- Aquí las **interfaces** son una manera de describir qué debería hacer una clase sin especificar el cómo.
- Entre caras -> Interfaz, plural interfaces
- Las **clases internas** son clases anidadas dentro de otras clases.
- Interfaces y clases internas son recursos esenciales en el manejo de *interfaces gráficas* en Java.



# Interfaces

---

- Una **interfaz** es la descripción de algún servicio que posteriormente alguna clase puede implementar (y por ende ofrecer).
- Por ejemplo, si un alumno sabe alemán, tenemos idea de lo que él es capaz. Además de ser persona (herencia) él cumple la interfaz “interprete de alemán”. También podríamos decir que él **es un** “interprete de alemán” (la misma relación que en herencia).



## Interfaces (cont.)

---

- En Java cada clase puede tener sólo una clase base (herencia no múltiple), por ello usamos Interfaces para abordar casos como el ejemplo previo.
- Se cumple también el principio de sustitución. Instancias de la clase que **implementa** una Interfaz pueden ser usadas donde se espera una instancia de la interfaz. Es similar a usar una instancias de una subclase cuando se espera un objeto de la clase base.



# Interfaces (cont.)

---

- No se permite crear objetos instancias de un Interfaz. Por la misma razón que no se puede crear instancias de clases abstractas. `New InterfazX();` ✗
- Todos los métodos de una Interfaz son públicos. No es necesario indicarlo.
- Pueden incluir constantes. En este caso son, por omisión, `public static` y `final`.
- Aspectos Sintácticos:

**Definición** de una interfaz(un archivo):

```
public interface Comparable
{
    int compareTo (Object other);
}
```

**Implementación** de una interfaz:

```
class Employee implements Comparable
{
    ....
    public int compareTo(Object other) {
        ....// implementación
    }
}
```



# Ejemplo: uso de interfaces

---

- Consideremos la extensión de la clase Employee para que podamos ordenar arreglos de empleados según su salario.
- Ver EmployeeSortTest.java
- Ver documentación de clase Arrays e interfaz Comparable, explicar clases genéricas.



# Clases Internas

---

- Son clases definidas dentro de otras (anidadas).
- Se usan para dar acceso a miembros de la clase anfitriona – incluso si son privados.
- Se usan como mecanismo de encapsulamiento.
- Son muy útiles para reducir código fuente. Especialmente cuando la clase sólo genera pocas instancias.
- Son comunes en el desarrollo de interfaces gráficas.



## Clases Internas (Cont.)

---

- Las clases internas existen sólo para el compilador, ya que éste las transforma en clases regulares separando la clase externa e interna con signo \$.
- La máquina virtual no distingue la clases internas.
- Ver `InnerClassTest.java`
- También se pueden definir al interior de un método.





# Clases internas: Ejemplo

---

```
public void start(double rate)
{
    private class InterestAdder implements ActionListener
    {
        public InterestAdder(double aRate)
        {    rate = aRate;    }
        public void actionPerformed(ActionEvent event)
        {
            double interest = balance * rate / 100; // update interest
            balance += interest;
            NumberFormat formatter = NumberFormat.getCurrencyInstance();
            System.out.println("balance=" + formatter.format(balance));
        }
        private double rate;
    }
    ActionListener adder = new InterestAdder(rate);
    Timer t = new Timer(1000, adder);
    t.start();
}
```



# Clases internas anónimas

---

Cuando necesitamos sólo una instancia de la clase, no necesitamos darle un nombre. Decimos que tal clase es interna y anónima. Ver `AnonymousInnerClassTest.java`

```
public void start(final double rate)
{
    ActionListener adder = new //sigue en otra línea..
        ActionListener() /* esta es una interfaz que implementamos directamente */
        {
            /* y da lugar a una clase interna anonima */
            public void actionPerformed(ActionEvent event)
            {
                double interest = balance * rate / 100;
                balance += interest;
                NumberFormat formatter = NumberFormat.getCurrencyInstance();
                System.out.println("balance=" + formatter.format(balance));
            }
        };

    Timer t = new Timer(1000, adder);
    t.start();
}
```