

Interfaces

- Para qué interfaces?
- Lo básico: métodos abstractos y atributos estáticos y finales
- Ejemplo de uso de interfaces para aprovechar el polimorfismo entre clases no relacionadas
- Herencia de interfaces
- Interfaz Comparable

Contratos a través de interfaces

- ❑ Las interfaces permiten que clases no necesariamente relacionadas, implementen un conjunto de métodos comunes.
- ❑ Para qué?
 - ❑ En DOO: Para poder usar distintas clases no relacionadas de manera polimórfica.
...Objetos de clases no relacionadas se pueden procesar de manera polimórfica
Ventaja: Se evita la herencia múltiple.
 - ❑ **A nivel arquitectura:**
 - ✓ Medio para definir un contrato entre un cliente y un servidor, permitiendo ocultar detalles de implementación.
 - ✓ Para implementar inyección de dependencia y facilitar testing.

Contratos

- ❑ Una interfaz define un contrato (conjunto de métodos) que debe ser cumplido (implementar los métodos) por la clase que implemente dicha interfaz.
- ❑ **A la interfaz la podemos usar como tipo. Quién tipe una variable con la interfaz (un parámetro, una variable de instancia, cualquier variable local) podrá sacar provecho del polimorfismo. Obviamente, sacrificando la especificidad!**
- ❑ **En esa variable se podrá alojar una referencia a cualquier instancia de una clase que implemente esa interfaz.** Este chequeo se hace a nivel compilación.

Interfaces en Java

- Las interfaces en las primeras versiones de Java permitían incorporar dos tipos de elementos
 - ☐ Variables que por defecto son public, static y final. Estos modificadores se agregan implícitamente. No es necesario agregarlos.
 - Por ser final debe llevar un valor de inicialización.
 - Por ser final no puede ser modificada.
 - ☐ Métodos abstractos sin implementación
- Con estos dos elementos alcanzaba para definir la especificación del contrato.

```
*UnaInterfaz.java  *SuperClase.java
1
2 public interface UnaInterfaz {
3
4     int constante=5;
5
6 }
7
```

```
*UnaInterfaz.java  *SuperClase.java
1
2 public class SuperClase {
3
4
5     public static void main(String[] args) {
6         UnaInterfaz.constante=5;
7     }
8
9 }
10
```

The final field UnaInterfaz.constante cannot be assigned
Press 'F2' for focus

Interfaces en Java

- Una clase que quiere implementar esta interfaz debe indicarlo en su declaración y obligatoriamente (si no es abstract) implementar todos los métodos especificados en la interfaz.

Interfaces en Java

UnaInterfaz.java SuperClase.java *Subclase.java

```
1
2 public interface UnaInterfaz {
3
4     int constante=5;
5
6     public void metodo();|
7
8 }
9
```

```
public class Subclase implements UnaInterfaz{
```

```
}
```

Piensen 3 formas de resolver este error.

- Empresa quiere implementar un facturador de todo lo que necesita pagar. Esto incluye Facturas de Luz, Factura de Gas, Facturas de agua, Servicio de Internet, todas las facturas presentadas en ese mes por sus proveedores. Los recibos de sueldos de sus empleados.
- La empresa debe implementar los siguientes dos métodos

>>imprimirComprobantes()

>>montoAPagar()

Interfaces en Java

```
public interface Costo {  
    public float montoAPagar();  
}
```


Interfaces en Java


```
/**  
 * @author Matias Butti  
 */  
public class FacturaDeLuz implements Costo {
```

```
    float monto;
```



```
    GregorianCalendar fecha;
```

```
    float porcentaje;
```

```
}
```

 The type FacturaDeLuz must implement the inherited abstract method Costo.montoAPagar()

2 quick fixes available:

-  [Add unimplemented methods](#)
-  [Make type 'FacturaDeLuz' abstract](#)

Press 'F2' for focus

Interfaces en Java

```
/**
 * @author Matias Butti
 *
 */
public class FacturaDeLuz implements Costo {

    float montoAPagar;
    GregorianCalendar fecha;
    float pocentajeDeDescuentoPorProntoPago;

    @Override
    public float montoAPagar() {

        return this.getMontoAPagar()
            - (this.getMontoAPagar()
                * this.getPocentajeDeDescuentoPorProntoPago() / 100);
    }
}
```

Herencia de Interfaces en Java

- Una clase puede extender otra clase e implementar una interfaz al mismo tiempo. Se suele utilizar para proveer una implementación base de la interfaz.
- Una interfaz puede heredar de otras interfaces.

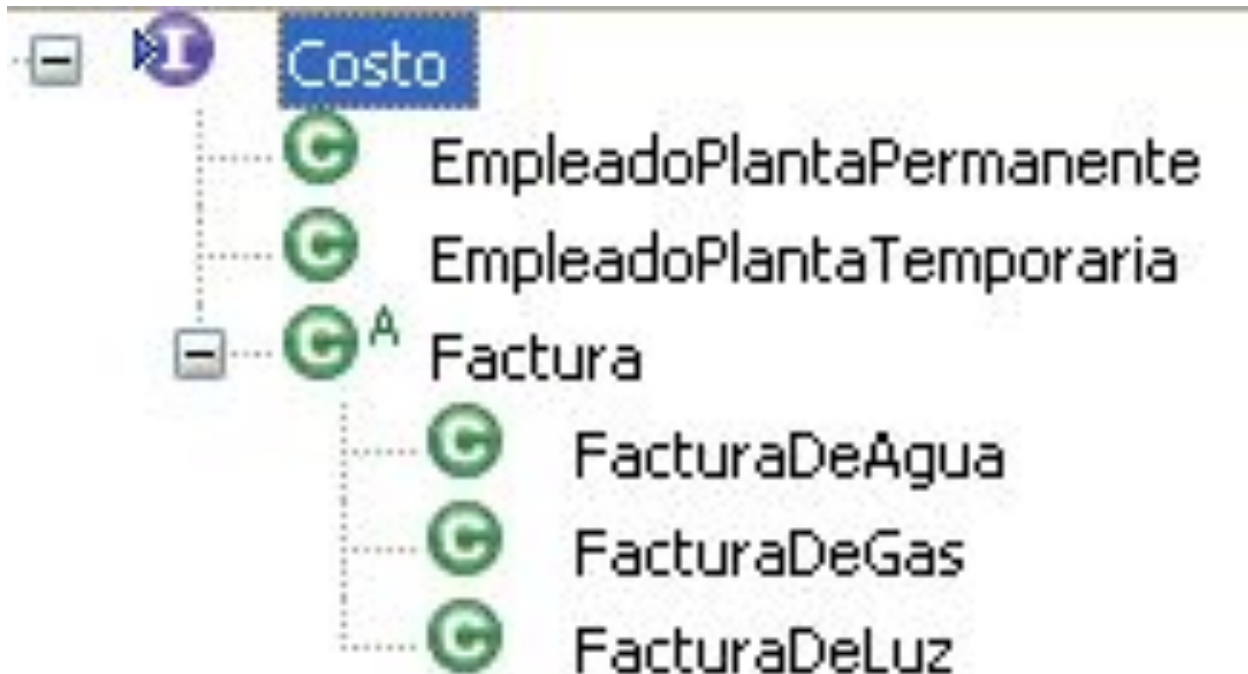
```
public interface Costo extends I1, I2 {  
  
    public float montoAPagar();  
  
}
```

- Si una clase es abstracta no necesita definir todos los métodos de la interfaz pero sí lo deberá hacer la primera clase hija no abstracta.

Interfaces

- En UML se representa con una caja de clase, con el estereotipo <<*interfaz*>>.
- En UML se utiliza la relación de realización (como jerarquía pero con líneas punteadas) para indicar que una clase implementa una interfaz.

Interfaces en Java



Interfaces en Java

```
/**
 * @author Matias Butti
 */
public class Facturador {

    public float facturar(ArrayList<Costo> costos) {
        float costoEnPesos=0;
        for (Costo costo : costos) {
            costoEnPesos += costo.montoAPagar();
        }
        return costoEnPesos;
    }
}
```

Ejemplo de interfaz de la API de Java

- LinkedList y ArrayList
- Comparable