

Tema 10: Interfaces

Antonio J. Sierra

Índice

1. Definición de una interfaz.
2. Implementación y uso de una interfaz.
3. Extensión de las interfaces.
4. Modelado UML de las interfaces.
5. Casos de uso. Diagrama UML de los casos de uso.

Introducción

- Abstracción completa de la implementación de una clase.
- La interfaz no define la implementación.
- Métodos sin cuerpo.
- Una clase puede implementar cualquier número de interfaces.
 - Una clase sólo puede heredar de una superclase (abstracta o no)
- Admiten resolución de método dinámica.

Definición de Interfaz (I)

```
acceso interface nombre{  
    tipo var_final1 = valor;  
    tipo var_final2 = valor;  
    // ...  
    tipo var_finalN = valor;  
  
    tipo_devuelto método1 (lista_de_parámetros);  
    tipo_devuelto método2 (lista_de_parámetros);  
    // ...  
    tipo_devuelto métodoN(lista_de_parámetros);  
}
```

Definición de Interfaz (II)

acceso: o `public` o no se utiliza (por defecto).

```
tipo var_final1 = valor;
```

Implícitamente **final** y **static**.

Implementación parcial

```
abstract class Imcompleto implements LaInterfaz{
    int a, b;
    void muestra() {
        System.out.println(a + " " + b);
    }
    // ...
}
```

Acceso a implementaciones a través de la interfaz

- Se pueden declarar variables referencias a la interfaz
- Cualquier instancia que implementa la interfaz puede almacenarse en una variable de ese tipo.
- El método al que llamar se determina dinámicamente durante la ejecución.
- Proceso similar al uso de una referencia a la superclase para acceder a un objeto de la subclase.

Herencia con interfaces

- Una interfaz puede heredar otra utilizando la palabra clave **extends**.
- La sintaxis es la misma que se utiliza en la herencia de clases.
- Cuando una clase implementa una interfaz que hereda de otra, tiene que implementar todos los métodos definidos en la cadena de herencia de la interfaz.

Interfaz en UML.Definición

- Una interfaz es una colección de operaciones que especifican un servicio de una clase o componente.
- Por lo tanto, una interfaz describe el comportamiento visible externamente de ese elemento.
- Una interfaz puede representar el comportamiento completo de una clase o componente o sólo una parte de este comportamiento.
- Una interfaz define un conjunto de especificaciones de operaciones (o sea, sus signaturas), pero nunca sus implementaciones.
- Una interfaz raramente se encuentra asilada, más bien, suele estar conectada a la clase o componente que la realiza.

Modelado UML de las interfaces

- Es un elemento estructural.




Caso de uso

- Un Caso de uso, es una descripción de un conjunto de secuencias de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular.
- Un caso de uso se utiliza para estructurar los aspectos de comportamiento en un modelo. Un caso de uso es realizado por una colaboración.

Modelado de los Casos de Uso

- Es un elemento estructural



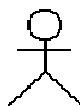
Realizar Pedido

Diagrama de casos de uso

- Un diagrama de casos de uso es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones.
- Todo sistema tiene como mínimo un diagrama *Main Use Case*, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).
- El diagrama muestra los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

Actor

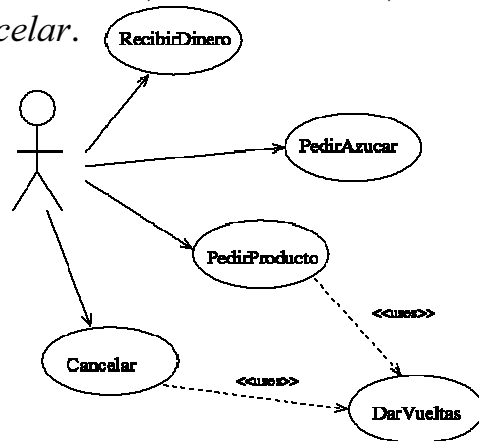
- Un **actor** es una entidad que utiliza alguno de los casos de uso del sistema.



- En el ejemplo observamos un único actor representando al usuario de la máquina de café.

Ejemplo. Máquina de Café

- Se tienen como casos de uso de la máquina de café *RecibirDinero*, *PedirAzucar*, *PedirProducto*, *DarVueltas* y *Cancelar*.



Tres tipos de Relaciones entre Casos de Uso

- "**comunica**" (<<communicates>>): Relación (asociación) entre un actor y un caso de uso que denota la participación del actor en dicho caso de uso. En el diagrama del ejemplo de la figura anterior todas las líneas que salen del actor denotan este tipo de asociación (en realidad estereotipada como <<communicates>>).
- "**usa**" (<<uses>>) (o <<include>> en la nueva versión de UML): Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro. En el caso del ejemplo el caso de uso *Cancelar* incluye en su comportamiento al de *DarVueltas* y *PedirProducto* incluye también *DarVueltas*.
- "**extiende**" (<<extends>>): Relación de dependencia entre dos casos de uso que denota que un caso de uso es una especialización de otro. Por ejemplo, podría tenerse un caso de uso que extienda la forma de pedir azúcar, para que permita escoger el tipo de azúcar (normal, dietético o moreno) y además la cantidad en las unidades adecuadas (cucharadas o bolsas).



Cuando usarlas

- Se utiliza una relación de tipo <<extends>> entre casos de uso cuando nos encontramos con un caso de uso similar a otro pero que hace algo más que éste (variante).
 - Por contra, utilizaremos una relación tipo <<uses>> cuando nos encontramos con una parte de comportamiento similar en dos casos de uso y no queremos repetir la descripción de dicho comportamiento común.
- En una relación <<extends>>, un actor que lleve a cabo el caso de uso base puede realizar o no sus extensiones.
 - Mientras, en una relación <<include>> el actor que realiza el caso de uso base también realiza el caso de uso incluido.
 - En general utilizaremos <<extends>> cuando se presenta una variación del comportamiento normal, y <<include>> cuando se repite un comportamiento en dos casos de uso y queremos evitar dicha repetición.