Interfaces

Interfaces en Java

Existen varias situaciones en la ingeniería de software en las que es importante que distintos grupos de programadores acepten un "contrato" que especifique cómo interactúa su software. Cada grupo debería poder escribir su código sin tener conocimiento alguno de cómo está escrito el código del otro grupo. En términos generales, *las interfaces* son contratos de este tipo.

Por ejemplo, imaginemos una sociedad futurista en la que automóviles robóticos controlados por ordenador transportan pasajeros por las calles de la ciudad sin necesidad de un operador humano. Los fabricantes de automóviles escriben software (Java, por supuesto) que hace funcionar el automóvil: para, arranca, acelera, gira a la izquierda, etcétera. Otro grupo industrial, los fabricantes de instrumentos de guía electrónicos, fabrican sistemas informáticos que reciben datos de posición del GPS (sistema de posicionamiento global) y la transmisión inalámbrica de las condiciones del tráfico y utilizan esa información para conducir el automóvil.

Los fabricantes de automóviles deben publicar una interfaz estándar de la industria que explique en detalle qué métodos se pueden invocar para hacer que el automóvil se mueva (cualquier automóvil, de cualquier fabricante). Los fabricantes de guías pueden entonces escribir un software que invoque los métodos descritos en la interfaz para controlar el automóvil. Ningún grupo industrial necesita saber cómo se implementa el software del otro grupo. De hecho, cada grupo considera que su software es altamente propietario y se reserva el derecho de modificarlo en cualquier momento, siempre y cuando continúe adhiriendo a la interfaz publicada.

En el lenguaje de programación Java, una *interfaz* es un tipo de referencia, similar a una clase, que *solo* puede contener constantes, firmas de métodos, métodos predeterminados, métodos estáticos (privados o públicos, no protegidos), métodos no abstractos de instancia (privados, no públicos, no protegidos) y tipos anidados. Los cuerpos de los métodos solo existen para los métodos predeterminados, los métodos privados y los métodos estáticos. Las interfaces no se pueden instanciar; solo pueden implementarse mediante clases o extenderse mediante otras interfaces. La extensión se analiza más adelante en esta sección.

Definir una interfaz es similar a crear una nueva clase:

Tenga en cuenta que las firmas de los métodos no tienen llaves y terminan con punto y coma.

Para utilizar una interfaz, se escribe una clase que implementa la interfaz. Cuando una clase instanciable implementa una interfaz, proporciona un cuerpo de método para cada uno de los métodos declarados en la interfaz. Por ejemplo,

```
public interface OperateCar {

   // constant declarations, if any

   // method signatures

   // An enum with values RIGHT, LEFT
   int turn(Direction direction,
            double radius,
            double startSpeed,
            double endSpeed);
   int changeLanes(Direction direction,
                   double startSpeed,
                   double endSpeed);
   int signalTurn(Direction direction,
                  boolean signalOn);
   int getRadarFront(double distanceToCar,
                     double speedOfCar);
   int getRadarRear(double distanceToCar,
                    double speedOfCar);
         ......
   // more method signatures
}
```

Tenga en cuenta que las firmas de los métodos no tienen llaves y terminan con punto y coma.

Para utilizar una interfaz, se escribe una clase que implementa la interfaz. Cuando una clase instanciable implementa una interfaz, proporciona un cuerpo de método para cada uno de los métodos declarados en la interfaz. Por ejemplo,

```
public class OperateBMW760i implements OperateCar {

    // the OperateCar method signatures, with implementation --
    // for example:
    public int signalTurn(Direction direction, boolean signalOn) {
       // code to turn BMW's LEFT turn indicator lights on
       // code to turn BMW's LEFT turn indicator lights off
       // code to turn BMW's RIGHT turn indicator lights on
       // code to turn BMW's RIGHT turn indicator lights off
    }

    // other members, as needed -- for example, helper classes not
    // visible to clients of the interface
}
```

En el ejemplo del coche robótico que se muestra más arriba, serán los fabricantes de automóviles los que implementarán la interfaz. La implementación de Chevrolet será sustancialmente diferente a la de Toyota, por supuesto, pero ambos fabricantes se ceñirán a la misma interfaz. Los fabricantes de sistemas de guía, que son los clientes de la interfaz, construirán sistemas que utilicen datos de GPS sobre la ubicación del coche, mapas digitales de calles y datos de tráfico para conducir el coche. Al hacerlo, los sistemas de guía invocarán los métodos de la interfaz: girar, cambiar de carril, frenar, acelerar, etc.

Interfaces como API

El ejemplo del coche robótico muestra una interfaz que se utiliza como *interfaz de programación de aplicaciones* (API) estándar de la industria. Las API también son comunes en los productos de software comerciales. Normalmente, una empresa vende un paquete de software que contiene métodos complejos que otra empresa desea utilizar en su propio producto de software. Un ejemplo sería un paquete de métodos de procesamiento de imágenes digitales que se vende a empresas que elaboran programas de gráficos para el usuario final:

- La empresa de procesamiento de imágenes escribe sus clases para implementar una interfaz, que hace pública para sus clientes.
- Luego, la empresa de gráficos invoca los métodos de procesamiento de imágenes utilizando las firmas y los tipos de retorno definidos en la interfaz.

Si bien la API de la empresa de procesamiento de imágenes se hace pública (para sus clientes), su implementación de la API se mantiene como un secreto muy bien guardado; de hecho, puede revisar la implementación en una fecha posterior siempre que continúe implementando la interfaz original en la que sus clientes han confiado. Las interfaces son tipos de referencia versátiles que le permiten definir métodos ***default*** y agregar funcionalidad a un tipo determinado sin romper las clases que las implementan. Además, a veces puede factorizar fragmentos comunes de código en los métodos *private*, lo que reduce la duplicación de código en los métodos ***default***.

Definición de una interfaz

Una declaración de interfaz consta de modificadores, la palabra clave ***interface*** el nombre de la interfaz, una lista separada por comas de las interfaces principales (si las hay) y el cuerpo de la interfaz. Por ejemplo:

```
public interface GroupedInterface extends Interface1, Interface2, Interface3 {

    // constant declarations

    // base of natural logarithms
    double E = 2.718282;

    // method signatures
    void doSomething (int i, double x);
    int doSomethingElse(String s);
}
```

El especificador de acceso ***public*** indica que la interfaz puede ser utilizada por cualquier clase en cualquier paquete. Si no especifica que la interfaz es pública, entonces su interfaz será accesible únicamente para las clases definidas en el mismo paquete que la interfaz.

Una interfaz puede extender otras interfaces, al igual que una subclase de una clase o extender otra clase. Sin embargo, mientras que una clase puede extender solo una clase más, una interfaz puede extender cualquier número de interfaces. La declaración de interfaz incluye una lista separada por comas de todas las interfaces que extiende.

El cuerpo de la interfaz puede contener métodos abstractos, métodos predeterminados y métodos estáticos.

Un método abstracto dentro de una interfaz va seguido de un punto y coma, pero sin llaves (un método abstracto no contiene una implementación).

Los métodos predeterminados se definen con el modificador ***default*** y los métodos estáticos con la palabra clave ***static***. Todos los métodos abstractos, predeterminados y estáticos de una interfaz son implícitamente públicos, por lo que puede omitir el modificador ***public***.

Además, una interfaz puede contener declaraciones de constantes. Todos los valores constantes definidos en una interfaz son implícitamente public, static, y final. Una vez más, puede omitir estos modificadores.

Bibliografía: https://dev.java/learn/interfaces/defining-interfaces/