APRENDERAPROGRAMAR.COM

CONCEPTO O DEFINICIÓN DE INTERFACE EN JAVA. ¿QUÉ ES UNA INTERFACE? TIPOS DE INTERFAZ. EJEMPLO PRÁCTICO. (CU00677B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº77 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

CONCEPTO DE INTERFAZ O INTERFACE JAVA. AMPLIACIÓN.

Ya hemos dicho que interface en Java es una palabra que puede tener diferentes significados. Vamos a repasar algunos significados que ya hemos visto y a introducir otros nuevos. Es habitual que entre las personas que estudian Java haya dificultades para entender el concepto de interface. Esto es hasta cierto punto normal porque dicho término tiene distintos usos.

Ya hemos dicho que interface en Java es una palabra que puede tener diferentes significados. Vamos a repasar algunos significados que ya hemos visto y a introducir otros nuevos.

a) **Interface:** parte visible y pública de una clase que describe qué hace y cómo usarla. La documentación de una clase en el API de Java vendría siendo su interface.

b) **Interface:** parte visible y pública de un método que describe qué hace y cómo usarlo (signatura del método + instrucciones de uso). La documentación de un método en el API de Java vendría siendo su interface.

c) **Interfaz Gráfica de Usuario**, interfaz de usuario o GUI (Graphical User Interface): es el entorno de objetos gráficos disponibles para un usuario en el marco de una aplicación o sistema operativo. El sistema operativo MS-Dos se basaba en intérpretes de comando (escritura de instrucciones por consola) pero Windows se basa en una interfaz gráfica de usuario (su entorno de escritorio), Linux en otra y Macintosh en otra.

d) **Herramientas para crear Interfaces gráficas de usuario** en Java. Hacemos referencia principalmente a los paquetes (packages) del API de Java *swing* y *awt* (Abstract Windowing Toolkit). Las clases de estos paquetes permiten crear interfaces gráficas de usuario basadas en ventanas estilo “Windows” para nuestras aplicaciones.

e) **Interfaces de Java:** son unas entidades abstractas conceptualmente por encima de las clases cuyo concepto vamos a introducir a continuación.

Para explicar el concepto de interface nos valdremos de un ejemplo. Supongamos que al crear un programa creamos una ciudad. Para crear la ciudad no partimos de cero: disponemos de edificios prefabricados (las clases del API de Java). Pero también disponemos de algo más: de normas de

urbanismo ya definidas (las interfaces del API de Java). Cada norma vamos a decir que es una interface: nos dice qué debemos cumplir para que al construir un edificio (clase) se pueda calificar con un nombre determinado. Supongamos una norma denominada “Edificio a dos aguas”, cuyo contenido incluye:

a) El edificio habrá de tener cuatro paredes.

b) El edificio habrá de tener un tejado formado por dos planos.

c) Otras especificaciones.

Si al construir un edificio se cumplen las condiciones de las normas de urbanismo, el ayuntamiento nos permite que en la publicidad y escrituras del edificio conste que se trata de un edificio a dos aguas. Si no cumplimos las especificaciones, no podemos usar esa denominación. Por ejemplo, estaría prohibido que un edificio con forma de pentágono y cinco paredes se denominara “Edificio a dos aguas”. Por el contrario, sería posible denominar edificio a dos aguas a una capilla que cumpla la norma, también a una vivienda unifamiliar que la cumpla, o a una biblioteca que la cumpla. Trasladémoslo a Java. Consideremos la interface “List” que equivale a nuestra norma, cuyo contenido incluye:

a) La clase habrá de tener un método cuya signatura sea *get(int index)* que devuelva el objeto de la lista en la posición especificada por el entero *index*.

b) La clase habrá de tener un método cuya signatura sea *isEmpty()* que devuelva un valor booleano true si la lista no contiene objetos o false en caso contrario.

c) Otras especificaciones.

Si una clase cumple y declara que cumple las especificaciones de la interfaz, decimos que esa clase implementa la interfaz. Si no cumple las especificaciones, no la implementa. Para implementar la interfaz la clase ha de incluir todos los métodos que defina la interfaz. Por ejemplo, una clase que no incluya el método *get(int index)* no cumplirá la especificación y por tanto no podrá decirse que implemente la interfaz List. Veamos ahora cómo empieza la documentación de esta interface del API de Java:

java.util

Interface List<E>

All Superinterfaces:

Collection<E>, Iterable<E>

All Known Implementing Classes:

AbstractList, AbstractSequentialList, ArrayList, AttributeList, LinkedList, RoleList, Stack, Vector y otras

Una interface puede tener "normas" superiores, a las que denominamos **"Superinterfaces"**, y "normas" inferiores, a las que denominamos **"Subinterfaces"**, de acuerdo con una jerarquía. A su vez, una interface puede tener clases que la implementan. Por ejemplo en el caso de List, esta interface está implementada por las clases ArrayList, LinkedList, Stack y otras. Todas las clases que implementan una interface podemos decir que tienen algo en común. De esta manera, todas las clases del API de Java están organizadas.

EJERCICIO

Busca información en la documentación oficial de Java sobre la interface Iterable. ¿Qué clases de las que conoces implementan esta interface? ¿Qué método han de tener disponible las clases que implementan esta interface? ¿Para qué sirve este método?

Puedes comprobar si tus respuestas son correctas consultando en los foros aprenderaprogramar.com.

Próxima entrega: CU00678B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188