INTERFACES

Una interfaz es un dispositivo o un sistema utilizado por entidades inconexas para interactuar. Analogamente una interfaz Java es un dispositivo que permite interactuar con objetos no relacionados entre si. Las interfaces Java en realidad definen un conjunto de mensajes que se puede aplicar a muchas clases de objetos, a los que cada una de ellas debe responder de forma adecuada, por eso una interfaz recibe también el nombre de protocolo.

DEFINIR UNA INTERFAZ

Una interfaz consta de dos partes, el nombre de la interfaz precedido por la palabra reservada interface y el cuerpo de la interfaz encerrado entre llaves.

El modificador de acceso public indica que la interfaz puede ser utilizada por cualquier clase de cualquier paquete. Si no se especifica, entonces solo estará accesible para las clases definidas en el mismo paquete que la interfaz. Una interfaz puede incluirse en un paquete exactamente igual que una clase.

El cuerpo de la interfaz puede incluir declaraciones de constantes y declaraciones de métodos.

La palabra clave extends significa que se esta definiendo una interfaz que es una extensión de otras, también se puede decir que es una interfaz derivada de otras, estas otras e especifican a continuacion de extends sepradas por comas. A diferencia de las clases, una interfaz puede derivarse de mas de una superinterfaz. Una interfaz asi definida hereda todas las constantes y metodos de sus superinterfaces, excepto las constantes y metodos que queden ocultos porque se redefinan.

El nombre de una interfaz se puede utilizar en cualquier lugar donde se pueda utilizar el nombre de una clase.

Las interfaces al igual que las clases y metodos bstractos proporcionan plantillas de comportamiento qe se espera sean implementadas por otras clases. Esto es, una interfaz Java declara un conjunto de metodos, pero no los define. Tambien puede incluir definiciones de constantes.

Una interfaz solo declara los metodos, no los define, y ademas puede definir constantes. Tambien cada una de las declaraciones y definiciones finaliza con “;”. Todos los metodos declarados en una interfaz son implicitamente publicos y abstractos y todas las constatnes son implicitamente publicas, finales y estaticas.

Cualquier clase puede tener acceso a las constantes de la interfaz a traves del nombre de la misma, en cambio una clase que implemente la interfaz puede tratar las constantes como su las hubiese heredado, esto es, accediendo directamente a su nombre.

UTILIZAR UNA INTERFAZ

Para utilizar una interfaz hay que añadir el nombre de la misma precedido por la palabra clave implements a la definicion de la clase. La palabra clave implements sigue a la palabra clave extends si existe.

CLASE ABSTRACTA FRENTE A INTERFAZ

UTILIZAR UNA INTERFAZ COMO UN TIPO

Una interfaz es un nuevo tipo de dato, un tipo referenciado. Por lo tanto, el nombre de una interfaz se puede utilizar en cualquier lugar donde pueda aparecer el nombre de cualquier otro tipo de datos.

Una variable del tipo de una interfaz espera referenciar un objeto que tenga implementada dicha interfaz, de lo contrario el compilador Java mostrara un error.

Asimismo se puede observar que es posible convertir implícitamente referencias a objetos que implementan una interfaz en referencias a esa interfaz y viceversa, pero en este caso explícitamente. Logicamente con lo que sabemos podemos deducir que con una referencia a la interfaz solo se tiene acceso a los métodos y constantes declarados en dicha interfaz.

INTERFACES FRENTE A HARENCIA MULTIPLE

La herencia multiple y las interfaces son bastante diferentes a pesar de que las interfaces pueden resolver problemas similares:

* Desde una interfaz, una clase solo hereda constantes
* Desde una interfaz, una clase no puede heredar definiciones de métodos
* La jerarquía de interfaces es independiente de la jerarquía de clases. De hecho, varias clases pueden implementar la misma interfaz y no pertenecer a la misma jerarquía de clases. En cambio, cuando se habla de herencia multiple, todas las clases pertenecen a la misma jerarquía.

PARA QUE SIRVE UNA INTERFAZ

Una interfaz se utiliza para definir un protocolo de conducta que puede ser implementado por cualquier clase en una jerarquía de clases. La utilidad que esto pueda tener puede resumirse en los puntos siguientes:

* Captar similitudes entre clases no relacionadas sin forzar entre ellas una relación artificial. Una acción de este tipo permitiría incluso definir una matriz de objetos de esas clases y aplicar si fuera necesario la definición de polimorfismo.
* Declarar métodos que una o mas clases deben implementar en determinadas situaciones
* Publicar la interfaz de programación de una clase sin descubrir como esta implementada.

IMPLEMENTAR MULTIPLES INTERFACES

Una clase puede implementar una o mas interfaces. Cuando una clase implemente multiples interfaces puede suceder que dos o mas interfaces diferentes implementen el mismo método. Si esto ocurre, proceda de alguna de las formas indicadas a continuación:

* Si los métodos tienen el mismo prototipo, basta con definir uno en la clase
* Si los métodos difieren en el numero o tipo de sus parámetros, estamos en el caso de uina sobrecarga del método, implemeta todas las sobrecargas.
* Si los métodos solo difieren en el tipo del valor retornado, no existe sobrecarga y el compilador produce un error, ya que dos métodos pertenecientes a la misma clase no pueden diferir solo en el tipo del resultado.

CLASES ANIDADAS

Una clase anidada es una clase que es un miembro de otra clase.

Una clase se debe definir dentro de otra solo cuidando tenga sentido en el contexto de la clase que la incluye o cuando depende de la función que desempeña la clase que la incluye. Por ejemplo una ventana puede definir su propio cursor, en este caso, la ventana puede ser un objeto de una clase y el cursor de una clase animada.

Una clase animada es un miembro mas de la clase que la contiene.

Recuerda: un miembro estatico es un miembro de la clase y uno no estatico es un miembro del objeto y como acurria con los métodos estaticosm una clase anidada estatica no puede referirse directamente a un miembro del objeto, so lo puede hacerlo a través de un objeto de su clase.

CLASES INTERNAS

Cuando un miembro de una clase es una clase anidada y no es estatic recibe el nombre de clase interna por tratarse de un miembro del objeto. Esto significa que cuando se cree un objeto de la clase externa también se creara uno de la interna.

CLASES DEFINIDAS DENTRO DE UN METODO

Java permite definir una clase dentro de un método. Una clase definida dentro de un método tiene unas reglas de acceso bastante restrictivas. Un método de una clase definida dentro de otro método solo tiene acceso a sus variables locales o parámetros formales declarados final.

CLASES ANONIMAS

Algunas veces podemos escribir una clase dentro de un método sin necesidad de identificarla. Una clase definida de esta forma recibe el nombre de clase anónima.

Por otra parte, para crear con new un objeto de una clase necesitamos conocer el nombre de la misma.

Se puede observar que la llamada a new va seguida de la definición de la clase y que no se utiliza el nombre de la clase, sino el nombre de la interfaz que la clase implementa, notar que en este caso no interviene la palabra clave implements y que la sentencia return finaliza con punto y coma.

En resumen una clase anónima es una forma de evitar el que tengamos que pensar en un nombre trivial para una clase pequeña en código, utilizada en un lugar concreto. Evidentemente, el precio que se apaga es que no podemos crear objetos de esta clase fuera del lugar donde haya sido definida.

**EXCEPCIONES**

El lenguaje Java incorpora soporte para manejar situaciones anómalas. Conocidas como “excepciones”, que pueden ocurrir durante la ejecución de un programa. Con el sistema de manipulación de excepciones de Java, un programa puede comunicar eventos inesperados a un contexto de ejecución mas capacitado para responder a tales eventos anormales. Estas excepciones son manejadas por código fuera del flujo normal de control del programa.

Las excepciones proporcionan una manera limpia de verificar errores, eso es, sin abarrotar el código básico de una aplicación utilizando sistemáticamente los códigos de retorno de los métodos en sentencias if y switch para controlar los posibles errores que se puedan dar.

ESXCEPCIONES DE JAVA

ArithmeticExcepction » Una condicion aritmetica excepcional ha ocurrido. Por ejemplo una division por 0

ArrayIndexOutOfBoundsException » Una matriz fue accedida con un indice ilegal (fuera de los limites permitidos).

NullPointerException » Se intento utilizar null donde se requeria un objeto

NumberFormatException » Se intento convertir una cadena con un formato inapropiado en un numero.

Las excepciones en Jaca son objetos de cases derivas de la clase Throwable definida en el paquete java.lang. Por ejemplo, cuando se lanza una excepción ArithmeticAException, automáticamente java crea un objeto de esta clase.