CLASES Y MÉTODOS ABSTRACTOS EN JAVA.

Supongamos un esquema de herencia que consta de la clase Profesor de la que heredan ProfesorInterino y ProfesorTitular. Es posible que todo profesor haya de ser o bien ProfesorInterino o bien ProfesorTitular, es decir, que no vayan a existir instancias de la clase Profesor. Entonces, ¿qué sentido tendría tener una clase Profesor?

El sentido está en que una superclase permite unificar campos y métodos de las subclases, evitando la repetición de código y unificando procesos. Ahora bien, una clase de la que no se tiene intención de crear objetos, sino que únicamente sirve para unificar datos u operaciones de subclases, puede declararse de forma especial en Java: como clase abstracta. La declaración de que una clase es abstracta se hace con la sintaxis

public abstract class NombreDeLaClase { … }.

Por ejemplo public abstract class Profesor. Cuando utilizamos esta sintaxis, no resulta posible instanciar la clase, es decir, no resulta posible crear objetos de ese tipo. Sin embargo, sigue funcionando como superclase de forma similar a como lo haría una superclase "normal". La diferencia principal radica en que no se pueden crear objetos de esta clase.

Declarar una clase abstracta es distinto a tener una clase de la que no se crean objetos. En una clase abstracta, no existe la posibilidad. En una clase normal, existe la posibilidad de crearlos aunque no lo hagamos. El hecho de que no creemos instancias de una clase no es suficiente para que Java considere que una clase es abstracta. Para lograr esto hemos de declarar explícitamente la clase como abstracta mediante la sintaxis que hemos indicado. Si una clase no se declara usando abstract se cataloga como "clase concreta". Una clase abstracta para Java es una clase de la que nunca se van a crear instancias: simplemente va a servir como superclase a otras clases. No se puede usar la palabra clave new aplicada a clases abstractas.

A su vez, las clases abstractas suelen contener métodos abstractos: la situación es la misma. Para que un método se considere abstracto ha de incluir en su firma la palabra clave abstract. Además un método abstracto tiene estas peculiaridades:

a) No tiene cuerpo (llaves): sólo consta de signatura con paréntesis.

b) Su signatura termina con un punto y coma.

c) Sólo puede existir dentro de una clase abstracta. De esta forma se evita que haya métodos que no se puedan ejecutar dentro de clases concretas. Visto de otra manera, si una clase incluye un método abstracto, forzosamente la clase será una clase abstracta.

d) Los métodos abstractos forzosamente habrán de estar sobreescritos en las subclases. Si una subclase no implementa un método abstracto de la superclase tiene un método no ejecutable, lo que la fuerza a ser una subclase abstracta. Para que la subclase sea concreta habrá de implementar métodos sobreescritos para todos los métodos abstractos de sus superclases.

Un método abstracto para Java es un método que nunca va a ser ejecutado porque no tiene cuerpo. Simplemente, un método abstracto referencia a otros métodos de las subclases. ¿Qué utilidad tiene un método abstracto? Podemos ver un método abstracto como una palanca que fuerza dos cosas: la primera, que no se puedan crear objetos de una clase. La segunda, que todas las subclases sobreescriban el método declarado como abstracto.

Sintaxis tipo: abstract public/private/protected TipodeRetorno/void (parámetros …);

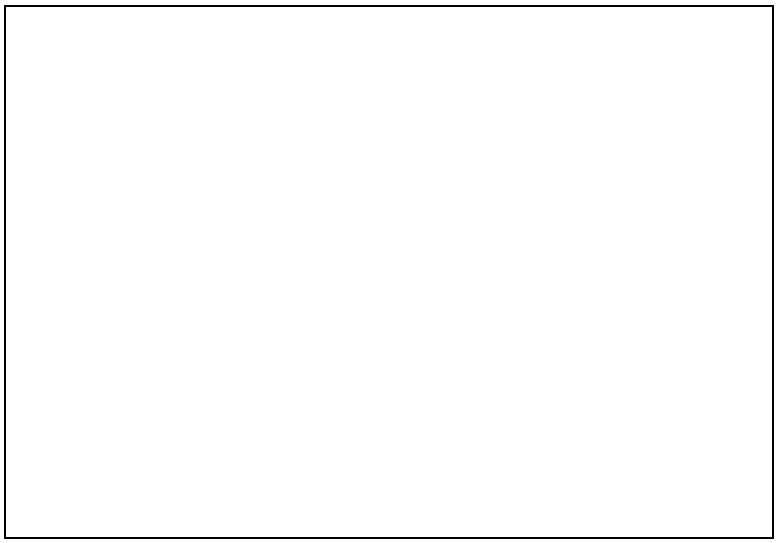
Por ejemplo: abstract public void generarNomina (int diasCotizados, boolean plusAntiguedad);

Que un método sea abstracto tiene otra implicación adicional: que podamos invocar el método abstracto sobre una variable de la superclase que apunta a un objeto de una subclase de modo que el método que se ejecute sea el correspondiente al tipo dinámico de la variable. En cierta manera, podríamos verlo como un método sobreescrito para que Java comprenda que debe buscar dinámicamente el método adecuado según la subclase a la que apunte la variable.

¿Es necesario que una clase que tiene uno o más métodos abstractos se defina como abstracta? Sí, si declaramos un método abstracto el compilador nos obliga a declarar la clase como abstracta porque si no lo hiciéramos así tendríamos un método de una clase concreta no ejecutable, y eso no es admitido por Java.

¿Una clase se puede declarar como abstracta y no contener métodos abstractos? Sí, una clase puede ser declarada como abstracta y no contener métodos abstractos. En algunos casos la clase abstracta simplemente sirve para efectuar operaciones comunes a subclases sin necesidad de métodos abstractos. En otros casos sí se usarán los métodos abstractos para referenciar operaciones en la clase abstracta al contenido de la sobreescritura en las subclases.

¿Una clase que hereda de una clase abstracta puede ser no abstracta? Sí, de hecho esta es una de las razones de ser de las clases abstractas. Una clase abstracta no puede ser instanciada, pero pueden crearse subclases concretas sobre la base de una clase abstracta, y crear instancias de estas subclases. Para ello hay que heredar de la clase abstracta y anular los métodos abstractos, es decir, implementarlos.



En este diagrama de clases vemos cómo hemos definido una clase abstracta denominada Profesor. Sin embargo, hereda de la clase Persona que no es abstracta, lo cual significa que puede haber instancias de Persona pero no de Profesor.

El test que hemos diseñado se basa en lo siguiente: ProfesorTitular y ProfesorInterino son subclases de la clase abstracta Profesor.

ListinProfesores sirve para crear un ArrayList de profesores que pueden ser tanto interinos como titulares y realizar operaciones con esos conjuntos. El listín se basa en el tipo estático Profesor, pero su contenido dinámico siempre será a base de instancias de ProfesorTitular o de ProfesorInterino ya que Profesor es una clase abstracta, no instanciable. En la clase de test creamos profesores interinos y profesores titulares y los vamos añadiendo a un listín. Posteriormente, invocamos el método imprimirListin, que se basa en los métodos toString de las subclases y de sus superclases mediante invocaciones sucesivas a super.

Ejercicio

Plantear una clase abstracta llamada 'Operacion' que defina los atributos enteros:

```
valor1
valor2
resultado
```

Define el método abstracto:

```
operar
```

Defina el método concreto imprimir que muestre el atributo resultado:

```
imprimir
```

Declarar cuatro clases que hereden de la clase Operacion llamadas: Suma, Resta, Multiplicacion y Division.

Crear un objeto de cada clase que hereda de Operacion.