MODIFICADORES DE ACCESO (visibilidad)

Los modificadores de acceso (visibilidad) son propiedades que se les puede dar a clases, métodos o atributos, las funciones/procedimientos también son métodos, y las variables son atributos. Dicho esto, veamos de menor a mayor visibilidad que modificadores tenemos:

-private

El modificador *private* solo se le puede otorgar a métodos (funciones/procedimientos) y a atributos (variables), veamos cómo afecta a cada uno:

- Métodos:
    - Solamente la clase que lo contiene puede utilizar este método.
- Atributos:
    - Solo la clase que lo contiene puede verlo y modificarlo.

***-friendly* (sin modificador)**

Cuando no especificamos la visibilidad de una clase, método o atributo, es de tipo *friendly*. Las clases, métodos y atributos con este modificador solo pueden ser usadas desde clases pertenecientes al mismo paquete.

- Clases:
    - Solamente las clases del mismo paquete podrán usar sus métodos y atributos, siempre y cuando éstos permitan ser usados por clases del mismo paquete.
- Métodos:
    - Solamente las clases del mismo paquete podrán usar este método.
- Atributos:
    - Solamente las clases del mismo paquete podrán verlo y modificarlo.

-protected

Se emplea en la herencia, es algo que se entenderá más adelante, pero acordaos de que la explicación está aquí, sus propiedades se parecen al modificador *friendly*:

- Métodos:
    - Las clases de mismo paquete o las subclases pueden usar éste método.
- Atributos:
    - Las clases de mismo paquete o las subclases pueden verlo o modificarlo.

-public

Nivel máximo de visibilidad, es aplicable a clases, métodos y atributos:

- Clases:
    - Todas las clases pueden usar sus métodos y atributos.
- Métodos:
    - Todas las clases pueden usar este método.
- Atributo:
    - Todas las clases pueden verlo y modificarlo.

MODIFICADOR DE ACCESO	EFECTO	APLICABLE A
private	Restringe la visibilidad al **interior de la clase**. Un atributo o método definido como private solo puede ser usado en el interior de su propia clase.	Atributos Métodos
<Sin modificador>	Cuando no se especifica un modificador, el elemento adquiere el *acceso por defecto o friendly*. También se le conoce como acceso de package (paquete). Solo puede ser usado por las **clases dentro de su mismo paquete**.	Clases Atributos Métodos
protected	Se emplea en la herencia. El elemento puede ser utilizado por **cualquier clase dentro de su paquete y por cualquier subclase** independientemente del paquete donde se encuentre.	Atributos Métodos
public	Es el nivel máximo de visibilidad. El elemento es visible desde cualquier clase.	Clases Atributos Métodos

http://puntocomnoesunlenguaje.blogspot.com/2012/07/clases-y-objetos-en-java.html

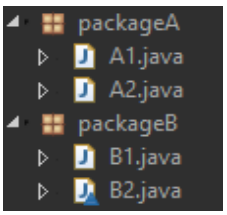
CONCEPTOS

- Una clase puede ser del tipo *friendly* o *public*, el hecho es que, si es de tipo *friendly*, da igual si a sus métodos o atributos le ponemos *protected* o *public*, cuando una clase es de tipo *friendly*, sus atributos y métodos solo pueden ser *private* o *friendly*, y si le ponemos una visibilidad superior (*protected/public*) éstos serán de tipo *friendly*.

  Por lo que nosotros siempre pondremos las clases del tipo *public*, a no ser que estemos seguros que esa clase no se va a usar fuera del paquete donde se encuentra.

- Los atributos de un objeto y los métodos auxiliares son siempre del tipo ***private***.

Recomiendo ver primero → "16. Herencia y Polimorfismo" para lo que queda de PDF

- 

Si nuestra clase *A1.java* tiene atributos del tipo *protected*, y la clase *B1.java* hereda de *A1*, *B1* solo puede modificar los atributos de Objetos del tipo *B1*, por lo que, si creamos un objeto del tipo *A1* en *B1,* a pesar de que *B1* herede de *A1*, *B1* no puede modificar objetos del tipo *A1.* Sé que es un lío, pero si lo leéis unas cuantas veces creo que lo entenderéis.

OTROS MODIFICADORES

Aunque hay otros 6 tipos de modificadores, solo vamos a ver tres; *abstract, final* y *static*:

-abstract

Este modificador se puede aplicar a clases y métodos, he indica que un método es solo un concepto, es decir, que no es real, que es abstracto:

- Clases:
    - No se pueden crear objetos en una clase abstracta.
    - Solo pueden ser usadas como superclases.
- Métodos:
    - Solamente se define la función del método.
    -

-final

Este varía mucho en base a donde se aplique:

- Clases:
    - No puede tener subclases, es decir, ninguna clase puede heredar de ésta.
- Métodos:
    - No puede ser sobrescrito en una subclase.
- Atributo:
    - Su valor no se puede modificar, y se debe inicializar cuando se crea.

-static

Es como diferenciamos si un atributo o variable es de una clase o de un objeto:

- Métodos:
    - Se puede invocar sin crear un objeto de su clase.
- Atributos:
    - Si no lleva *static* es un atributo, con un valor diferente para cada objeto.
    - Si lleva *static*, es una variable de clase, con un solo valor para todos los objetos.

MODIFICADOR	EFECTO	APLICABLE A
abstract	Aplicado a una clase, la declara como clase abstracta. No se pueden crear objetos de una clase abstracta. Solo pueden usarse como superclases. Aplicado a un método, la definición del método se hace en las subclases.	Clases Métodos
final	Aplicado a una clase significa que no se puede extender (heredar), es decir que no puede tener subclases. Aplicado a un método significa que no puede ser sobrescrito en las subclases. Aplicado a un atributo significa que contiene un valor constante que no se puede modificar	Clases Atributos Métodos
static	Aplicado a un atributo indica que es una variable de clase. Esta variable es única y compartida por todos los objetos de la clase. Aplicado a un método indica que se puede invocar sin crear ningún objeto de su clase.	Atributos Métodos

http://puntocomnoesunlenguaje.blogspot.com/2012/07/clases-y-objetos-en-java.html

- Los otros tres modificadores son; *volatile, transient, Sincronizad*, pero no nos interesan
- Las clases, métodos y atributos pueden tener varios modificadores, pero solo un modificador de acceso o visibilidad.
- Los métodos *abstract* tienen ésta forma → *public abstract int method();*
  Como veis especificamos todo menos el interior del método.
- Si declaramos una variable dentro de un método, nunca lleva *static*.
- Si un método es del tipo *abstract*, su clase también lo es obligatoriamente.
- Cuando un método es del tipo *abstract*, es obligatorio sobrescribirlo en sus subclases.