Control de acceso

Hasta el momento hemos visto que desde cualquier clase se puede acceder a cualquier otra clase y a cualquier atributo o método sin ningún control. Esto lo hemos hecho de este modo para facilitar la comprensión de los distintos conceptos tratados a lo largo del curso y de este modo no distraernos con cuestiones de control de acceso pero a partir de este momento vamos a hablar del **control de acceso** en clases, atributos, constructores y métodos para así empezar a desarrollar nuestros programas de un modo mas adecuado.

a clases

Cuando definimos una clase podemos hacerlo con la palabra reservada public, abstract, final o ninguna de los anteriores. Existe alguna otra posibilidad pero no entraremos en ella.

|Clase|Descripción| |:--|:--| |public|Una clase pública es accesible desde cualquier clase. Para ser utilizada desde otro paquete debe ser importada.| |abstract|Una clase abstracta es una clase destinada a que se herede de ella. No puede ser instanciada. Debe tener al menos un método abstracto.| |final|Una clase final es aquella que no permite que se herede de ella.|

Así podemos combinar las distintas palabras clave:

```
public class Ejemplo1{

class Ejemplo2{

public abstract class Ejemplo3{

abstract class Ejemplo4{

public final class Ejemplo5{

final class Ejemplo6{
```

En las 6 líneas anteriores tenemos todas las combinaciones con las que vamos a trabajar en el curso.

La clase Ejemplo1 es una clase pública por lo que será accesible desde cualquier paquete y se podrán crear instancias de ella y usarla como superclase.

La clase Ejemplo2 será accesible únicamente desde el paquete en que sea definida y se podrán crear instancias de ella y usarla como superclase.

La clase Ejemplo3 es una clase pública por lo que será accesible desde cualquier paquete y al ser abstracta no se podrán crear instancias de ella, deberá contener al menos un método abstracto y se podrá usar como superclase.

La clase Ejemplo4 será accesible únicamente desde el paquete en que sea definida y al ser abstracta no se podrán crear instancias de ella, deberá contener al menos un método abstracto y se podrá usar como superclase.

La clase Ejemplo5 es una clase pública por lo que será accesible desde cualquier paquete pudiendo ser instanciada pero no usada como superclase.

La clase ejemplo6 será accesible únicamente desde el paquete en que sea definida pudiendo ser instanciada pero no usada como superclase.

a atributos

Existen las siguientes palabras clave cuando queremos controla el acceso a las variables miembro de nuestras clases. Vamos a verlas:

|Atributo|Descripción| |:--|:--| |public|El campo es accesible desde todas las clases| |private|El campo es accesible únicamente desde la propia clase| |final|El campo no puede ser modificado y al definirse debe establecerse su valor.|

Vamos a ver algún ejemplo:

public String ejemplo1;

String ejemplo2;

private String ejemplo3;

public final String ejemplo4 = "trivinet.com";

final String ejemplo5 = "recurso didáctico gratuito";

private final String ejemplo6 = "de alta calidad";

Las 6 combinaciones anteriores serían todas las posibles.

ejemplo1 sería accesible y modificable desde cualquier clase.

ejemplo2 sería accesible únicamente desde las clases que pertenezcan al mismo paquete y cualquiera de ellas podría modificarla.

ejemplo3 sería accesible únicamente desde la clase en que se ha definido y podría ser modificada.

ejemplo4 sería accesible desde cualquier clase. No se podría modificar su valor.

ejemplo5 sería accesible únicamente desde las clases que pertenezcan al mismo paquete. No se podría modificar su valor.

ejemplo6 sería accesible únicamente desde la clase en que se ha definido. No se podría modificar su valor.

Es común que los campos se definan como privados para así sacar todo el potencial de la **encapsulación** (la veremos en el siguiente capítulo).

a constructores

|Constructor|Descripción| |:--|:--| |public|Cualquier clase puede crear instancias de la clase que contenga un constructor público.| |protected|Solamente las subclases de la clase que contiene un constructor protegido pueden crear instancias de la clase.| |private|Ninguna clase puede crear instancias de una clase que tiene un constructor privado excepto la propia clase a través de alguno de sus métodos públicos.|

El patrón de diseño singleton es un ejemplo a través del cual ver la utilidad de definir un constructor privado.

a métodos

A la hora de establecer el control de acceso a un método en el curso vamos a trabajar con public, private, protected, abstract y final.

|Método|Descripción| |:--|:--| |public|El método es accesible desde la propia clase, el paquete al que pertenezca la clase, las subclases y el resto del mundo.| |protected|El método es accesible desde la propia clase, el paquete al que pertenezca la clase y las subclases.| |private|El método es accesible desde la propia clase.| |abstract|El método no está definido en esta clase sino que será definido en la clase que herede de la clase que contenga este método.| |final|El método no puede ser sobreescrito.|

Cuando un método es abstracto no se pone el bloque de código que va entre las llaves ({}) ni las propias llaves, esto tiene sentido porque el método en cuestión debe definirse en la clase que herede la clase que contenga el método abstracto.

Vamos a ver algunas combinaciones:

public void ejemplo1(){

protected void ejemplo2(){

void ejemplo3(){

private void ejemplo4(){

public abstract void ejemplo5();

protected abstract void ejemplo6();

abstract void ejemplo7();

public final void ejemplo8(){

protected final void ejemplo9(){

final void ejemplo10(){

private final void ejemplo11(){

En los 11 ejemplos anteriores no hay valor de retorno (void) y como puede apreciarse no existe la combinación private abstract ya que no tendría sentido. Existe alguna otra palabra reservada que da lugar a mas combinaciones pero estas son suficientes para el objetivo del curso.

En la documentación oficial de Java sobre el control de acceso podemos encontrar mas información (en inglés).

Revision #1
Created 1 February 2022 11:11:20 by Equipo CATEDU
Updated 1 February 2022 11:11:20 by Equipo CATEDU