MODIFICADORES DE ACCESO EN JAVA.

Encapsulamiento en Java

Los *modificadores* de acceso nos introducen al concepto de ***encapsulamiento***. El encapsulamiento busca de alguna forma controlar el *acceso a los datos* que conforman un objeto o instancia, de este modo podríamos decir que los objetos que hacen uso de modificadores de acceso (especialmente privados) son objetos encapsulados.

Los modificadores de acceso permiten dar un nivel de seguridad mayor a nuestras aplicaciones restringiendo el acceso a diferentes atributos, métodos, constructores asegurándonos que el usuario deba seguir una "ruta" especificada por nosotros para acceder a la información.

Es muy posible que nuestras aplicaciones vayan a ser usadas por otros programadores o usuarios con cierto nivel de experiencia; haciendo uso de los modificadores de acceso podremos asegurar que un valor no será modificado incorrectamente por parte de otro programador. Generalmente, el acceso a los atributos se consigue por medio de los métodos get y set, pues es necesario que los atributos de una clase sean privados.

Nota: Siempre se recomienda que los atributos de una clase sean privados y por tanto cada atributo debe tener sus propios métodos get y set para obtener y establecer respectivamente el valor del atributo.

Nota 2: Siempre que se use una clase de otro paquete, se debe importar usando **import**. Cuando dos clases se encuentran en el mismo paquete no es necesario importar, pero esto no significa que se pueda acceder a sus componentes directamente.

Veamos un poco en detalle cada uno de los modificadores de acceso.

Vamos a probar todos los ejemplos, pero a diferencia de lo que se muestra en este texto, algunos los ponemos no estáticos.

Modificador de acceso private

El modificador ***private*** en Java es el más restrictivo de todos, básicamente cualquier elemento de una clase que sea privado puede ser accedido únicamente por la misma clase. Es decir, si por ejemplo un atributo es privado, solo puede ser accedido por los métodos o constructores de la misma clase. Ninguna otra clase, sin importar la relación que tengan podrá tener acceso a ellos.

```
package ejemplo1;

public class Ejemplo1
{
   private int atributo1; //Este atributo es privado
   private int contador = 0; //Contador de registro

   //Si un atributo es privado podemos crear método get y set
   //para él y permitir el acceso desde otras instancias

   public void setAtributo1(int valor)
   {
      contador++; // Contador que lleva el registro de ediciones del atributo1
      atributo1 = valor; // Establecemos el valor del atributo
   }

   public int getAtributo1()
   {
      return atributo1;//Retornamos el valor actual del atributo
   }

   //Get para el contador
   public int getContador()
   {
      return contador;
   }
   // Notar que no ponemos un set, pues no nos interesa que el contador
   // pueda ser cambiado.
}
```

En el ejemplo anterior, tenemos un atributo privado y permitimos el acceso a él únicamente por medio de los métodos de get y set, notemos que estos métodos son públicos y por tanto cualquiera puede acceder a ellos.

Lo realmente interesante con los métodos get y set es que nos permiten realizar cualquier operación, como, por ejemplo, llevar una cuenta de las veces que se estableció el valor para el atributo permitiéndonos mantener nuestro sistema sin problemas, validar que el valor que se quiere dar a un atributo cumple las especificaciones dadas... También debemos notar que debido a que los métodos get y set son propios de la clase no tienen problemas con acceder al atributo directamente.

Modificador por defecto (default o paquete)

Java nos da la opción de no usar un modificador de acceso y al no hacerlo, el elemento tendrá un acceso conocido como ***default*** o acceso por defecto. Este modo de acceso permite que tanto la propia clase como las clases del mismo paquete accedan a dichos componentes (de aquí la importancia de declarar siempre en un paquete nuestras clases).

```
package ejemplo2;

public class Ejemplo2
{
   private static int atributo1; //Este atributo es privado
   static int contador = 0; //Contador con acceso por defecto

   public static void setAtributo1(int valor)
   {
      contador++; //Contador que lleva el registro de ediciones del atributo1
      atributo1 = valor; //Establecemos el valor del atributo
   }

   public static int getAtributo1()
   {
      return atributo1;//Retornamos el valor actual del atributo
   }
}
```

```
package ejemplo2;

public class Ejemplo2_1
{
   public static int getContador()
   {
         return Ejemplo2.contador;
        // Accedemos directamente al contador desde otra clase
   }
}
```

Modificador de acceso protected

El modificador de acceso ***protected*** nos permite acceso a los componentes con dicho modificador desde la misma clase, clases del mismo paquete y clases que hereden de ella (incluso en diferentes paquetes). Veamos:

```
package ejemplo3;

public class Ejemplo3
{
   protected static int atributo1;//Atributo protected
   private static int atributo2; //Atributo privado
   int atributo3;//Atributo por default

   public static int getAtributo2()
   {
      return atributo2;
   }
}
```

```
package ejemplo3_1;

import ejemplo3.Ejemplo3;//Es necesario importar la clase del ejemplo 3

public class Ejemplo3_1 extends Ejemplo3
{
   public static void main(String[] args)
   {
      // La siguientes dos líneas generan error, pues atributo2 es privado y
      // atributo 3 es default y está en otro paquete

      //System.out.println(atributo2);
      //System.out.println(atributo3);

      System.out.println(atributo1); //Sí tenemos acceso a atributo1
   }
}
```

Modificador public

El modificador de acceso ***public*** es el más permisivo de todos, esto quiere decir que si un componente de una clase es public, tendremos acceso a él desde cualquier clase o instancia sin importar el paquete o procedencia de esta.

```
package ejemplo4;

public class Ejemplo4
{
   public static int atributo1; //Atributo publico

   public static void metodo1()
   {
      System.out.println("Método publico");
   }
}
```

```
package ejemplo4_1;

import ejemplo4.Ejemplo4; //importamos la clase del ejemplo4

public class ClaseExterna
{
   public static void main(String[] args)
   {
      System.out.println(Ejemplo4.atributo1);
      //Tenemos acceso directo por ser publico

      Ejemplo4.metodo1(); //Metodo1 también es publico
   }
}
```

En el caso del modificador de acceso protected y el acceso desde subclases, es un error común pensar que se puede crear un objeto de la clase padre en una clase hija y luego acceder al atributo con acceso protected sin problemas. Esto no es cierto, puesto que **el modificador protected lo que nos permite es acceder al atributo heredado desde el ámbito de la clase hija y no directamente.**

Para que quede más claro este caso, accede a esta información:

https://www.youtube.com/watch?v=cPcjiwSN1YU&t=7s

https://www.programarya.com/Cursos/Java/Modificadores-de-Acceso

https://www.youtube.com/watch?v=D7nGxCS0v-E

Tabla que resume el funcionamiento de los modificadores de acceso en Java:

	public	protected	(sin modificador)	private
Clase	SI	SI	SI	SI
Subclase en el mismo paquete	SI	SI	SI	NO
No-Subclase en el mismo paquete	SI	SI	SI	NO
Subclase en diferente paquete	SI	SI/NO **(*)**	NO	NO
No-Subclase en diferente paquete (Universo)	SI	NO	NO	NO

(*) Los miembros (variables y métodos) de clase (es decir, static) sí son visibles. Los miembros de instancia no son visibles salvo si acceden a través de la herencia.