5.B. Atributos.

1. Atributos.

1.1. Declaración de atributos.

La sintaxis general para la declaración de un atributo en el interior de una clase es:

```
[modificadores] <tipo> <nombreAtributo>;
```

Ejemplos:

```
int x;
```

```
public int elementoX, elementoY;
```

```
private int x, y, z;
```

```
static double descuentoGeneral;
```

```
final bool casado;
```

Te suena bastante, ¿verdad? La declaración de los atributos en una clase es exactamente igual a la declaración de cualquier variable tal y como has estudiado en las unidades anteriores y similar a como se hace en cualquier lenguaje de programación. Es decir mediante la indicación del tipo y a continuación el nombre del atributo, pudiéndose declarar varios atributos del mismo tipo mediante una lista de nombres de atributos separada por comas (exactamente como ya has estudiado al declarar variables).

La declaración de un **atributo** (o **variable miembro** o **variable de objeto**) consiste en la declaración de una variable que únicamente existe en el interior del objeto y por tanto su vida comenzará cuando el objeto comience a existir (el objeto sea creado). Esto significa que cada vez que se cree un objeto se crearán tantas variables como atributos contenga ese objeto en su interior (definidas en la clase, que es la plantilla o "molde" del objeto). Todas esas variables estarán encapsuladas dentro del objeto y sólo tendrán sentido dentro de él.

En el ejemplo que estamos utilizando de objetos de tipo `Punto` (instancias de la clase `Punto`), cada vez que se cree un nuevo `Punto p1`, se crearán sendos atributos `x`, `y` de tipo `int` que estarán en el interior de ese punto `p1`. Si a continuación se crea un nuevo objeto `Punto p2`, se crearán otros dos nuevos atributos `x`, `y` de tipo `int` que estarán esta vez alojados en el interior de `p2`. Y así sucesivamente...

Dentro de la declaración de un atributo puedes encontrar tres partes:

- **Modificadores**. Son palabras reservadas que permiten modificar la utilización del atributo (indicar el control de acceso, si el atributo es constante, si se trata de un atributo de clase, etc.). Los iremos viendo uno a uno.

- **Tipo**. Indica el tipo del atributo. Puede tratarse de un tipo primitivo (`int`, `char`, `bool`, `double`, etc) o bien de uno referenciado (objeto, array, etc.).

- **Nombre**. Identificador único para el nombre del atributo. Por convenio se **suelen utilizar las minúsculas**. En caso de que se trate de un identificador que contenga varias palabras, **a partir de la segunda palabra se suele poner la letra de cada palabra en mayúsculas**. Por ejemplo: primerValor, valor, puertaIzquierda, cuartoTrasero, equipoVecendor, sumaTotal, nombreCandidatoFinal, etc. Cualquier identificador válido de Java será admitido como nombre de atributo válido, pero es importante seguir este convenio para facilitar la legibilidad del código (todos los programadores de Java lo utilizan).

Como puedes observar, los atributos de una clase también pueden contener modificadores en su declaración (como sucedía al declarar la propia clase). Estos modificadores permiten indicar cierto comportamiento de un atributo a la hora de utilizarlo. Entre los modificadores de un atributo podemos distinguir:

- **Modificadores de acceso**. Indican la forma de acceso al atributo desde otra clase. Son modificadores excluyentes entre sí. Sólo se puede poner uno.

- **Modificadores de contenido**. No son excluyentes. Pueden aparecer varios a la vez.

- **Otros modificadores**: `transient` y `volatile`. El primero se utiliza para indicar que un atributo es transitorio (no persistente) y el segundo

es para indicar al compilador que no debe realizar optimizaciones sobre esa variable. Es más que probable que no necesites utilizarlos en este módulo.

Aquí tienes la sintaxis completa de la declaración de un atributo teniendo en cuenta la lista de todos los modificadores e indicando cuáles son incompatibles unos con otros:

```
[private | protected | public] [static] [final] [transient] [volatile] <tipo> <nombreAtributo>;
```

Vamos a estudiar con detalle cada uno de ellos.