5.A. Introducción a las clases.

Sitio: Aula Virtual
Curso: Programación_DAM
Libro: 5.A. Introducción a las clases.
Imprimido por: LUIS PUJOL
Día: lunes, 6 de enero de 2020, 18:43

Tabla de contenidos

1 Concepto de clase.

Como ya has visto en anteriores unidades, las **clases** están compuestas por **atributos** y **métodos**. Una **clase** especifica las características comunes de un conjunto de **objetos**. De esta forma los programas que escribas estarán formados por un conjunto de **clases** a partir de las cuales irás creando **objetos** que se interrelacionarán unos con otros.

Recomendación

En esta unidad se va a utilizar el concepto de **objeto** así como algunas de las diversas **estructuras de control** básicas que ofrece cualquier lenguaje de programación. Todos esos conceptos han sido explicados y utilizados en las unidades anteriores. Si consideras que es necesario hacer un repaso del concepto de objeto o del uso de las estructuras de control elementales, éste es el momento de hacerlo.

1.1 Repaso del concepto de objeto.

Desde el comienzo del módulo llevas utilizando el concepto de **objeto** para desarrollar tus programas de ejemplo. En las unidades anteriores se ha descrito un objeto como una entidad que contiene **información** y que es capaz de realizar ciertas **operaciones** con esa información. Según los valores que tenga esa información el objeto tendrá un **estado** determinado y según las operaciones que pueda llevar a cabo con esos datos será responsable de un **comportamiento** concreto.

Recuerda que entre las características fundamentales de un objeto se encontraban la **identidad** (los objetos son únicos y por tanto distinguibles entre sí, aunque pueda haber objetos exactamente iguales), un **estado** (los atributos que describen al objeto y los valores que tienen en cada momento) y un determinado **comportamiento** (acciones que se pueden realizar sobre el objeto).

Algunos ejemplos de objetos que podríamos imaginar podrían ser:

- Un coche de color rojo, marca SEAT, modelo Toledo, del año 2003. En este ejemplo tenemos una serie de atributos, como el color (en este caso rojo), la marca, el modelo, el año, etc. Así mismo también podríamos imaginar determinadas características como la cantidad de combustible que le queda, o el número de kilómetros recorridos hasta el momento.
- Un coche de color amarillo, marca Opel, modelo Astra, del año 2002.
- Otro coche de color amarillo, marca Opel, modelo Astra y también del año 2002. Se trataría de otro objeto con las mismas propiedades que el anterior, pero sería un segundo objeto.
- Un cocodrilo de cuatro metros de longitud y de veinte años de edad.
- Un círculo de radio 2 centímetros, con centro en las coordenadas (0,0) y relleno de color amarillo.
- Un círculo de radio 3 centímetros, con centro en las coordenadas (1,2) y relleno de color verde.

Si observas los ejemplos anteriores podrás distinguir sin demasiada dificultad al menos tres familias de objetos diferentes, que no tienen nada que ver una con otra:

- Los coches.
- Los círculos.
- Los cocodrilos.

Es de suponer entonces que cada objeto tendrá determinadas posibilidades de **comportamiento** (**acciones**) dependiendo de la familia a la que pertenezcan. Por ejemplo, en el caso de los **coches** podríamos imaginar acciones como: arrancar, frenar, acelerar, cambiar de marcha, etc. En el caso de

los **cocodrilos** podrías imaginar otras acciones como: desplazarse, comer, dormir, cazar, etc. Para el caso del **círculo** se podrían plantear acciones como: cálculo de la superficie del círculo, cálculo de la longitud de la circunferencia que lo rodea, etc.

Por otro lado, también podrías imaginar algunos **atributos** cuyos valores podrían ir cambiando en función de las acciones que se realizaran sobre el objeto: ubicación del coche (coordenadas), velocidad instantánea, kilómetros recorridos, velocidad media, cantidad de combustible en el depósito, etc. En el caso de los cocodrilos podrías imaginar otros atributos como: peso actual, el número de dientes actuales (irá perdiendo algunos a lo largo de su vida), el número de presas que ha cazado hasta el momento, etc.

Como puedes ver, un **objeto** puede ser cualquier cosa que puedas describir en términos de **atributos** y **acciones**.

Un objeto no es más que la representación de cualquier entidad concreta o abstracta que puedas percibir o imaginar y que pueda resultar de utilidad para modelar los elementos el entorno del problema que deseas resolver.

Autoevaluación

Tenemos un objeto bombilla, de marca ACME, que se puede encender o apagar, que tiene una potencia de 50 vatios y ha costado 3 euros. La bombilla se encuentra en este momento apagada. A partir de esta información, ¿sabrías decir qué atributos y qué acciones (comportamiento) podríamos relacionar con ese objeto bombilla?

- Objeto bombilla con atributos potencia (50 vatios), precio (3 euros), marca (ACME) y estado (apagada). Las acciones que se podrían ejercer sobre el objeto serían encender y apagar.
- Objeto bombilla con atributos precio (3 euros), marca (ACME) y apagado. Las acciones que se podrían ejercer sobre el objeto serían encender y apagar.
- Objeto bombilla con atributos precio (3 euros), marca (ACME), potencia (50 vatios) y estado (apagada). No se puede ejercer ninguna acción sobre el objeto.
- Se trata de un objeto bombilla cuyas posibles acciones son encender, apagar y arreglar. Sus atributos serían los mismos que en el caso a).

1.2 El concepto de clase.

Está claro que dentro de un mismo programa tendrás la oportunidad de encontrar decenas, cientos o incluso miles de objetos. En algunos casos no se parecerán en nada unos a otros, pero también podrás observar que habrá muchos que tengan un gran parecido, compartiendo un mismo comportamiento y unos mismos atributos. Habrá muchos objetos que sólo se diferenciaran por los valores que toman algunos de esos atributos.

Es aquí donde entra en escena el concepto de **clase**. Está claro que no podemos definir la estructura y el comportamiento de cada objeto cada vez que va a ser utilizado dentro de un programa, pues la escritura del código sería una tarea interminable y redundante. La idea es poder disponer de una **plantilla** o **modelo** para cada conjunto de objetos que sean del mismo tipo, es decir, que tengan los mismos atributos y un comportamiento similar.

Una clase consiste en la definición de un tipo de objeto. Se trata de una descripción detallada de cómo van a ser los objetos que pertenezcan a esa clase indicando qué tipo de información contendrán (atributos) y cómo se podrá interactuar con ellos (comportamiento).

Como ya has visto en unidades anteriores, una clase consiste en un plantilla en la que se especifican:

- Los **atributos** que van a ser comunes a todos los objetos que pertenezcan a esa clase (información).

- Los **métodos** que permiten interactuar con esos objetos (comportamiento).

A partir de este momento podrás hablar ya sin confusión de objetos y de clases, sabiendo que los primeros son instancias concretas de las segundas, que no son más que una abstracción o definición.

Si nos volvemos a fijar en los ejemplos de objetos del apartado anterior podríamos observar que las clases serían lo que clasificamos como "familias" de objetos (coches, cocodrilos y círculos).

En el lenguaje cotidiano de muchos programadores puede ser habitual la confusión entre los términos clase y objeto. Aunque normalmente el contexto nos permite distinguir si nos estamos refiriendo realmente a una clase (definición abstracta) o a un objeto (instancia concreta), hay que tener cuidado con su uso para no dar lugar a interpretaciones erróneas, especialmente durante el proceso de aprendizaje.

Autoevaluación

Un objeto y una clase en realidad hacen referencia al mismo concepto. Podría decirse que son sinónimos. ¿Verdadero o falso?

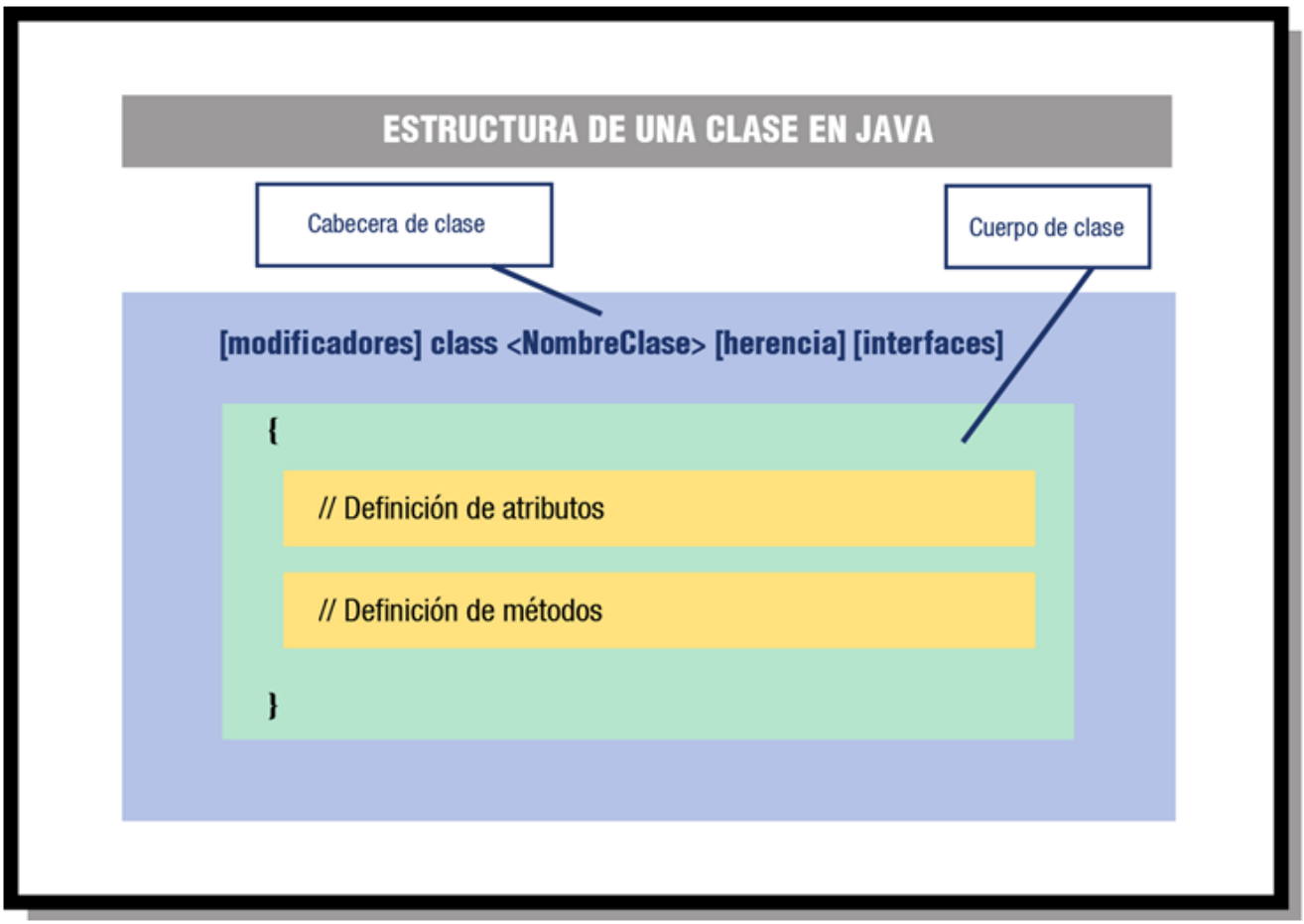
- ○ Verdadero.
- ○ Falso.

2 Estructura y miembros de una clase.

En unidades anteriores ya se indicó que para declarar una clase en Java se usa la palabra reservada `class`. En la declaración de una clase vas a encontrar:

- **Cabecera de la clase**. Compuesta por una serie de modificadores de acceso, la palabra reservada `class` y el nombre de la clase.

- **Cuerpo de la clase**. En él se especifican los distintos miembros de la clase: **atributos** y **métodos**. Es decir, el contenido de la clase.



Como puedes observar, el **cuerpo de la clase** es donde se declaran los **atributos** que caracterizan a los objetos de la clase y donde se define e implementa el comportamiento de dichos objetos; es decir, donde se declaran e implementan los **métodos**.

Autoevaluación

Toda definición de una clase consta de cabecera y cuerpo. En la cabecera se definen los atributos de los objetos que se crearán a partir de esa clase y en el cuerpo estarán definidos los distintos métodos disponibles para manipular esos objetos. ¿Verdadero o falso?

- ○ Verdadero.
- ○ Falso.

2.1 Declaración de una clase.

La declaración de una clase en Java tiene la siguiente estructura general:

```
[modificadores] class  <NombreClase> [herencia] [interfaces] { // Cabecera de la clase
```

```
    // Cuerpo de la clase
```

```
    Declaración de los atributos
```

```
    Declaración de los métodos
```

```
}
```

Un ejemplo básico pero completo podría ser:

```
/**
 *
 * Ejemplo de clase Punto 2D
 */
class Punto {
    // Atributos
    int x,y;

    // Métodos
    int obtenerX () { return x; }
    int obtenerY()  {return y;}
    void establecerX (int vx) { x= vx; };
    void establecerY (int vy) { y= vy; };
}
```

El mismo código copiable:

```
/**
 *
 * Ejemplo de clase Punto
 */


class Punto {
    // Atributos
    int x,y;


    // Métodos
    int obtenerX () { return x; }
    int obtenerY()  {return y;}
    void establecerX (int vx) { x= vx; };
    void establecerY (int vy) { y= vy; };
```

```
}
```

En este caso se trata de una clase muy sencilla en la que el cuerpo de la clase (el área entre las llaves) contiene el código y las declaraciones necesarias para que los objetos que se construyan (basándose en esta clase) puedan funcionar apropiadamente en un programa (declaraciones de atributos para contener el estado del objeto y métodos que implementen el comportamiento de la clase y los objetos creados a partir de ella).

Si te fijas en los distintos programas que se han desarrollado en los ejemplos de las unidades anteriores, podrás observar que cada uno de esos programas era en sí mismo una clase Java: se declaraban con la palabra reservada `class` y contenían algunos atributos (variables) así como algunos métodos (como mínimo el método `main`).

En el ejemplo anterior hemos visto lo mínimo que se tiene que indicar en la **cabecera de una clase** (el nombre de la clase y la palabra reservada `class`). Se puede proporcionar bastante más información mediante modificadores y otros indicadores como por ejemplo el nombre de su **superclase** (si es que esa clase hereda de otra), si implementa algún **interfaz** y algunas cosas más que irás aprendiendo poco a poco.

A la hora de implementar una clase Java (escribirla en un archivo con un editor de textos o con alguna herramienta integrada como por ejemplo **Netbeans** o **Eclipse**) debes tener en cuenta:

- Por convenio, se ha decidido que en lenguaje Java los nombres de las clases deben de **empezar por una letra mayúscula**. Así, cada vez que observes en el código una palabra con la primera letra en mayúscula sabrás que se trata de una clase sin necesidad de tener que buscar su declaración. Además, **si el nombre de la clase está formado por varias palabras, cada una de ellas también tendrá su primera letra en mayúscula**. Siguiendo esta recomendación, algunos ejemplos de nombres de clases podrían ser: Recta, Circulo, Coche, CocheDeportivo, Jugador, JugadorFutbol, AnimalMarino, AnimalAcuatico, etc.

- El archivo en el que se encuentra una clase Java debe tener el mismo nombre que esa clase si queremos poder utilizarla desde otras clases que se encuentren fuera de ese archivo (**clase principal del archivo**).

- Tanto la definición como la implementación de una clase se incluye en el mismo archivo (archivo ".java"). En otros lenguajes como por ejemplo C++, definición e implementación podrían ir en archivos separados (por ejemplo en C++, serían sendos archivos con extensiones ".h" y ".cpp").

Para saber más

Si quieres ampliar un poco más sobre este tema puedes echar un vistazo a los tutoriales de iniciación de Java en el sitio web de Oracle (en inglés):

Java Classes.

2.2 Cabecera de una clase.

En general, la declaración de una clase puede incluir los siguientes elementos y en el siguiente orden:

Modificadores tales como `public`, `abstract` o `final`.

El nombre de la clase (con la primera letra de cada palabra en mayúsculas, por convenio).

El nombre de su **clase padre** (**superclase**), si es que se especifica, precedido por la palabra reservada `extends` ("extiende" o "hereda de").

Una lista separada por comas de **interfaces** que son implementadas por la clase, precedida por la palabra reservada `implements` ("implementa").

El cuerpo de la clase, encerrado entre llaves {}.

La sintaxis completa de una cabecera (los cuatro primeros puntos) queda de la forma:

```
[modificadores]
```

```
class <NombreClase> [extends <NombreSuperClase>][implements
```

```
<NombreInterface1>] [[implements <NombreInterface2>] ...] {
```

En el ejemplo anterior de la clase `Punto` teníamos la siguiente cabecera:

```
class Punto {
```

En este caso no hay **modificadores**, ni indicadores de **herencia**, ni implementación de **interfaces**. Tan solo la palabra reservada `class` y el nombre de la clase. Es lo mínimo que puede haber en la cabecera de una clase.

La **herencia** y las **interfaces** las verás más adelante. Vamos a ver ahora cuáles son los **modificadores** que se pueden indicar al crear la clase y qué efectos tienen. Los **modificadores de clase** son:

```
[public] [final | abstract]
```

Veamos qué significado tiene cada uno de ellos:

- Modificador `public`. Indica que la clase es visible (se pueden crear objetos de esa clase) desde cualquier otra clase. Es decir, desde cualquier otra parte del programa. Si no se especifica este modificador, la clase sólo podrá ser utilizada desde clases que estén en el mismo **paquete**. El concepto de paquete lo veremos más adelante. Sólo puede haber una clase `public` (clase principal) en un archivo .java. El resto de clases que se definan en ese archivo no serán públicas.

- Modificador `abstract`. Indica que la clase es **abstracta**. Una clase abstracta no es instanciable. Es decir, no es posible crear objetos de esa clase (habrá que utilizar clases que hereden de ella). En este momento es posible que te parezca que no tenga sentido que esto pueda suceder (si no puedes crear objetos de esa clase, ¿para qué la quieres?), pero puede resultar útil a la hora de crear una jerarquía de clases. Esto lo verás también más adelante al estudiar el concepto de **herencia**.

- Modificador `final`. Indica que no podrás crear clases que hereden de ella. También volverás a este modificador cuando estudies el concepto de **herencia**. Los modificadores `final` y `abstract` son excluyentes (sólo se puede utilizar uno de ellos).

Todos estos modificadores y palabras reservadas las iremos viendo poco a poco, así que no te preocupes demasiado por intentar entender todas ellas en este momento.

En el ejemplo anterior de la clase `Punto` tendríamos una clase que sería sólo visible (utilizable) desde el mismo paquete en el que se encuentra la clase (modificador de acceso por omisión o de paquete, o `package`). Desde fuera de ese paquete no sería visible o accesible. Para poder utilizarla desde cualquier parte del código del programa bastaría con añadir el atributo public: `public class Punto`.

Autoevaluación

Si queremos poder instanciar objetos de una clase desde cualquier parte de un programa, ¿qué modificador o modificadores habrá que utilizar en su declaración?

- ○ private.
- ○ public.
- ○ abstract.
- ○ Ninguno de los anteriores.

2.3 Cuerpo de una clase.

Como ya has visto anteriormente, el cuerpo de una clase se encuentra encerrado entre llaves y contiene la declaración e implementación de sus miembros. Los miembros de una clase pueden ser:

- **Atributos**, que especifican los datos que podrá contener un objeto de la clase.

- **Métodos**, que implementan las acciones que se podrán realizar con un objeto de la clase.

Una clase puede no contener en su declaración atributos o métodos, pero debe de contener al menos uno de los dos (la clase no puede ser vacía).

En el ejemplo anterior donde se definía una clase `Punto`, tendríamos los siguientes atributos:

- Atributo `x`, de tipo `int`.
- Atributo `y`, de tipo `int`.

Es decir, dos valores de tipo entero. Cualquier objeto de la clase `Punto` que sea creado almacenará en su interior dos números enteros (`x` e `y`). Cada objeto diferente de la clase `Punto` contendrá sendos valores **x** e **y**, que podrán coincidir o no con el contenido de otros objetos de esa misma clase `Punto`.

Por ejemplo, si se han declarado varios objetos de tipo `Punto`:

```
Punto p1, p2, p3;
```

Sabremos que cada uno de esos objetos `p1`, `p2` y `p3` contendrán un par de coordenadas `(x, y)` que definen el estado de ese objeto. Puede que esos valores coincidan con los de otros objetos de tipo `Punto`, o puede que no, pero en cualquier caso serán objetos diferentes creados a partir del mismo molde (de la misma clase).

Por otro lado, la clase `Punto` también definía una serie de métodos:

```
    * int obtenerX () { return x; }
```

```
    * int obtenerY() { return y;}
```

```
    * void establecerX (int vx) { x= vx; };
```

```
    * void establecerY (int vy) { y= vy; };
```

Cada uno de esos métodos puede ser llamado desde cualquier objeto que sea una instancia de la clase `Punto`. Se trata de operaciones que permiten manipular los datos (atributos) contenidos en el objeto bien para calcular otros datos o bien para modificar los propios atributos.

Autoevaluación

Si disponemos de varios objetos que han sido creados a partir de la misma definición de clase, en realidad tendremos un único objeto, pues hacen referencia a un mismo tipo de clase (plantilla). ¿Verdadero o falso?

- ○ Verdadero.
- ○ Falso.

2.4 Miembros estáticos o de clase.

Cada vez que se produce una instancia de una clase (es decir, se crea un objeto de esa clase), se desencadenan una serie de procesos (construcción del objeto) que dan lugar a la creación en memoria de un espacio físico que constituirá el objeto creado. De esta manera cada objeto tendrá sus propios miembros a imagen y semejanza de la plantilla propuesta por la clase.

Por otro lado, podrás encontrarte con ocasiones en las que determinados miembros de la clase (atributos o métodos) no tienen demasiado sentido como partes del objeto, sino más bien como partes de la clase en sí (partes de la plantilla, pero no de cada instancia de esa plantilla). Por ejemplo, si creamos una clase Coche y quisiéramos disponer de un atributo con el nombre de la clase (un atributo de tipo `String` con la cadena "Coche"), no tiene mucho sentido replicar ese atributo para todos los objetos de la clase Coche, pues para todos va a tener siempre el mismo valor (la cadena "Coche"). Es más, ese atributo puede tener sentido y existencia al margen de la existencia de cualquier objeto de tipo Coche. Podría no haberse creado ningún objeto de la clase Coche y sin embargo seguiría teniendo sentido poder acceder a ese atributo de nombre de la clase, pues se trata en efecto de un atributo de la propia clase más que de un atributo de cada objeto instancia de la clase.

Para poder definir miembros estáticos en Java se utiliza el modificador `static`. Los miembros (tanto atributos como métodos) declarados utilizando este modificador son conocidos como miembros estáticos o miembros de clase. A continuación vas a estudiar la creación y utilización de atributos y métodos. En cada caso verás cómo declarar y usar atributos estáticos y métodos estáticos.