UF 7.1
Objetos

JAVA™

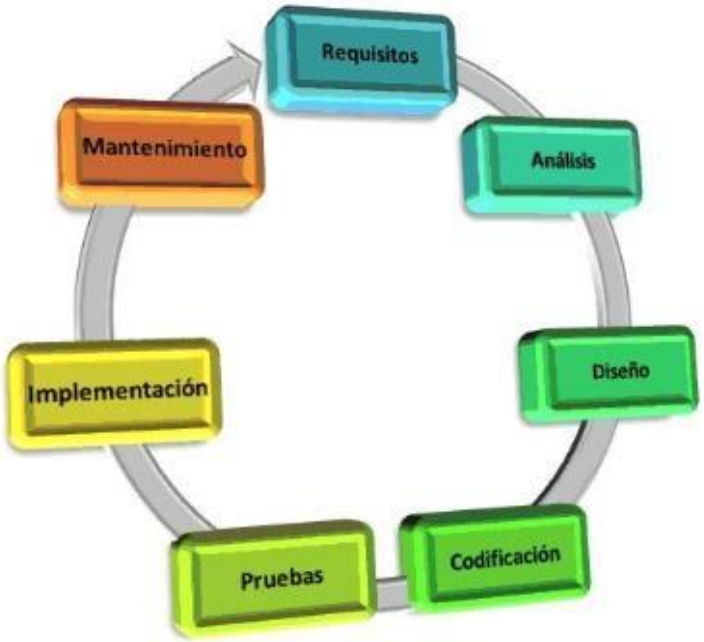
Centro Profesional
Universidad Europea Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

CONTENIDOS

1. Introducción
2. UML: Clase Coche

Lo primero

Cuando empecemos a programar en Java, el primer objetivo es construirnos el conjunto de clases adecuadas para resolver los requerimientos de nuestro programa.
Esta fase habitualmente se llama análisis

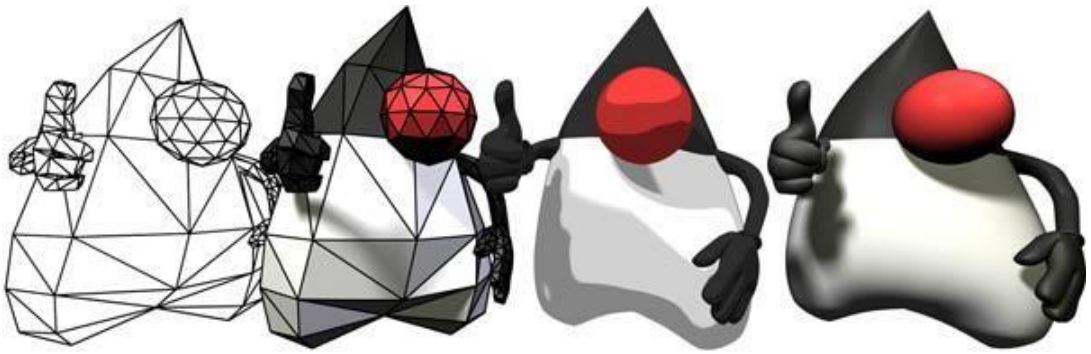


Fases o Etapas del Ciclo de Vida del Software

Una clase se compone de dos partes fundamentales: los atributos y los métodos.

Los atributos se definen por medio de variables, así que podemos considerarlas como variables del propio objeto.

Los métodos permiten añadir comportamiento a un objeto. Esta es la diferencia fundamental respecto a lo que conocéis como entidad en el *Modelo Entidad Relación* aquellos que habéis estudiado el módulo BB.DD (con algún otro profesor :-), puesto que la entidad era algo "estático".

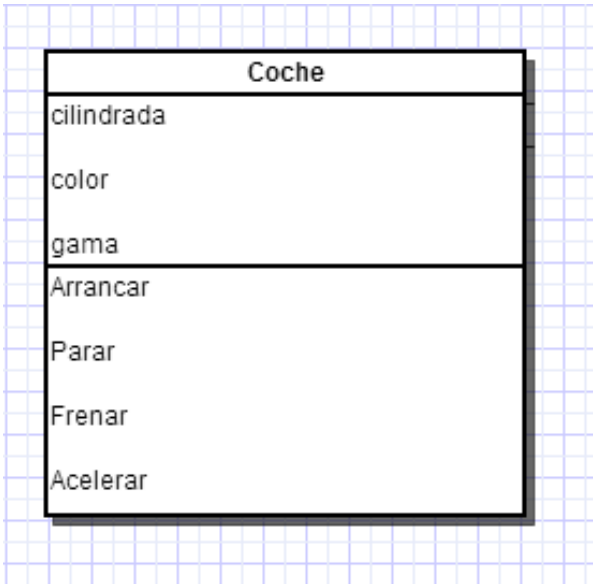


UML - EJEMPLO

Clase Coche

Un método nos permite interactuar con el objeto al que pertenece, o bien interactuar con otros objetos. Si continuamos con el ejemplo de la clase coche, podríamos tener los siguientes métodos

Vamos a implementar esta clase:



DEFINIR LA CLASE

Primer paso

La El primer paso es definir la clase, de la siguiente manera:

```
class Coche {

}
```

Prueba a hacerlo en eclipse…No parece tan difícil, ¿verdad?

DEFINIR LOS ATRIBUTOS

Segundo paso

Lo siguiente es especificar qué atributos necesitamos en nuestro coche.

Por ejemplo:

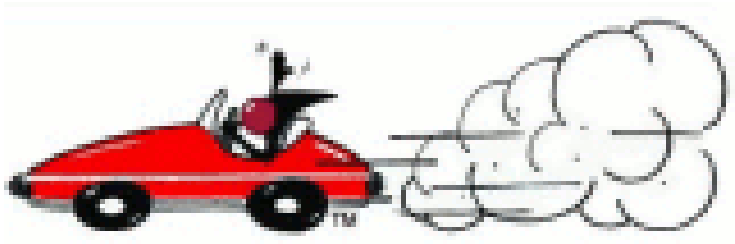
```
class Coche {
        Integer cilindrada;
        String color;
        Double precio;
        boolean isArrancado:
}
```

DEFINIR LOS ATRIBUTOS

Segundo paso

Bien, ya tenemos cuatro atributos:

•*cilindrada* de tipo Integer (fijaos que no he utilizado el tipo básico int, sino su envoltorio Integer);
•*color* de tipo String;
•*precio* de tipo Double (al igual que en el caso de cilindrada, no utilizo el tipo básico, sino su envoltorio, en este caso Double en lugar de double);
•*isArrancado* de tipo boolean. Esta última indica si el motor del coche está arrancado o no lo está

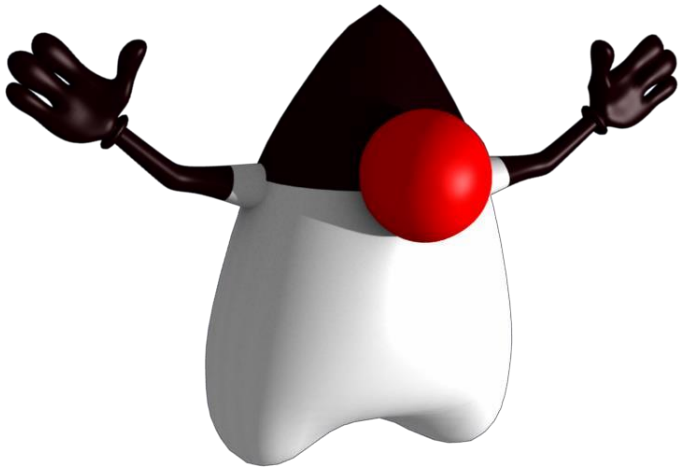


DEFINIR LOS MÉTODOS

Último paso

El tercer y último paso para construir nuestra clase es definir el comportamiento (métodos) de nuestro coche.
En concreto, únicamente vamos a tratar que el coche haya arrancado o esté parado.

Para ello, haremos lo siguiente:



```
void arrancar() {
        if (isArrancado != true) {
                isArrancado = true;
                System.out.println ("¡¡¡Brrrrrrr!!! El coche acaba de arrancar.");

        } else  {
                System.out.println ("¡¡¡Kisch, kishch!!! El coche ya está  arrancado.");
                }
        }
```

DEFINIR LOS MÉTODOS

Último paso

El método arrancar() comprueba si el coche está arrancado.

Si no lo está, lo arranca (con la línea isArrancado = true) y nos avisa con un mensaje. Si lo está, simplemente nos informa que el coche ya había arrancado.

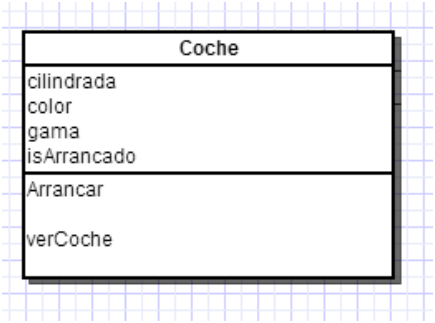
Fijaos que primero se definen los atributos y a continuación los métodos, esto es por convención de Java.

Este fichero debemos grabarlo con el nombre de la clase que hemos definido, seguido de la extensión .java. Es decir, debemos guardar el fichero con el nombre Coche.java.

AÑADIENDO FUNCIONALIDAD A LA CLASE

Implementación

Vamos a añadir un método para conseguir mayores prestaciones sobre la clase y para que veáis qué sencillo es ampliar el comportamiento de una clase. El nuevo método nos permitirá mostrar el valor de cada uno de los atributos.



Vamos a llamarlo verCoche(). El código es este:

```
void verCoche() {
        System.out.println("Este coche es de color " + color + " tiene una cilindrada de " +
        cilindrada + " y cuesta " + precio + " euros.");

        if (isArrancado == true) {
                System.out.println ("Ahora mismo está arrancado.");
        } else  {
                System.out.println ("Ahora mismo está parado.");
                }
}
```

Programa para trabajar con la clase

Ya podemos compilar el fichero.

Pero únicamente hemos definido una clase, con lo que no es posible ejecutar nada, ya que no hay nada que ejecutar.

Necesitamos un programa que nos permita trabajar con la clase coche que acabamos de crear. Este programa deberá crear objetos de la clase (recordad que “una clase” no es algo que exista en realidad, sino una plantilla de esa realidad, que es el objeto) y trabajar con ellos.

Veamos un ejemplo de programa y después lo comentaremos.

UTILIZACIÓN DE LA CLASE

Ejemplo

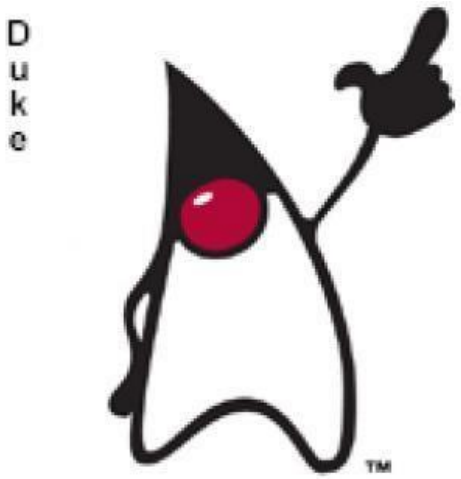
```
public static void main (String args[]) {
        Coche c;

        c = new Coche();
        c.cilindrada= 1800;
        c.color = “plata”;
        c.precio = 25000.0;

        System.out.println (“¿Qué coche es este?”);
        c.verCoche();

        System.out.println(“Vamos a arrancar el coche”);
        c.arrancar();

        System.out.println (“¿Qué coche es este?”);
        c.verCoche();

        System.out.println(“Vamos a arrancar el coche”);
        c.arrancar();
}
```

Explicación 1/3

Comentemos el programa anterior para entender cómo se trabaja en Java con los objetos.

En la primera línea, Coche c; , se declara un objeto de la clase Coche. A diferencia de C, la declaración no significa reserva en memoria. En el momento de la declaración, el objeto tiene el valor null. Para “crear” un objeto en realidad (y por tanto poder trabajar con él), se necesita utilizar el operador new.

La segunda línea c = new Coche();, crea un objeto de la clase Coche y lo asigna al objeto que se había declarado en la anterior línea.



Explicación 2/3

A partir de la tercera línea, se comienza a trabajar con el objeto recién creado. Con estas tres líneas:

```
c.cilindrada= 1800;
c.color = “plata”;
c.precio = 25000;
```

hemos modificado los atributos cilindrada, color y precio de nuestro objeto. Fijaos que el acceso a los atributos del objeto se realiza mediante el operador punto (.). Simplemente hemos de utilizar la siguiente sintaxis:

```
objeto.atributo
```

Explicación 3/3

A partir de ese momento, realizamos varias llamadas a los métodos. Por ejemplo:

System.out.println (“¿Qué coche es este?”);

c.verCoche();

Primero se muestra un mensaje de información y a continuación se accede al método verCoche() para comprobar si todo el código anterior de modificación de atributos ha funcionado correctamente. De la misma manera que ocurría para el acceso a los atributos, el acceso a los métodos utiliza la siguiente sintaxis:

objeto.método

DESTRUCCIÓN OBJETOS

Recordando

Respecto a la destrucción de objetos, no debemos preocuparnos en absoluto, ya que Java tiene el llamado garbage collector (recolector de basura) que se encarga de realizar toda la gestión de memoria por nosotros automáticamente.