APUNTES DE JAVA

INTRODUCCIÓN

- Basado en clases
- Object Oriented programming languaje (OOP)
- Java was released in 1995.
- Used for desktop apps, webs, mobile devices.
- Minecraft uses javascript (almost identical to java)
- Java is updated every once in a while, mainly security updates.
- Lenguaje ‘case-sensitive’. IMPORTANTE APRENDERSE TODAS LAS MAYUSCULAS "CocheRojo" y "cocherojo" no son lo mismo.

Objeto: bloque de codigo que tiene un nombre, atributos, metodos

PALABRAS CLAVE (*las tenia desperdigadas a medida que se iban nombrando en clase, pero las he juntado, son las del github*):

- Class: A basic building block.
- Static: Always in memory.
- Public: All the code can access it.
- Objeto: bloque de codigo que tiene un nombre, atributos, metodos
- Case sensitive: The program distinguishes lowercase letters from uppercase letters.
- PC: Personal computer.
- Interface: Through where a person communicates with a machine.
- GUI: Graphic user interface.
- CLI: Command line interface.
- Abstraction: "The process of generalizing concrete details,[1] such as attributes, away from the study of objects and systems to focus attention on details of greater importance."
- Refactorize: Make small changes in a program to make it work as you want.

- OOP: Object Oriented Programming.
- Final: When placed before a variable, it means it cannot be changed during execution, like a constant.
- Instance: "A specific realization of any object."
- UML diagram: Unified Modeling Language.
- Void: The return value is 0.

EJEMPLO DE CÓMO EMPIEZA UN CODIGO EN JAVA

```
public class Main {

  public static void main(String[] args) {

        System.out.println("Hello World");
```

Main: El primer *main* tiene que ser SÍ O SÍ el mismo que el nombre del archivo.java. (si mi archivo se llama *coche.java*, el *main* ese se substituye por *coche*.

Class: Java está basado en clases. Define objetos y tipos de datos y métodos.

Static: Siempre en memoria

Public: Todas las partes del código pueden acceder a esta clase. También podría poner private, y solo pueden acceder las partes del código que tienen permiso.

YA NO USAMOS *Serial.println()*

Usar: System.out.println()

- Fuciona de la misma manera que *Serial.println()* de C++

(per si fa falta, explicar en el github com funciona de nou)

!! Para hacer una variable constante en java se usa *final* delante de la variable.

!! Se declaran igual las variables que en C++. Se puede declarar una variable sin inicializar, y esto hacerlo más tarde en el código.

!! Comparadores funcionan de la misma manera.

(repasar el modul (%) per el día de l'examen (data al teams))

BUCLE WHILE

Ejecutan un codigo mientras (while en ingles, por eso el nombre) la condición dada sea **true**.

Ejemplo:

```
while (condición) {
  // codigo a ejecutar
}
```

Ejemplo:

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

Esto imprime:

0

1

2

3

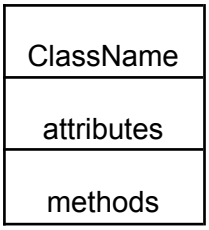
4

Esto ya se sabe pero es **importante:** no olvidarse de incementar la variable usada en la condición, ya que sino el bucle seguirá para siempre.

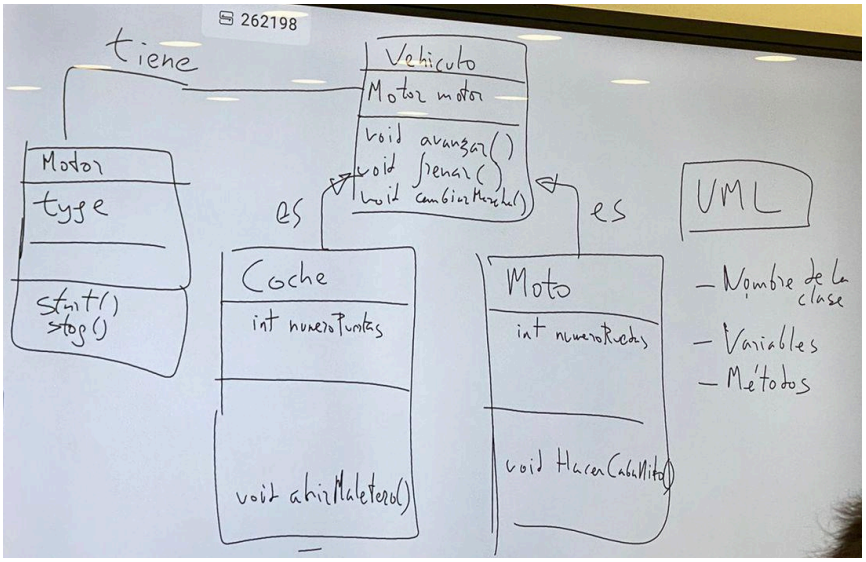
DIAGRAMA UML

- Para que sea más fácil visualizar un código complejo y la relación entre los distintos objetos.
- Muy visual
- Se tiene que marcar la relación entre objetos

Estructura:



Ejemplo de clase:



Vehículo **es** una clase padre (parent) de Coche y Moto.

Coche y Moto **son** vehiculos.

Coche y Moto extienden a Vehículo.

Vehículo **tiene** motor.

Coche y Moto, por trascendencia, también tienen Motor.

REPASO

DIFFERENCES BETWEEN PARAMETERS/ATTRIBUTE/ LOCAL VARIABLE

When we instance an object we initialize the variables

Parameters are what we send to a function

Local variable – a variable which only functions on a specific part of the code

WHY USING OOP?

OOP aims to modularity

Modularity in computing and programming refers to dividing a system into separate modules or components. Each module handles a specific functionality and operates independently. But these blocks are usable between them.

Example of modularity: LEGO