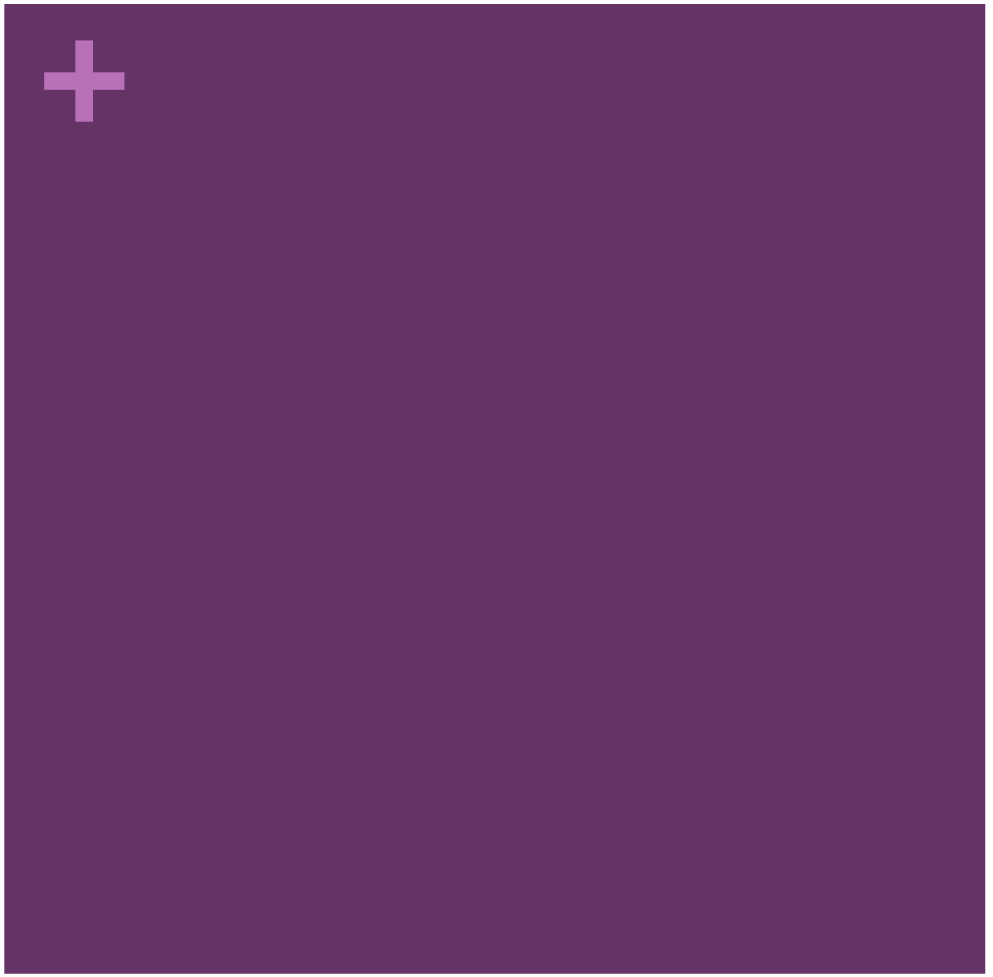




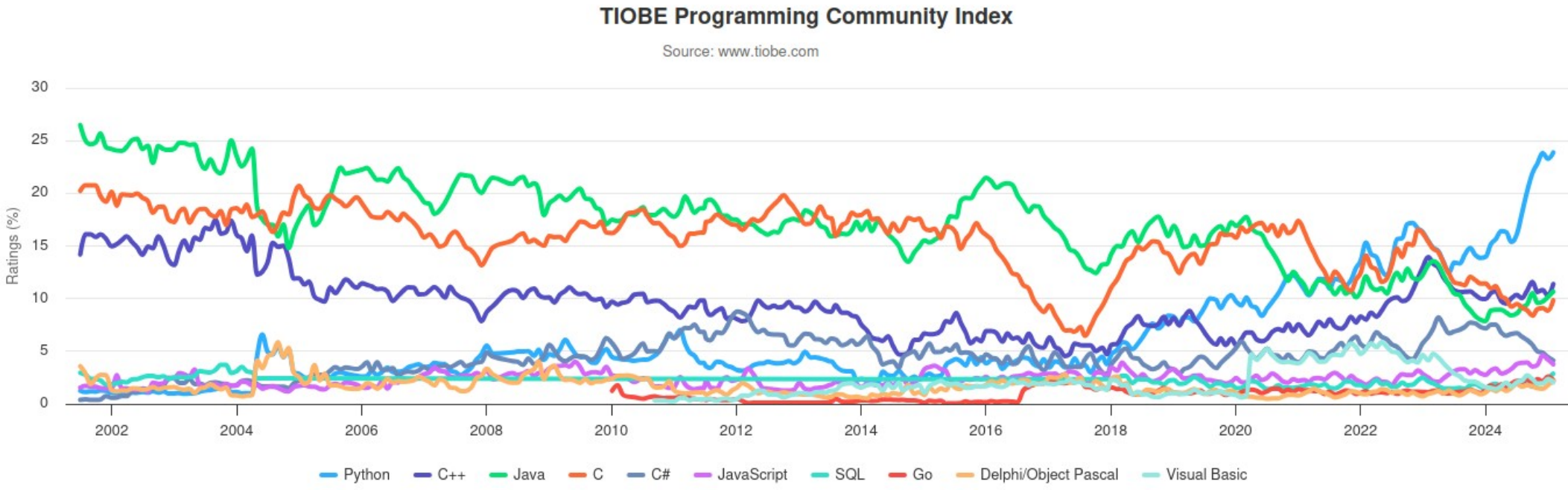
Java en 2 horas

Rodrigo Santamaría

Generalidades

- Desarrollado por  1995
- Hereda mucha de la sintaxis de **C** (1972)
- **Fuertemente tipado** y **orientado a objetos**
- Aplicaciones compiladas a **bytecode**
- Gestión interna de la reserva de **memoria**
- Desde 2007, el Java de Sun es **software libre**
  - Salvo las bibliotecas de clases para ejecutar programas java (OpenJDK)
  - IcedTea es una versión de la OpenJDK totalmente libre

Java: popularidad



Python vs Java

Comparison	Python	Java
Learning curve and readability	Easy	Not as easy
Syntax	Whitespace and indentions	Brackets and semicolons
Types	Dynamic	Static
Building and running	Executed by the Python Interpreter	Compiled, then run on the JVM
Performance	Slower than Java	Relatively fast
Community and popularity	Very popular	Very popular
Use cases	Machine learning, data processing	Android, enterprise development
Jobs and salaries	Well paying	Well paying

https://blog.udemy.com/python-vs-java/

Sobre servicios web en Python, tenéis este documento

Bytecode

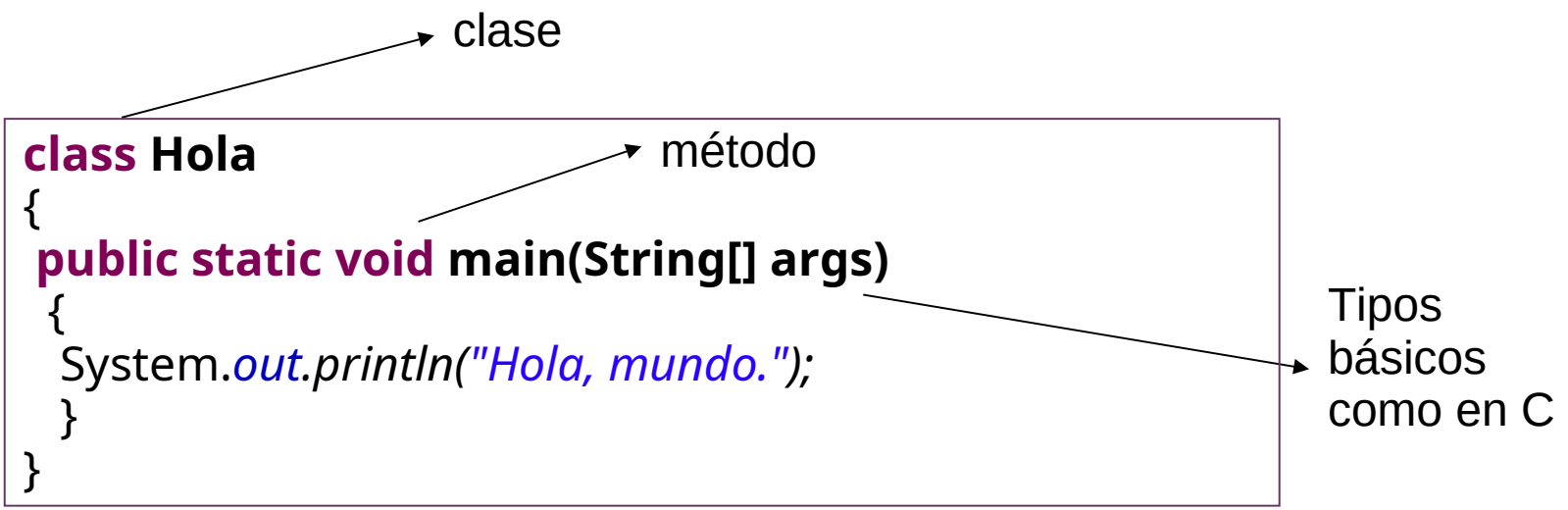
- Un programa Java no se compila a código de la máquina, si no a *bytecode*
  - **Código intermedio** entre el código máquina y el lenguaje de alto nivel que permite que un programa Java corra en distintas plataformas sin recompilar
  - Tan sólo es necesario tener instalada la **Máquina Virtual de Java (JVM)**, que se encarga de ejecutar las instrucciones del bytecode

JRE y JDK

- Java Runtime Environment (**JRE**)
  - Contiene la JVM en su versión más simple
  - Suele incluir las bibliotecas de clases estándar (API de Java, o J2SE, hoy Java Standard Edition o **JSE**)
- Java Development Kit (**JDK**)
  - JRE para desarrollar programas java
  - API de Java (código fuente abierto*)
  - Compilador de Java (`javac`), generador de documentación (`javadoc`), visor de applets, depurador, etc.

* https://en.wikipedia.org/wiki/OpenJDK#OpenJDK_builds

Hola Mundo




```
class Hola
{
 public static void main(String[] args)
  {
   System.out.println("Hola, mundo.");
   }
}
```

```
javac Hola.java
java Hola
```

Sintaxis básica

- Control de flujo
- Tipos básicos
- Arrays
- Operadores aritmético-lógicos

Idéntico a C

- Formato de cadenas: variables entre texto
  - `String cad="hola,"+nombre+", buenos días";`
- Palabras clave particulares: `package, class, public, private, extends, implements, this...`

Entornos de Desarrollo Integrado (IDE)

- Integran bibliotecas, compilador, intérprete, depurador, editor y otros complementos en un solo entorno:
  - Netbeans
  - **Eclipse**
  - IntelliJ IDEA
  - Anjuta
- En esta asignatura utilizaremos Eclipse
  - Es gratuito y con licencia de software libre (EPL)
  - Tiene una instalación limpia, disponible en el aula
  - Versiones para desarrollo web

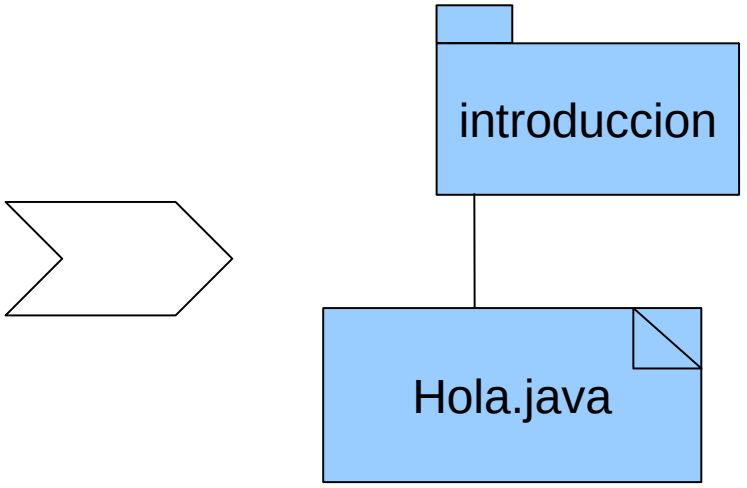
Paquetes y Clases

- **Paquete**: contenedor de clases (**carpeta**)
- **Clase**: contenedor de métodos y atributos (**fichero**)

Convención:
en minúscula

en mayúscula

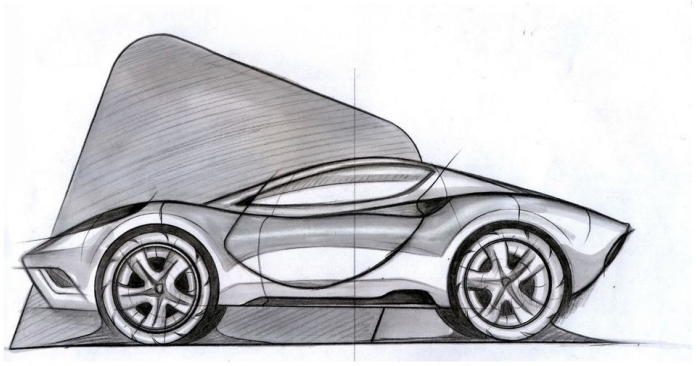
```
package introduccion

class Hola
 {
 ...
 }
```



- paquete2 dentro de paquete1: paquete1.paquete2

Clases y objetos

- **Clase**: diseño abstracto de un concepto
  - Define atributos (variables de la clase)
  - Define métodos (operaciones de la clase)
- **Objeto**: cada instancia concreta de la clase
  - Permite dar valor a los atributos
  - Permite ejecutar los métodos



Clases y Objetos

clase

```
public class Coche
 {
 float velocidad;
 int numPuertas;
 String color;
 public Coche(int numPuertas, float velocidad, String color)
                {
                this.velocidad=velocidad;
                this.numPuertas=numPuertas;
                this.color=color;
                }
        public int getNumPuertas()
                {
                return numPuertas;
                }
        public void acelerar(int nuevaVelocidad)
                {
               if(nuevaVelocidad>velocidad) velocidad=nuevaVelocidad;
                }
 }
```

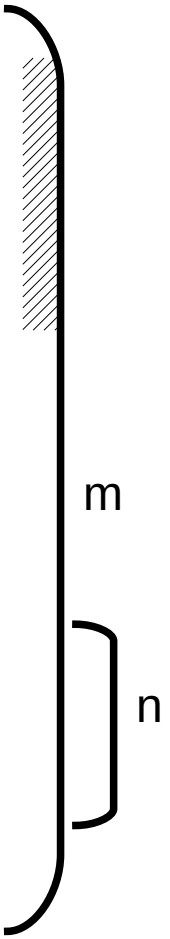
objeto

```
import Coche

public class Main
        {
        public static void main(String [] args)
                {
                Coche miFerrari=new Coche(3,0, ”rojo”);
                System.out.println(” Mi coche tiene "
                                +miFerrari.getNumPuertas()+” puertas”);
                miFerrari.acelerar(25);
                }
        }
```

Variables y atributos

Alcance

```
public class Ejemplo
{
int m; //atributo

public static void main(String args[])
    {
    if(args.length==0)
        System.err.println("Error");
    //...
    }
public int cuadrado()
    {
    return m*m;
    }

public int operacion()
    {
    int n=33; //variable
    n=m*n;
    //...
    return n;
    }
//...
}
```



Un atributo puede usarse en toda la clase, salvo en métodos estáticos*.

Una variable sólo en el espacio desde su declaración hasta el final del método.

**¿Por qué no en métodos estáticos?*

Privacidad

- **public**: accesible desde fuera y dentro de la clase
- **private**: accesible sólo desde la clase
- **protected** (por defecto) accesible desde dentro de la clase o de clases hijas

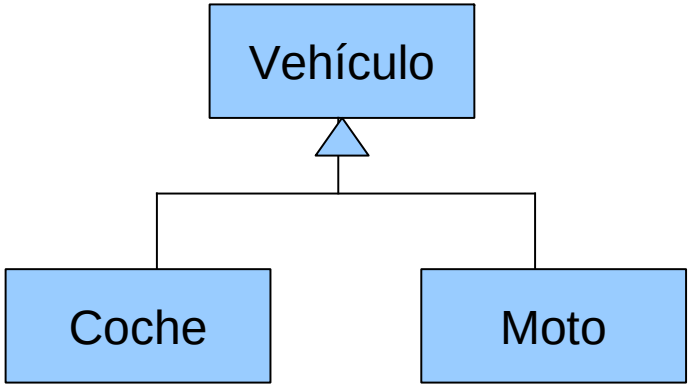
- Atendiendo a la **modularidad**, es recomendable que los atributos de una clase sean privados o protegidos.
- Atendiendo a la **claridad** de código y **eficiencia** de recursos, los atributos públicos pueden ser útiles.

Herencia

```
class Vehiculo
    {
    float velocidad;
    int numPuertas;
    String color;
    int numRuedas;

    public int getNumPuertas()
        {
        return numPuertas;
        }
    public void acelerar(int nuevaVelocidad)
        {
        if(nuevaVelocidad>velocidad)
            velocidad=nuevaVelocidad;
        }

    public Vehiculo()
        {
        velocidad=0;
        numPuertas=1;
        color="negro";
        numRuedas=2;
        }
    public Vehiculo(int numPuertas, int velocidad, String color)
        {
        this.velocidad=velocidad;
        this.numPuertas=numPuertas;
        this.color=color;
        this.numRuedas=4;
        }
    }
```

```
class Coche extends Vehiculo
    {
    public Coche(int numPuertas, int velocidad, String color)
        {
        this.velocidad=velocidad;
        this.numPuertas=numPuertas;
        this.color=color;
        this.numRuedas=4;
        }
    }

class Moto extends Vehiculo
    {
    public Moto(int numPuertas, int velocidad, String color)
        {
        super(numPuertas, velocidad,color);
        this.numRuedas=2;
        }
    }
```

Constructor por defecto ~ **super**()



Polimorfismo

```
public class Coche extends Vehiculo implements Cloneable
{
public Coche(int numPuertas, int velocidad, String color)
 {
 this.velocidad=velocidad;
 this.numPuertas=numPuertas;
 this.color=color;
 this.numRuedas=4;
 }
public Coche(String color)
 {
 this.velocidad=0;
 this.numPuertas=3;
 this.color=color;
 this.numRuedas=4;
 }
public Coche()
 {
 this.velocidad=0;
 this.numPuertas=3;
 this.color="negro";
 this.numRuedas=4;
 }
}
```

Varios métodos con el mismo nombre, siempre que los parámetros sean distintos

Métodos estáticos

```
public class Coche extends Vehiculo
{
public Coche(int numPuertas, int velocidad, String color)
 {
 this.velocidad=velocidad;
 this.numPuertas=numPuertas;
 this.color=color;
 this.numRuedas=4;
 }

...

public static String getAutorDeLaClase()
 {
 return "Rodrigo Santamaría";
 }

}
```


En un método estático no podemos hacer uso de atributos o métodos no estáticos

```
public class Main {
public static void main(String [] args)
 {
 System.out.println("Autor de la clase Coche: "
     +Coche.getAutorDeLaClase());
 ...
 }
```

Gestión de la memoria

- Reserva de memoria explícita (**new**)
- Liberación automática (**garbage collector**)
  - Detecta objetos no utilizados/accesibles y los borra
  - Puede reducir el rendimiento
  - Solicitud explícita con `System.gc();`
- Asignación de objetos **por referencia**
  - Para asignación por valor: `coche1=coche2.clone();`
    - Sólo si Coche implementa `Cloneable`
    - Y redefine el método `clone()`

```
public Object clone()
        {
        Coche obj=new Coche();
        obj.numPuertas=this.numPuertas;
        obj.velocidad=this.velocidad;
        obj.color=this.color;
        return obj;
        }
```

Control de Errores

```
public void acelerar(int nuevaVelocidad) throws Exception
    {
    if(nuevaVelocidad>velocidad)
        velocidad=nuevaVelocidad;
    else
        throw new Exception("no se puede acelerar a "+nuevaVelocidad);
    }
```

El método indica qué excepciones pueden ocurrirle

Y cuando ocurren, las lanza

```
public static void main(String[] args)
    {
    Coche miFerrari=new Coche(3,0, "rojo");
    System.out.println("Mi coche tiene"+miFerrari.getNumPuertas()+" puertas");
    try{
        miFerrari.acelerar(100);
    }catch(Exception e)
        {
        System.err.println("Error al acelerar: "+e.getMessage());
        e.printStackTrace();
        }
    }
```

Al usar métodos que pueden lanzar excepciones, deben estar en una sección **try/catch**, de forma que si se producen, las detectemos y gestionemos correctamente

Diccionarios

- Estructura de datos organizada como conjunto de pares <clave,valor>
- En Java, varias versiones que heredan de la interfaz **Map**
  - En particular para la asignatura recomendamos trabajar con **ConcurrentHashMap**
    - **Hash**: las claves se almacenan internamente con códigos hash para una búsqueda más eficiente
    - **Concurrent**: a prueba de modificaciones concurrentes (funciona como una sección crítica)

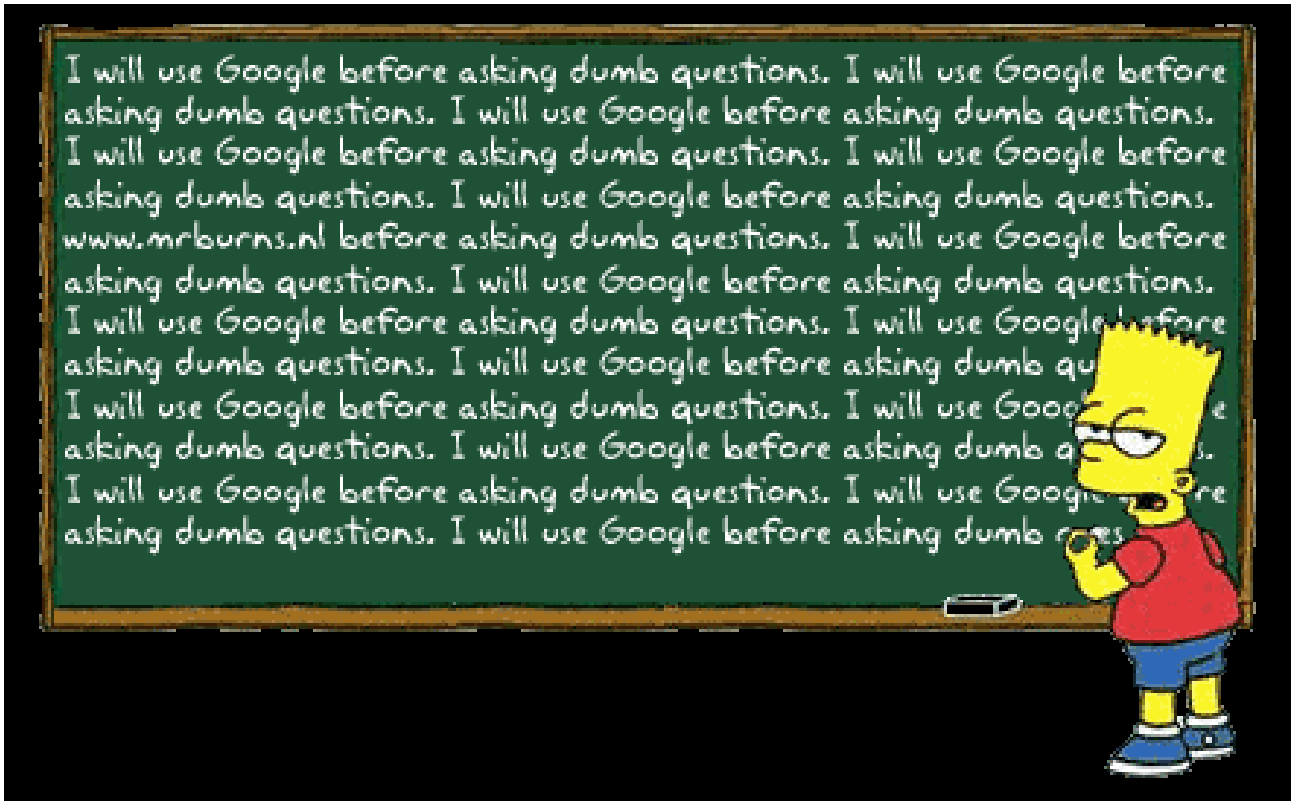
```
HashMap<String, Coche> garage=new HashMap<String,Coche>();

...
garage.get("2008 ZGZ"); //'Recojo' el coche con esta matrícula
garage.put("4315 BHH, miCoche); //'Aparco' mi coche
```

Concluyendo…

- Organización en:
  - paquetes, clases y objetos
  - atributos y métodos
- Programación
  - Orientada a objetos: modularidad, herencia, polimorfismo
  - Gestión sencilla de memoria, errores, I/O
- Multiplataforma (bytecode + máquina virtual)
- Sofware “libre”
  - API oficial
  - Multitud de paquetes desarrollados

Documentación

- Documentación:
  - https://docs.oracle.com/en/java/javase/11/docs/api/index.html
- Paquetes en la red para todo tipo de propósitos



Ejercicio

- Definir una clase *Fruta* y las clases hijas *Naranja* y *Manzana*.
  - Deben tener los atributos genéricos
    - *peso*, en kilos (atributo real)
    - *hechaZumo*, falso por defecto (atributo lógico)
    - La manzana debe tener también el atributo entero *numPepitas*
  - Deben tener los métodos siguientes
    - Constructor por defecto y constructor que permita al usuario iniciar todos los atributos
    - Un método *hacerZumo*() que modifique el estado de la fruta si *hechaZumo* es falso.

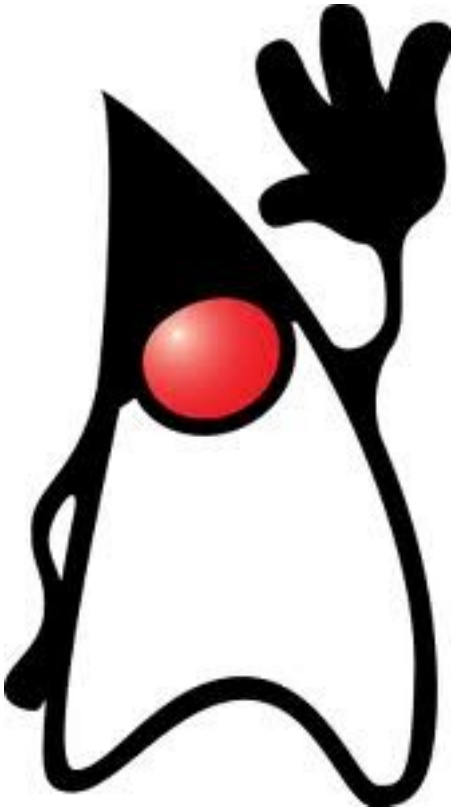
Ejercicio

- Definir una clase *Principal* que:
  - Instancie dos frutas, una manzana y una naranja
  - Nos diga el peso de la manzana
  - Haga zumo de naranja
- Tratamiento de errores
  - El método *hacerZumo* debe lanzar una excepción si se intenta hacer zumo de una fruta ya exprimida
  - La clase *Principal* debe capturar la excepción y avisar al usuario

Ejercicio

cont.

- Clasificación
  - Define un objeto *nevera* de clase ConcurrentHashMap para guardar las frutas.
  - Mete en la nevera la manzana y la naranja, con las claves “manzanaVerde” y “naranjaDeZumo”



Duke, la mascota
de Java