UD1.- Introducción a la programación

Módulo: Programación
1.º DAM

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

¿Qué es un programa?

- **Conjunto de instrucciones** que se le proporcionan a un ordenador en un lenguaje que él puede entender, para decirle exactamente el que queremos que haga.
- Conjunto de instrucciones que realizan una tarea determinada
- Si el ordenador no entiende alguna instrucción, generalmente lo comunicará mediante mensajes en la pantalla.

Tipo de lenguajes

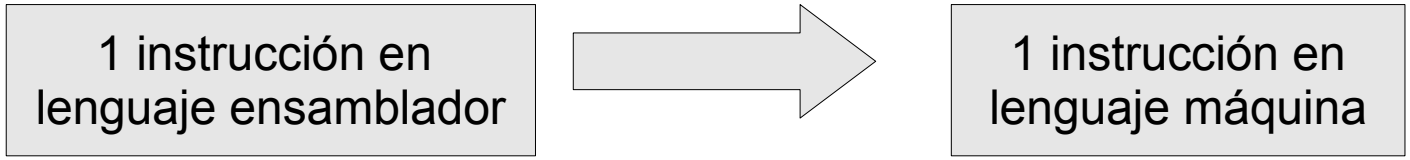
- **Lenguajes de programación**
  - Son lenguajes “artificiales” diseñados para especificar las órdenes que tiene que seguir el ordenador.
  - Por qué no podemos utilizar el lenguaje natural con los ordenadores?
    - Complejos y ambiguos.
    - No permiten definir acciones con claridad y precisión

Tipo de lenguajes

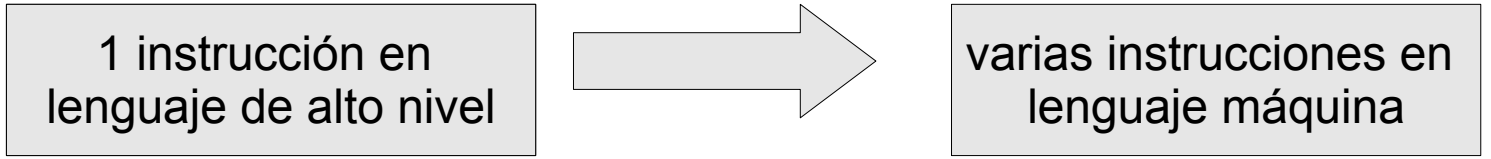
- **Lenguaje máquina**: es el lenguaje que el ordenador entiende directamente. Características:
  - Las instrucciones se expresan en binario.
  - Los datos se referencian mediante la dirección de memoria donde se encuentran. No aparecen nombres.
  - No pueden incluirse comentarios.
  - Depende del procesador del ordenador. Un programa escrito para un ordenador no se puede ejecutar en otro (baja portabilidad).

Tipo de lenguajes

- **Ensamblador**
  - Es el primer intento de sustituir el lenguaje máquina por un más próximo al nuestro.



- **De alto nivel**
  - Independientes de la arquitectura del ordenador.



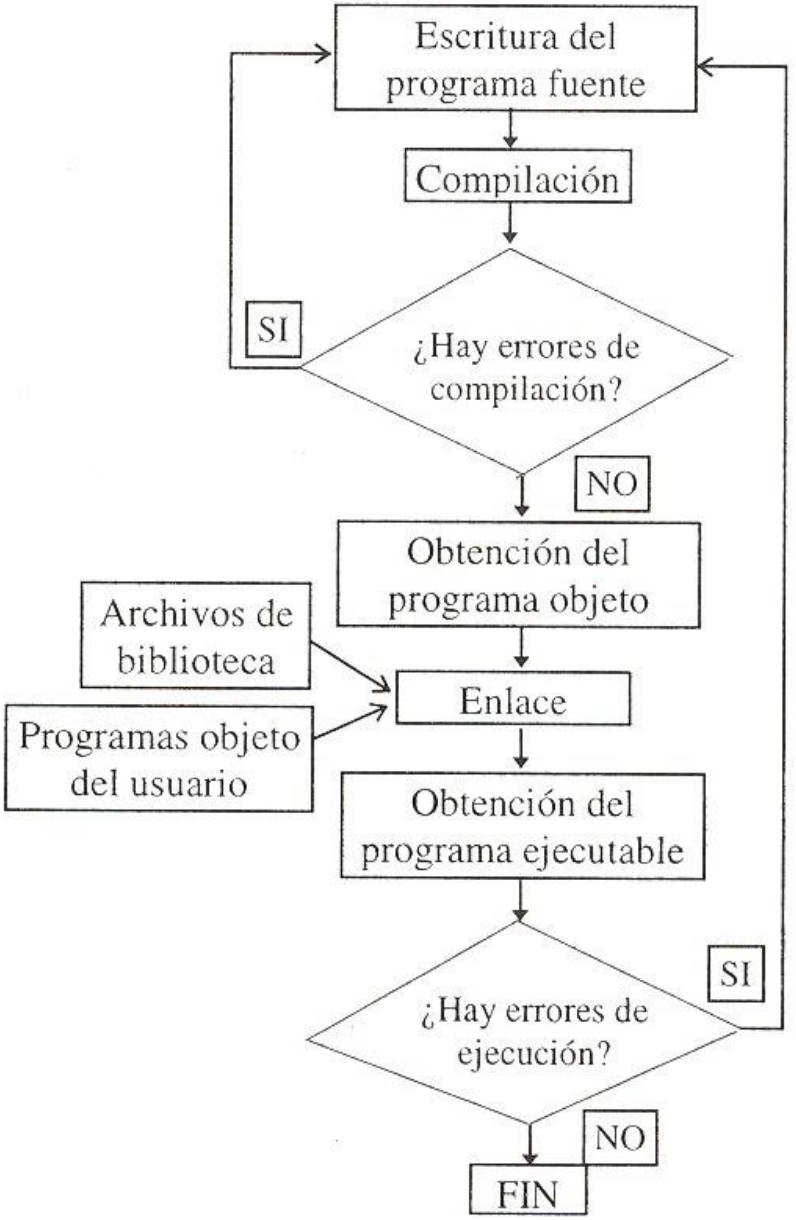
Traductores

- **Compiladores**
  - Traducen completamente uno programa fuente (aquel que está escrito en un lenguaje de alto nivel) y generan un programa objeto (semánticamente equivaliendo) escrito en lenguaje de bajo nivel.
  - El compilador informa al usuario de los posibles errores en lo programa fuente. Lo programa objeto solo se creará si no hay errores en el código fuente.
  - El programa objeto se almacenará en un fichero que se podrá ejecutar sin necesidad de volverlo a traducir (excepto si se han hecho cambios).

Traductores

- **Intérpretes**
  - Permiten que uno programa fuente vaya traduciéndose y ejecutándose directamente sentencia a sentencia por el ordenador.
  - Capta una sentencia del código fuente, la analiza y la interpreta y lo ejecuta directamente. No se crea ningún fichero objeto.

Proceso de compilación



Ciclo de vida

- Es el proceso que se sigue desde el planteamiento de un problema hasta que se tiene una solución instalada en el ordenador y en funcionamiento por parte del usuario, mientras sea de utilidad.

Ciclo de vida

- Se compone de varias fases agrupadas en dos bloques:
  - Fases de diseño.
    - Análisis (especificaciones, E/S)
    - Diseño (algoritmo)
    - Implementación (programa)
  - Fases **de instalación**
    - Edición (programa fuente)
    - Compilación (programa objeto)
    - Montaje (programa ejecutable)
    - Prueba de ejecución (aplicación)
    - Explotación y mantenimiento

¿Qué es Java?

Java es un lenguaje de programación de propósito general y por tanto es válido para desarrollar cualquier tipo de aplicaciones profesionales.

Los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura.

- Se ejecutan en una gran variedad de dispositivos con diferentes microprocesadores y sistemas operativos.

Características de Java (I)

Es intrínsecamente orientado a objetos (OO).

Funciona perfectamente en red

Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes.

Tiene una gran funcionalidad gracias a sus librerías (clases).

NO tiene punteros que tenga que manejar el programador, aunque los maneja internamente y transparentemente.

Es Java compilado o interpretado?

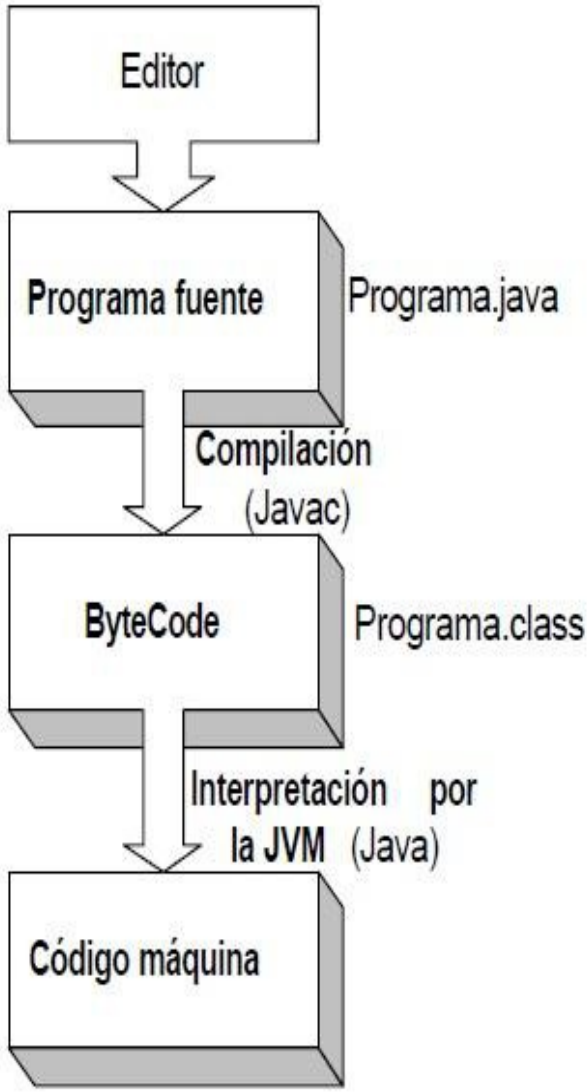
Aunque estrictamente hablando es interpretado, necesita un proceso previo de compilación.

Una vez “compilado” el programa, se crea un fichero que almacena *bytecodes* (pseudocodigo básicamente al nivel de lenguaje máquina).

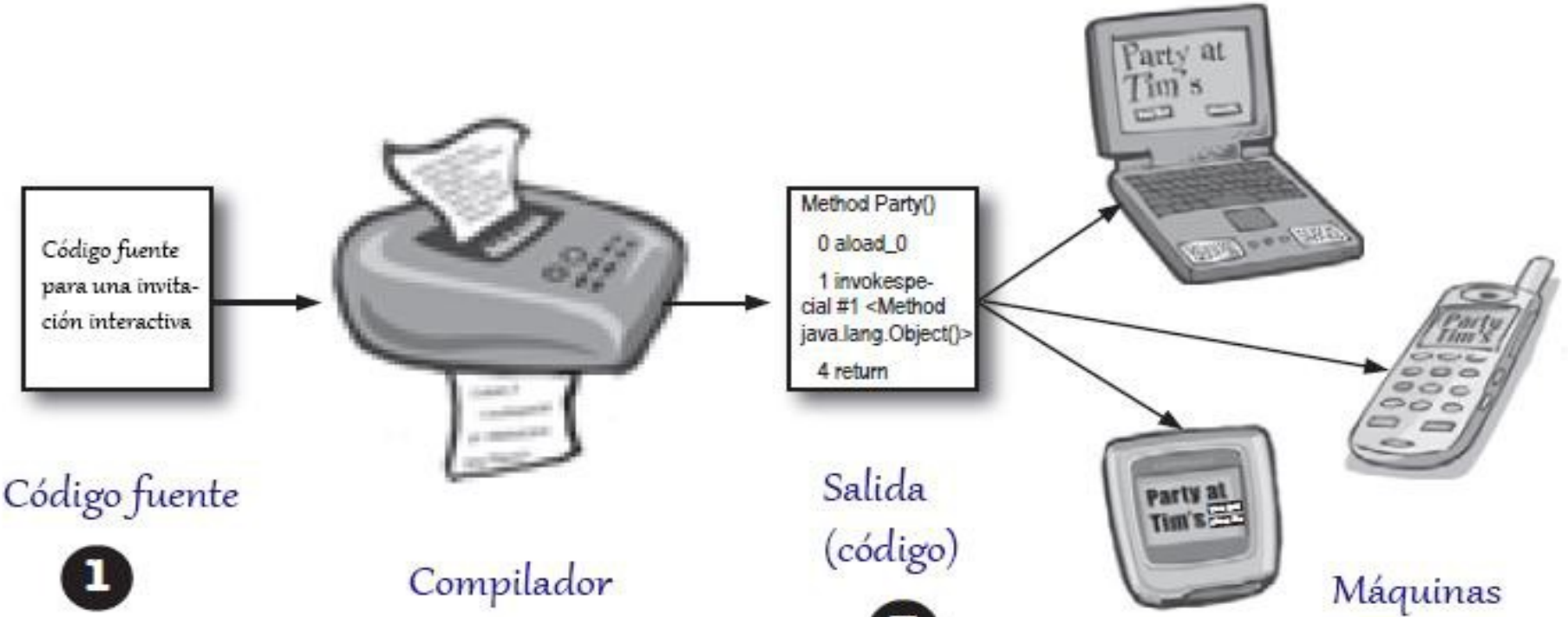
Para ejecutarlo, es necesario un “intérprete”, la JVM (*Java Virtual Machine*, Máquina Virtual de Java).

Es posible compilar el programa en una máquina con Linux y ejecutarla en otra con Windows utiltizando la máquina virtual Java para Windows.

La JVM se encarga de leer los bytecodes y traducirlos a instrucciones ejecutables directamente en un determinado procesador, de forma eficiente.



¿Cómo trabaja Java?



Código fuente

1

Se crea un documento en código fuente, en nuestro caso, en **Java**

Compilador

2

Se compila el documento con un compilador de código fuente que encuentra si hay errores, y no deja que el código se compile hasta que no se resuelvan los errores

Salida (código)

3

El compilador crea un documento de salida compilado (**bytecode** de Java)

Máquinas virtuales

4

Cualquier máquina virtual de Java (**JVM**) será capaz de entender y ejecutar dicho documento de salida

¿Cómo trabaja Java?

```
import java.awt.*;
import java.awt.event.*;
class Party {
  public void buildInvite() {
    Frame f = new Frame();
    Label l = new Label("Party at Tim's");
    Button b = new Button("You bet");
    Button c = new Button("Shoot me");
    Panel p = new Panel();
    p.add(l);
  } // more code here...
}
```

Código fuente

1

Escribe tu código fuente
en un editor de texto plano
Grábalo como: ***Party.java***

```
File Edit Window Help Plead
%javac Party.java
```

Compilador

2

Entrando en la línea de comandos
El compilador *javac* compila el archivo ***Party.java.***
Si no encuentra errores, generará un documento nuevo llamado ***Party.class,*** que será el código de salida.

```
Method Party()
  0 aload_0
  1 invokespecial #1 <Method java.lang.Object()>
  4 return
Method void buildInvite()
  0 new #2 <Class java.awt.Frame>
  3 dup
  4 invokespecial #3 <Method java.awt.Frame()>
```

Código de salida

3

Código compilado: ***Party.class***

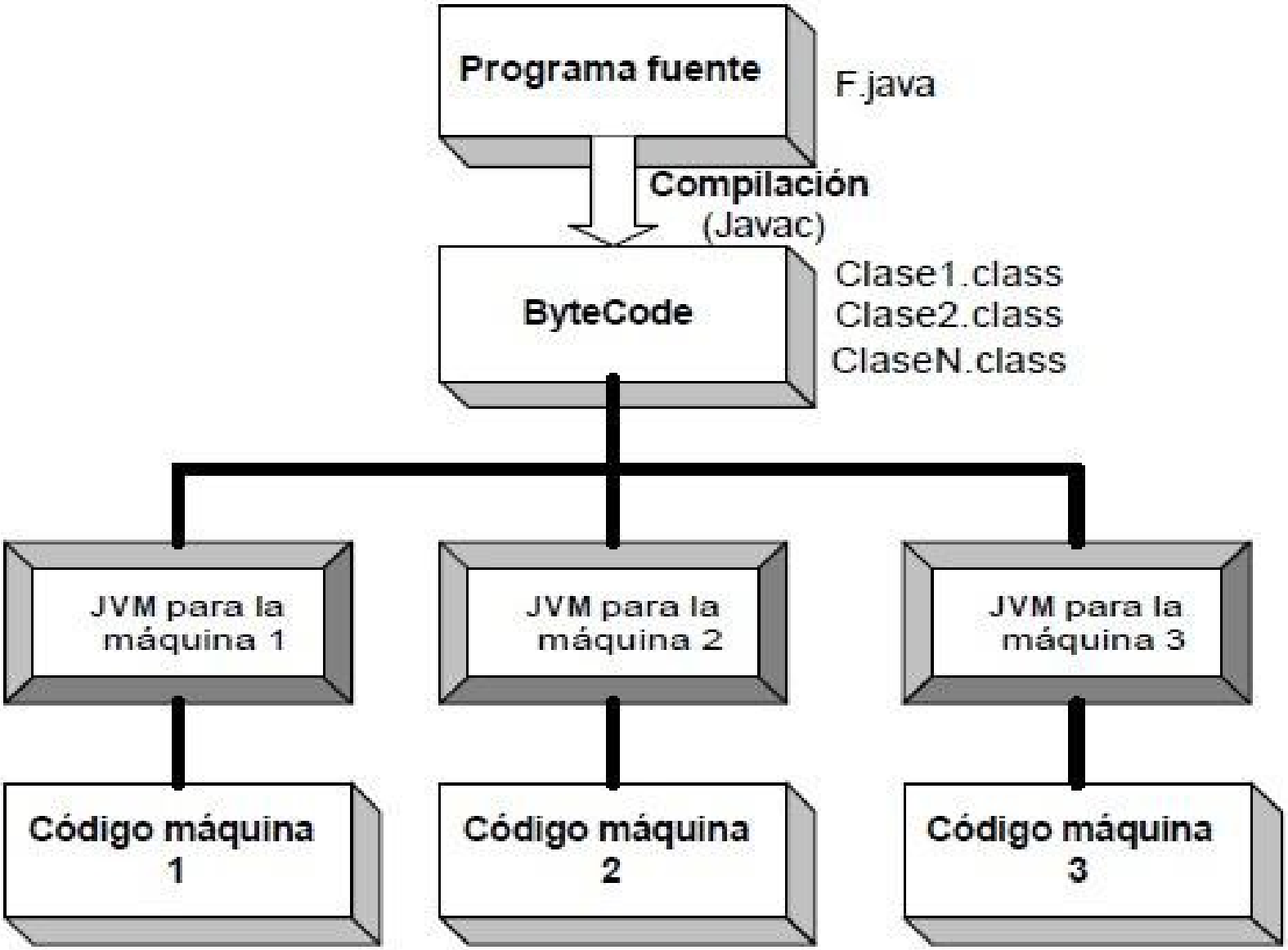
```
File Edit Window Help Swear
%java Party
```

Máquinas virtuales

4

La máquina virtual de Java (***JVM***) ejecuta el programa desde el fichero ***Party.class***

La Máquina Virtual Java (JVM) (I)

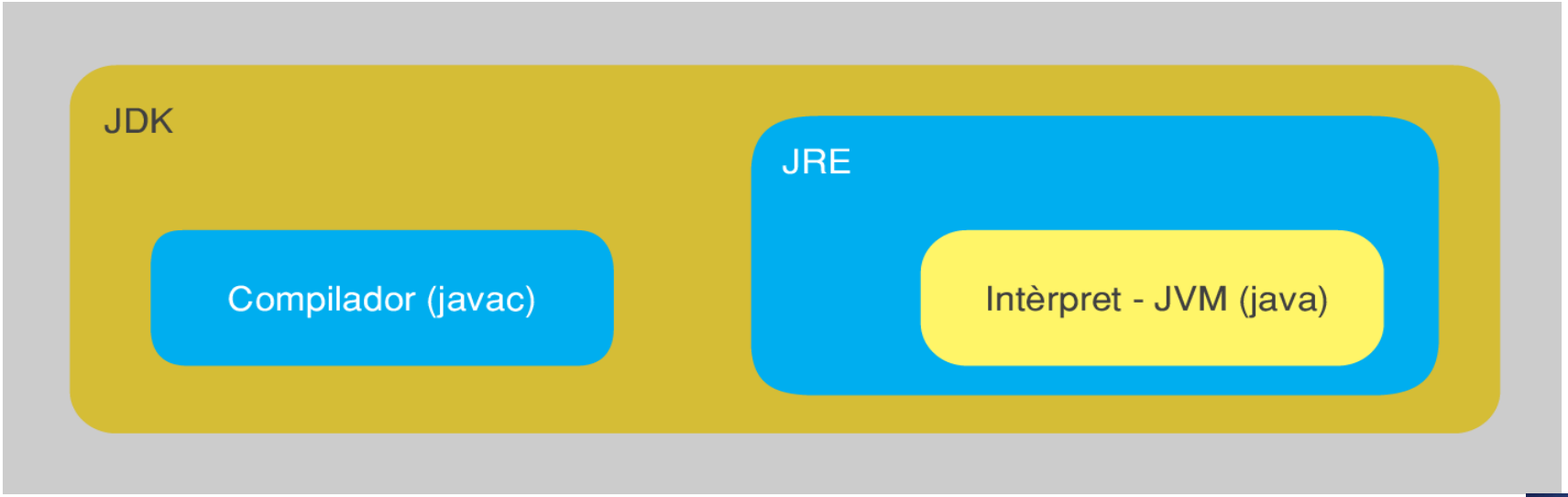


La Máquina Virtual Java (JVM) (II)

- Un mismo programa fuente compilado en distintas plataformas o sistemas operativos, genera el mismo fichero en bytecode.
- Hace la traducción del bytecode a código nativo de la máquina sobre la cual se ejecuta.
- Existe una versión diferente de la JVM para cada plataforma. La JVM se carga en memoria y va traduciendo “al vuelo” bytecodes a código máquina.
- No ocupa mucho de espacio en memoria.

El entorno de desarrollo JDK (I)

- La herramienta básica para empezar a desarrollar aplicaciones o applets en Java es lo JDK (*Java Developer's Kit,*Kit de Desarrollo Java.
- Consiste básicamente en un compilador y un intérprete (JVM) para la línea de mandos.



El entorno de desarrollo JDK(II)

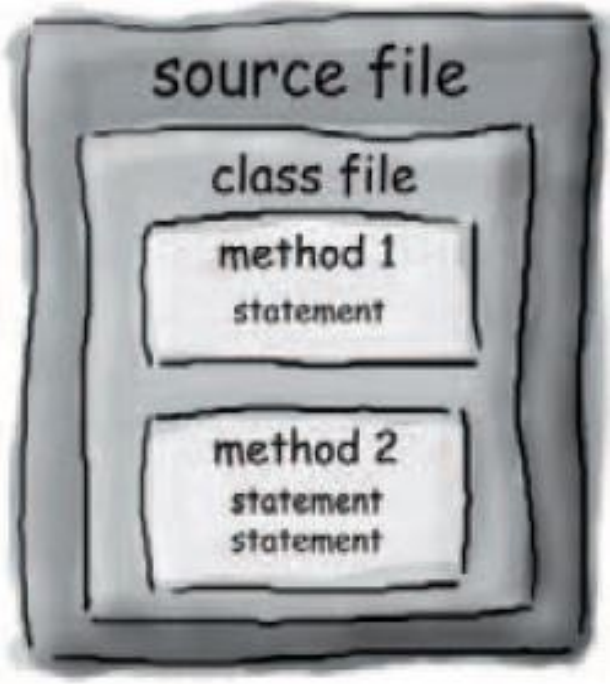
- El kit contiene básicamente:
  - El **compilador**: **javac.exe**
  - El depurador: jdb.exe
  - El **intérprete**: **java.exe** y javaw.exe
  - El visualizador de applets: appletviewer.exe
  - El generador de documentación: javadoc.exe
  - Un desensamblador de clases: javap.exe
  - El generador de archivos fuente y de cabecera (.c y .h) para clases nativas en C: javah.exe

En torno a Desarrollo Integrado

- Un IDE (*Integrated Development Environment*, En torno a Desarrollo Integrado) es una herramienta que integra todo aquello necesario para generar aplicaciones de forma que el trabajo sea más cómodo.

- Ejemplos de IDE:
  - IntelliJ IDEA CE, NetBeans, Eclipso y JCreator para Java
  - VisualStudio para C#, C++ y Visual Basic

Estructura de código en Java



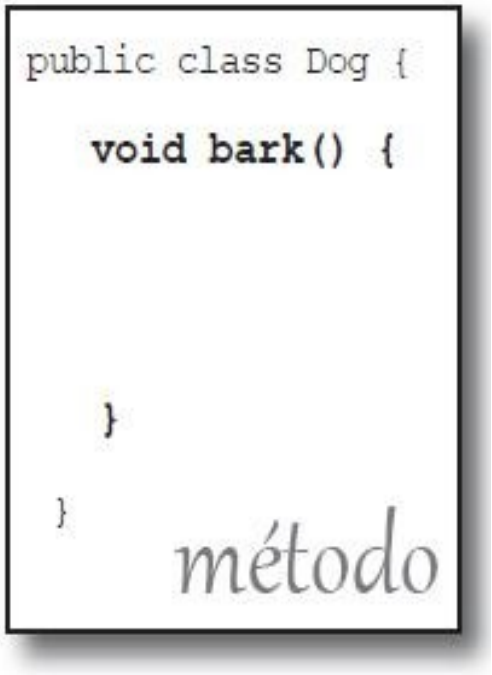
- **Source hilo**: fichero fuente
- **Class hilo**: fichero de clase
- **Method**: método
- **Statements**: sentencias

- Crea clases en un fichero fuente.
- Crea métodos en una clase.
- Crea sentencias en un método.

¿Qué contiene un fichero fuente?

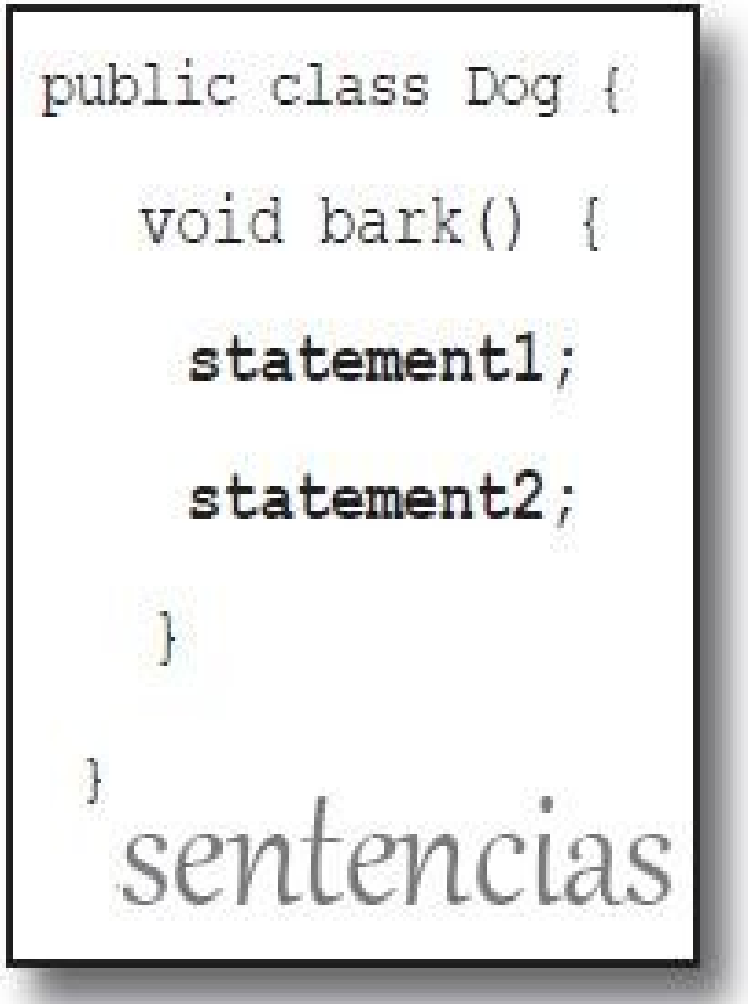
- Contiene, al menos, la definición de una clase. El contenido tiene que ir entre { y .. }
- Extensión **.java**.
- En el caso concreto de Java hay una convención para dar nombres a los ficheros:
  - UpperCamelCase
    - inicial de cada palabra en mayúscula (preferentemente sin acentos ni espacios)
    - No es imprescindible, pero sí recomendable
  - Ejemplos: HolaMundo.java, Prueba.java, ElMeuPrograma.java,...

¿Qué contiene una clase?



- Contiene uno o más métodos.
- Para declarar una clase: class NomClasse. El contenido irá entre { y .. }
- El nombre de la clase tiene que coincidir exactamente con el nombre del fichero que contiene el código. El nombre del fichero tendrá la extensión .java.
- MUY IMPORTANTE!!! Java es sensible a mayúsculas (*caso sensitive*). Holamon, HolaMon y holaMon son diferentes.

¿Qué contiene un método?



- Conjunto de sentencias que realizan una acción.
- Escribiremos instrucciones para especificar como tiene que funcionar cada método.

Ejecución de programas

- Cuando JVM arranca (al ejecutar), busca la clase creada y concretamente busca el método main()

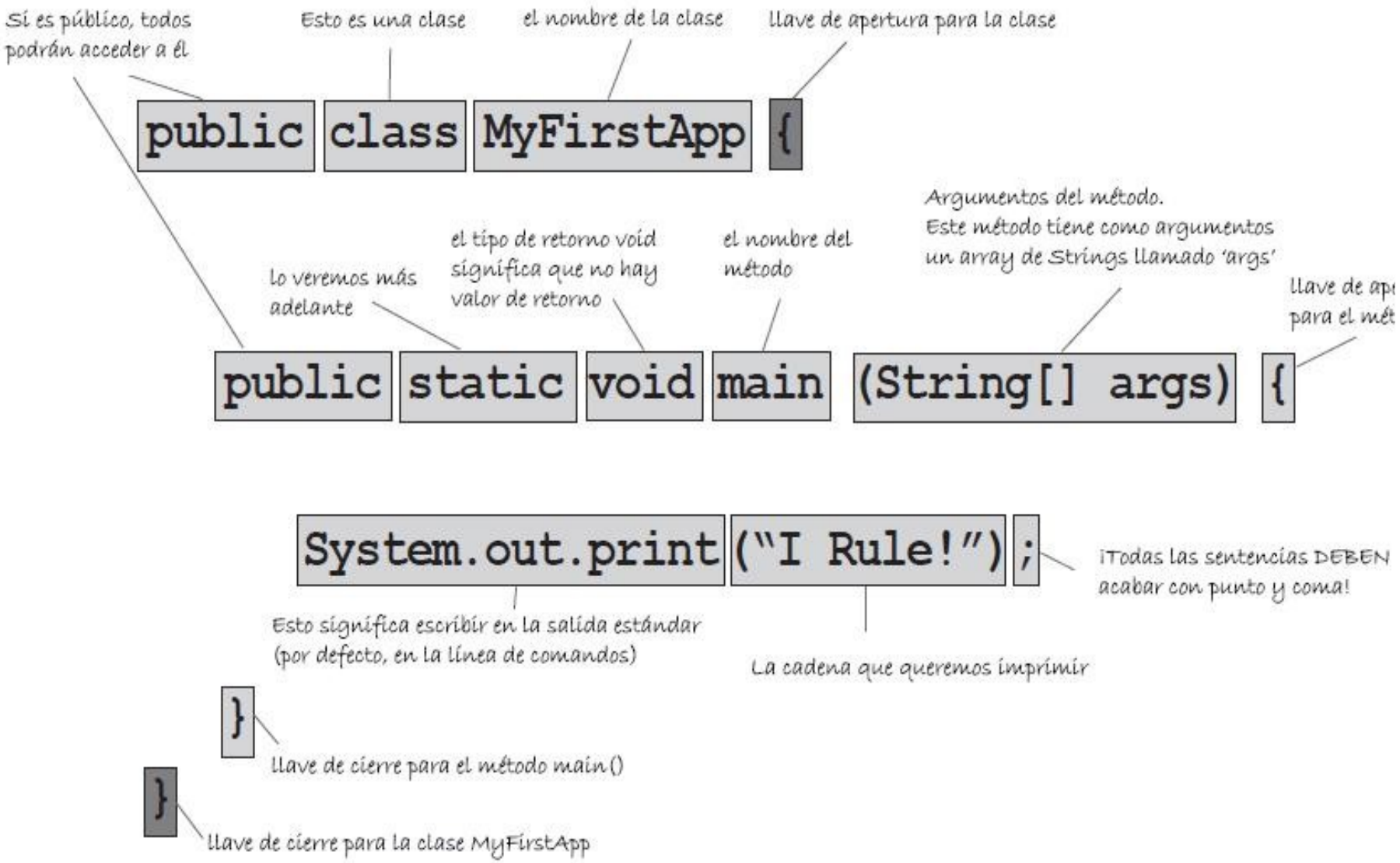
```
public static void main(String args[ ]){

        //el código va aquí


}
```

- JVM ejecuta todo el que encuentro entre { y .. }
- Cada aplicación Java tiene, al menos, una clase y cada clase, un método.
- Correcto: un main() por aplicación
- Incorrecto: un main() por clase

Anatomía de una clase



Compilar y ejecutar

Empezaremos con el compilador javac (consola de comandos, “símbolo del sistema”)

Instalar el JDK y configurar las variables de entorno

Compilación: javac nomClasse.java

Ejecución: java nomClasse

Pinceladas sobre Java

- Cada sentencia tiene que acabar en punto y com

```
x = x+1;
```

- Un comentario empieza con doble barra.

```
x = 22;
//esta línea es un comentario
```

- Más de un espacio no es un problema.

```
x  =  3;
```

Pinceladas sobre Java

- Las variables se declaran con nombre y tipo.

```
int pes;
```

tipo nombre

- Las clases y métodos tienen que definirse entre { }.

```
public void nadar() {
   //código para nadar
}
```

Pinceladas sobre Java

- Sentencias

1 Hacer algo

Sentencias: declaraciones, asignaciones, llamadas a métodos, etc.

```
int x = 3;
String name = "Dirk";
x = x * 17;
System.out.print("x is " + x);
double d = Math.random();
// esto es un comentario
```

Pinceladas sobre Java

- Bucles

2 Hacer algo una y otra vez

Bucles: *for* y *while*

```
while (x > 12) {
   x = x -1;
}

for (int x = 0; x < 10; x = x + 1) {
   System.out.print("x is now " + x);
}
```

Pinceladas sobre Java

- Bifurcaciones

3 Hacer algo bajo una condición

Ramificaciones: condiciones ***if/else***

```
if (x == 10) {
   System.out.print("x must be 10");
} else {
   System.out.print("x isn't 10");
}
if ((x < 3) & (name.equals("Dirk"))) {
   System.out.println("Gently");
}
System.out.print("this line runs no matter what");
```

Pinceladas sobre Java. Bucles

- Un bucle *while* se ejecuta mientras se cumpla una condición: while (isTrue) {…}.

- Dentro del bucle, las instrucciones van entre {…}

- Operadores de comparación:

  < (menor que)

  > (mayor que)

  == (igualdad)

- Operador de asignación: x = 4;

Ejemplo de funcionamiento

```
int x = 4; // asigna a x el valor 4
while (x > 3) {
    // el código del bucle se ejecutará
    // porque x es mayor que 3
    x = x - 1; // para que el bucle tenga fin
}
int z = 27;
while (z == 17) {
    // el código del bucle no se ejecutará
    // porque z no es igual a 17
}
System.out.print("z = "+z);
```

Traza de un programa

- Es la secuencia de estados por los cuales pasa un programa.
- Es cómo si nosotros fuésemos los compiladores.
- Es una forma de seguir el funcionamiento del programa y saber en todo momento:
  - El valor de las variables
  - El comportamiento del programa (si se cumplen las condiciones de los bucles)

Ejemplo de traza

- Sobre el ejemplo anterior...

x	z	…	Código programa	Cumple condición?	Resultado por pantalla
4			int x = 4;	---	---
4			while (x>3)	SÍ	---
3			x = x - 1;	---	---
3			while (x>3)	NO	---
3	27		int z = 27;	---	---
3	27		while (z == 17)	NO	---
3	27		System.out.println(“z = “ + z);	---	z = 27;

Pinceladas sobre Java. Bifurcaciones

- Si se cumple la condición → ejecuta la acción

```
public class ifTest {
    public static void main (String[] args) {
        int x = 3;

        if (x == 3) {
            System.out.println ("x debe ser 3");
        }

        System.out.println ("Esto va bien");
        }

    }
```

- ¿Qué muestra por pantalla este programa?

Pinceladas sobre Java. Bifurcaciones

- Si se cumple la condición → ejecuta la acción1
- Si no se cumple la condición → ejecuta la acción2

```
public class ifTest2 {
    public static void main (String[] args) {
        int x = 2;

        if (x == 3) {
            System.out.println ("x debe ser 3");
        } else {
            System.out.println ("x no se 3");
        }
        System.out.println ("Esto funciona bien");
    }
}
```

- ¿Qué muestra por pantalla este programa?

Pinceladas sobre Java. Imprimir por pantalla

- ¿Cuál es la diferencia entre System.out.print y System.out.println?
- Haz un programa donde muestras tu nombre y primer apellido con dos sentencias separadas. Denomina el programa MiNombre.java y haz la prueba.
- Tanto a print() como println() le podemos pasar entre “” el texto que queremos escribir, como hemos visto en ejemplos anteriores.