# Bloque I: Introducción a la Programación 1.1 Introducción a Java

1r DAW Semipresencial
Programación
Maria José Lozano Pérez
IES Pere Maria Orts i Bosch

#### Tema 1.1: Introducción a Java

- Cómo trabaja Java
- La estructura del código de Java
- Anatomía de una clase
- El método main()
- Bucles
- Estructuras condicionales
- Ejemplos
- Prácticas

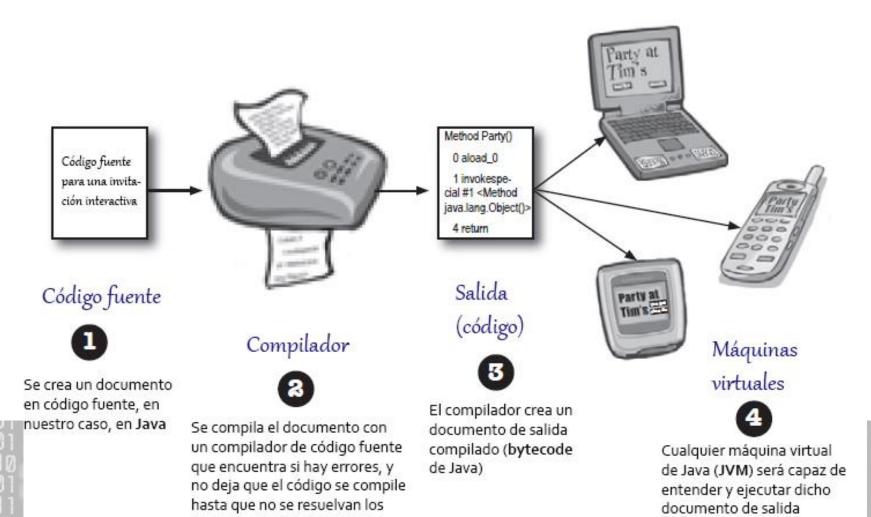


#### Introducción a Java

- Seduce a los programadores desde la 1º versión por su:
  - Sintaxis amistosa
  - Características de la POO (programación orientada a objetos)
  - Manejo de memoria
  - Portabilidad
- Si eres nuevo en Java
  - ¡Tienes suerte!

# Cómo trabaja Java

errores



# Cómo trabaja Java

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

File Edit Window Help Plead

\* javac Party. java

Compilador

3

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

4 return

java.lang.Object()>

Method void buildInvite()

0 new #2 <Class java.awt.Frame>

3 dup

4 invokespecial #3 <Method java.awt.Frame()>

\* File Edit Window Help Swear

\* java Party

Party at Tim's!

Máquinas virtuales



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

Código fuente

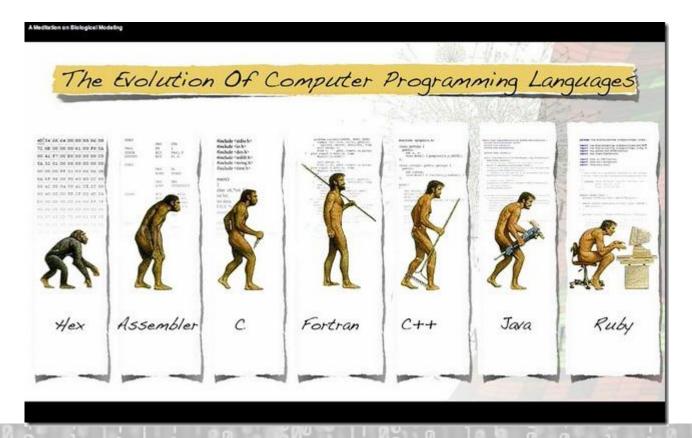
0

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

3

Código compilado: Party.class

#### Evolución: Lenguajes de Programación





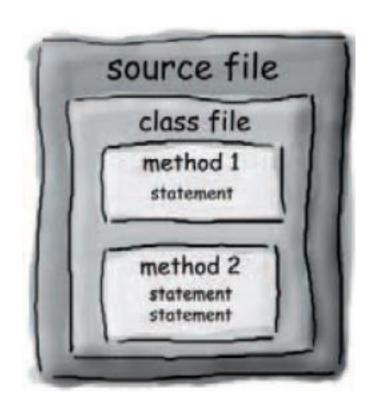


#### Práctica P1.0: Averigua qué hace el código

```
int size = 27;
String name = "Fido";
Dog myDog = new Dog(name, size);
x = size - 5;
if (x < 15) myDog.bark(8);
while (x > 3) {
  myDog.play();
}
int[] numList = {2,4,6,8};
System.out.print("Hello");
System.out.print("Dog: " + name);
String num = "8";
int z = Integer.parseInt(num);
try {
   readTheFile("myFile.txt");
catch(FileNotFoundException ex) {
   System.out.print("File not found.");
```

leclara una	ı variable entera llamada 'size' y le asigna el valor 27	
		_
		_
		_
		_
		_
		_
		_

#### Estructura del código en Java



- ✓ Source file: fichero fuente
- ✓ Class file: fichero de clase
- ✓ Method: métodos
- ✓ **Statements:** sentencias

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

## ¿Qué contiene un fichero fuente?

```
public class Dog {
          clase
```

- Fichero fuente: con extensión .java, almacena, al menos, la definición de una clase.
- Debe ir entre llaves {...}

#### ¿Qué contiene una clase?

```
public class Dog {
  void bark() {
     método
```

- Clase: Contiene uno o más métodos.
- Deben ir definidos dentro de la clase, entre llaves
  - **{...}**

#### ¿Qué contiene un método?

```
public class Dog {
   void bark() {
    statement1;
    statement2;
   }
} sentencias
```

- Método: Es un conjunto de sentencias que realizan una acción.
- Escribiremos instrucciones para especificar cómo ha de funcionar cada método.

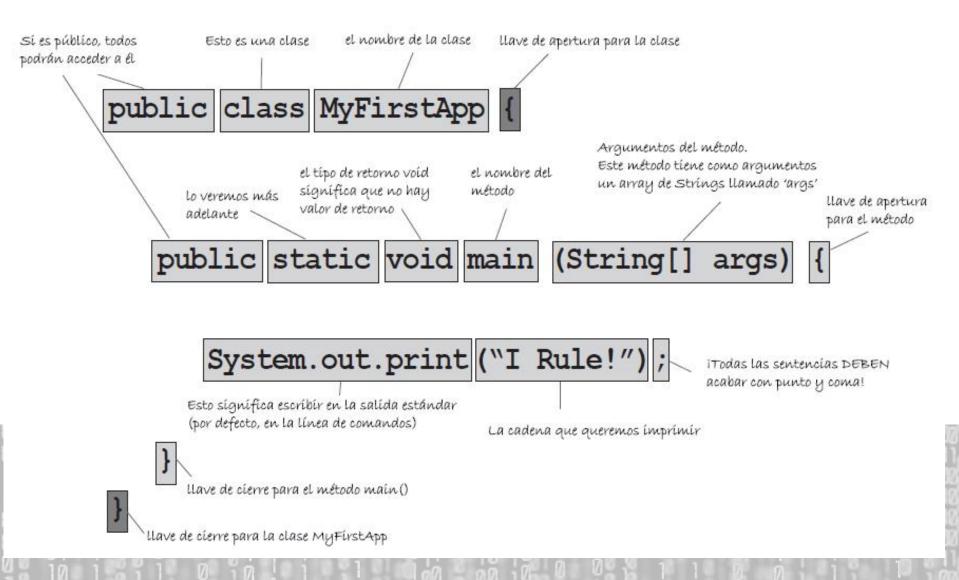
#### JVM: Java Virtual Machine

Cuando el JVM arranca, busca la clase creada y busca un método especial llamado main():

```
public static void main (String[] args) {
  // tu código va aquí
}
```

- El JVM (Máquina Virtual de Java) ejecuta todo lo que encuentra entre corchetes del método main.
- Cada aplicación Java tiene, al menos, una clase y cada clase, al menos, tiene un método.
- Es erróneo: un main() por clase
- Es correcto: un main() por aplicación

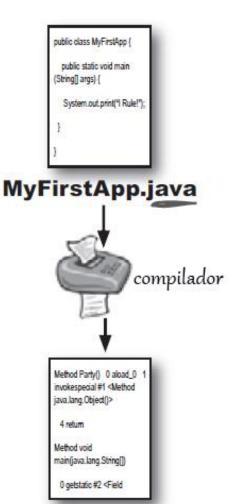
#### Anatomía de una clase



# Mi primera clase con main()

- En Java, todo va dentro de una clase (POO)
- Lo que se ejecuta es el método main() en una clase. Sólo habrá uno en todo el programa.
- Ejecutar un programa en Java es decirle al JVM:
  - Carga la clase y ejecuta su método main()
  - Es lo mínimo para que funcione
- No importa el nº de clases que tengamos en el programa, quien comienza es el método main()

# Mi primera clase con main()



```
public class MyFirstApp {
   public static void main (String[] args) {
      System.out.println("I Rule!");
      System.out.println("The World");
   }
}
```

- 1 Guardar MyFirstApp.java
- Compilar
  javac MyFirstApp.java
- Ejecutar

```
File Edit Window Help Scream
% java MyFirstApp
I Rule!
The World
```

# Cómo escribimos el código

- Empezaremos a escribir el código del programa mediante un editor de texto plano, por ejemplo:
  - PSPAD\* o Notepad++ en Windows
  - Algunos otros\* en Linux

\* Encontraréis el enlace de descarga en el apartado de Recursos, tanto para Windows como para Linux

# Cómo compilamos y ejecutamos

- Empezaremos a utilizar el compilador javac (consola de comandos, símbolo del sistema)
- Habrá que instalar el Entorno JDK\*
  - Habrá que configurar las variables de entorno\*
- También compilaremos con el Entorno Eclipse
  - Para ello deberemos crear un proyecto\* yb una nueva clase

\* Encontraréis enlaces y vídeos de descarga y ayuda, en el apartado de *Recursos* 

#### Práctica P1.1: Hola Mundo

- Requerimientos previos:
  - JDK (Javac)
  - Eclipse
  - Leer y visualizar *Recursos* (enlaces y vídeos)
- Crear primer programa, compilarlo y ejecutarlo
- Enviar como resultado un fichero comprimido llamado B1-1NombreyApellidos, que contenga:
  - Fichero MyFirstApp.java
  - Captura de pantalla del javac
  - Captura de pantalla del Eclipse

#### Repaso a la sintaxis de Java

Cada sentencia debe acabar en punto y coma:

$$x = x + 1;$$

Un comentario comienza con doble barra

$$x = 22;$$

- // esta línea me molesta
- Más de un espacio no es un problema

$$x = 3;$$

#### Repaso a la sintaxis de Java

```
Las variables se declaran con nombre y tipo:
  int peso;
  // tipo: int, nombre: peso
Las clases y métodos deben definirse entre un
  par de llaves:
  public void nadar {
  // código para nadar
```

#### Sentencias

# 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
```



#### Bucles

# loops 6

# 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);
}</pre>
```

# Ramificaciones

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");</pre>
```

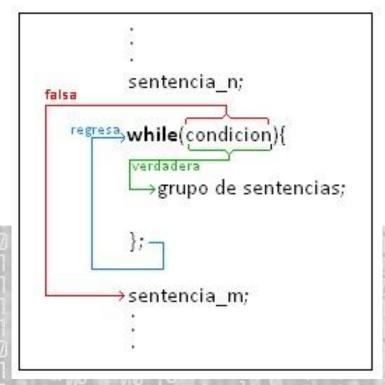


#### Funcionamiento del bucle while

El bucle while se ejecuta mientras se cumple una condición: while (isTrue) {...}

Dentro del bucle, las instrucciones van entre

llaves, {...}



# Operadores de comparación

```
Indican condiciones:
```

```
< (menor que), <= (menor o igual)
> (mayor que), >= (mayor o igual)
== (igualdad)
```

#### **Ejemplos**:

```
while (a<5) {...} while (b>0) {...}
```

```
while (x==0) {...}
```

# Operadores de asignación

Operadores de asignación:

$$x = 4;$$

A la variable x se le asigna el valor 4

$$x = x + 1;$$

A la variable x se le asigna su valor más una unidad

$$\mathbf{x} = \mathbf{x} - 2;$$

A la variable x se le asigna su valor menos dos unidades

# Ejemplo de funcionamiento

```
int x = 4; // asigna 4 a x
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 que pasa un programa. Es hacer nosotros de compilador.
- Es una forma de seguir su funcionamiento y saber en todo momento:
  - El valor de las variables
  - El comportamiento del programa (si se cumplen las condiciones de los bucles)
  - La salida por pantalla
- Seguiremos un formato en forma de tabla

## Ejemplo de traza

#### Traza del programa:

x	z	 Código programa	¿Cumple condición?	Salida por pantalla
4		int x = 4;	-	-
4		while $(x > 3)$	SI	_
3		x = x - 1	-	-
3		while $(x > 3)$	NO	-
3	27	int $z = 27$ ;	-	_
3	27	while (z == 17)	NO	-
3	27	System.out.print("z = "+z);	-	Z = 27

DAW Sempresencial

#### Programa:

```
int x = 4; // asigna 4 a x
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);
```

#### Práctica P1.2: Traza de un programa

```
public class Loopy {
  public static void main (String[] args) {
     int x = 1;
     System.out.println("Antes del bucle");
     while (x < 4) {
       System.out.println("En el bucle");
      System.out.println("El valor de x es " +
      x);
       x = x + 1;
```

#### Práctica P1.2: Traza de un programa

- Prueba a realizar la traza del programa anterior en papel, es decir, qué es lo que aparecerá por pantalla en la línea de comandos. Después pásalo a un archivo de texto llamado trazaB1-2.odt
- Después, comprueba que lo has realizado bien, creando el archivo *Loopy.java* y ejecutando el código compilado, en la consola con *javac* y en *Eclipse*, capturando pantallas.
- Envía todo en el archivo comprimido llamado B1-2NombreyApellidos.

# Tests condicionales: ifTest

■ Si se cumple condición → ejecuta acción

```
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");
```

#### Salida por consola:

```
%java ifTest
x debe ser 3
Esto va bien
```

## Tests condicionales: ifElseTest

 $\blacksquare$  Si se cumple condición  $\rightarrow$  ejecuta acción1, si no se cumple condición  $\rightarrow$  ejecuta acción2

```
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 es 3");
```

Programación

System.out.println ("Esto funciona bien");

Salida por consola:

```
%java ifTest
x no es 3
Esto funciona bien
```

# Diferencias entre print y println

- System.out.print: sigue escribiendo en la misma línea, no salta de línea.
- System.out.println: Inserta una línea nueva tras escribir el texto.

#### Práctica P1.3: Rellena los huecos

Para que el siguiente programa con esta salida:

```
public class DooBee {
  public static void main (String[] args) {
    int x = 1;
    while (x < 1 ) {
        System.out. 2 ("Doo");
        System.out. 3 ("Bee");
        x = x + 1;
    }
    if (x == 4 ) {</pre>
```

System.out.print ("Do"); Salida por consola:

%java DooBee DooBeeDooBeeDo

# Puntos importantes (I)

- Las sentencias acaban todas con punto y coma: ;
- Los bloques de código se definen todos entre llaves { }
- Para declarar una variable entera, asígnale un nombre y un tipo: int x;
- El operador de asignación es el signo =
- El operador de comparación de igualdad es el signo ==
- Un bucle while ejecuta todo el código entre llaves mientras se cumpla la condición
- Si la condición del bucle no se cumple, no se ejecutará el código y automáticamente saldrá del bucle
- Como condición del bucle while, ponemos una comparación con un valor booleano

# Práctica P1.4: Ordena el código

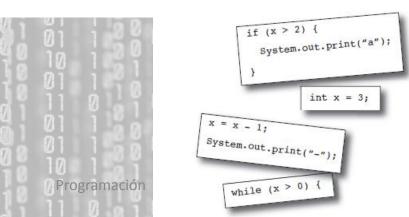
Ordena las sentencias del siguiente programa para obtener esta salida:

```
if (x == 1) {
    System.out.print("d");
    x = x - 1;

if (x == 2) {
    System.out.print("b c");
  }

class Shuffle1 {
  public static void main(String [] args) {
```







# Práctica P1.5: Haz de compilador

De cada uno de los siguientes archivos, sólo uno de ellos compila, el resto dan errores. ¿Podrías identificar cuál compila y cuáles no y encontrar los errores?

```
class Exercise1b {
                                                    public static void main(String [] args) {
class Exercise1b {
                                                                                                                 int x = 5;
 public static void main(String [] args) {
                                                        int x = 5;
                                                                                                                 while (x > 1) {
   int x = 1;
                                                        while (x > 1) {
                                                                                                                   x = x - 1;
   while ( x < 10 ) {
                                                          x = x - 1;
                                                                                                                   if (x < 3) {
     if (x > 3) {
                                                          if (x < 3) {
                                                                                                                     System.out.println("small x");
       System.out.println("big x");
                                                            System.out.println("small x");
```

Intenta resolver el ejercicio observando el código, y después comprueba tus resultados con el compilador de Java.

# Práctica P1.6: Empareja

Empareja los candidatos con las posibles salidas, realizando las trazas en papel del siguiente programa:

```
class Test {
 public static void main(String [] args) {
    int x = 0:
    int y = 0;
    while (x < 5) {
                                                 El código
                                                 candidato irá
      System.out.print(x + "" + y +" ");
      x = x + 1;
```

# Práctica P1.6: Empareja

Estos son los candidatos y las posibles salidas:

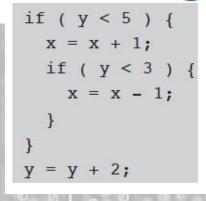
1r DAW Sempresencial

# Candidatos: y = x - y;

$$y = y + x;$$

$$x = x + 1;$$
  
 $y = y + x;$ 

#### 5



#### Posibles salidas:

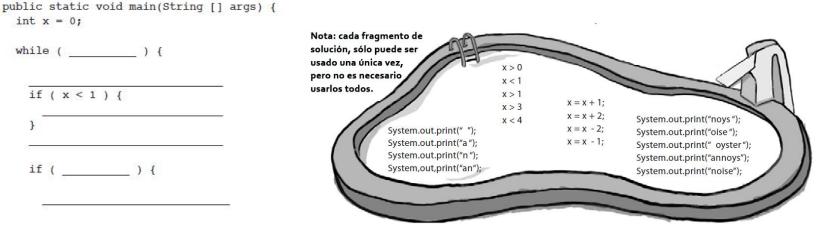
а

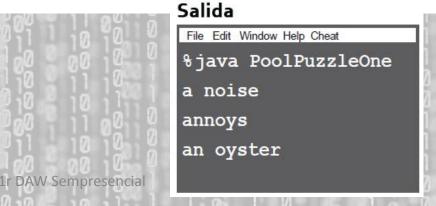
# Práctica P1.7: Puzzle en la piscina

Escoge los fragmentos de código de la piscina y colócalos en los espacios en blanco, de forma que compile y la ejecución produzca la salida esperada.

```
int x = 0;
while ( ) {
 if (x < 1) {
 if ( ) {
 if(x == 1) {
 System.out.println("");
```

class PoolPuzzleOne {





#### Práctica P1.7: Puzzle en la piscina

#### Consideraciones a tener en cuenta:

- Esta actividad es más complicada de lo que parece, no te desanimes
- Cada fragmento de solución sólo podrá ser usado una vez
- No será necesario utilizar todos los fragmentos
- Cada línea representa una línea de código
- Las líneas largas son para instrucciones largas
- Las líneas cortas para instrucciones cortas
- Las condiciones van dentro de bucles o tests condicionales
- Las sentencias van sueltas
- Tened en cuenta cuando hay un print o un println

Programación

#### pFinalTema1: Series de números

Programación

Realizar un programa que muestre por pantalla una serie de números de la siguiente forma:

```
C:\maisse\PROG\pràctiques>java pFinalTema1
Primeros 25 enteros:
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25
Los primeros pares hasta el 50:
2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50
Los primeros impares hasta el 49:
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49
TABLAS DE MULTIPLICAR
Tabla de multiplicar del 1:
1 \times 2 = 2
1 *8 =8
1*9=9
1×10=10
Tabla de multiplicar del 2:
2*9=18
Tabla de multiplicar del 3:
3*1=3
3*10=30
Tabla de multiplicar del 4:
4×2=8
```