

# Sesión 0: Puesta en marcha

## Programación 2

---

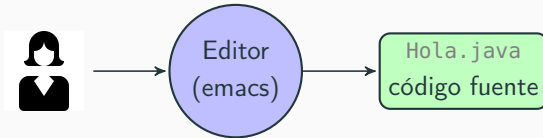
Ángel Herranz

Enero 2019

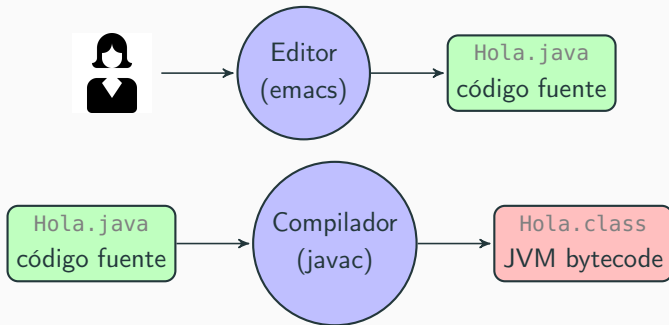
Universidad Politécnica de Madrid

# ¿Cómo funciona Java? i

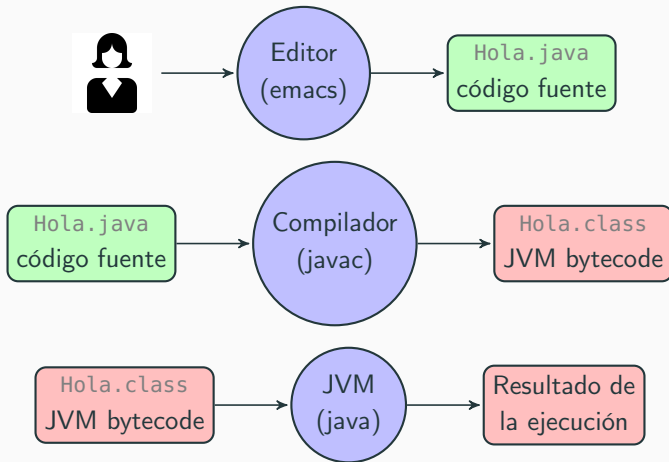
# ¿Cómo funciona Java? i



# ¿Cómo funciona Java? i



# ¿Cómo funciona Java? i



# ¿Cómo funciona Java? ii

- **Compilador** (javac, JDK<sup>1</sup>)
- **JVM** *Java Virtual Machine* (java, JRE<sup>2</sup>)  
Es como una CPU simulada
- Hay múltiples implementaciones  
**HotSpot** (Oracle), **Harmony** (Apache), **Azul** (Azul), **Open J9** (IBM), ...

---

<sup>1</sup> *Java Development Kit*

<sup>2</sup> *Java Runtime Environment*

# ¿Y Eclipse?

- Eclipse es un IDE<sup>3</sup>:

Editor

JDK

JRE

Depurador

Analizadores

- Eclipse IDE, IntelliJ IDEA, DrJava, BlueJ, etc.

---

<sup>3</sup>Integrated Development Environment

## Aprender a programar con un IDE es un error

- Te aleja de los procesos del desarrollo
- Te aleja del modelo computacional
- Te da sugerencias que no puedes entender



## Un editor y un CLI<sup>4</sup>



The screenshot shows a code editor on the left and a terminal window on the right. The code editor contains the following Java code:

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

The terminal window shows the following commands and output:

```
$ ls Hola.*  
Hola.java  
$ javac Hola.java  
$ ls Hola.*  
Hola.class  Hola.java  
$ java Hola  
Hola mundo  
$
```

The status bar at the bottom of the editor shows the file name "Hola.java" and the cursor position "All (6,6)". The terminal window shows the command prompt "U:\*\*\*" and the shell "p2-shell".

<sup>4</sup> *Command Line Interface*

# Guía para un IDE

---

# Aprende a usar la *línea de comandos*

- El interfaz visual de ventanas de los sistemas operativos es muy útil para muchas tareas pero no para todas
- Por ejemplo: crear ficheros, poner en marcha programas, analizar el estado de la red, etc. son tareas que puedes hacer más rápidamente desde la línea de comandos.
- Si usas Unix es más que probable que estés habituado a usar la línea de comandos.
- Si usas Windows puedes encontrar ayuda en este video <https://www.youtube.com/watch?v=VyiGZW0fTxk> (aviso: la música es horrible)
- Después de poner en marcha el *command prompt* necesitas aprender el lenguaje de los comandos. Este enlace puede ayudarte: [https://www.ntu.edu.sg/home/ehchua/programming/howto/CMD\\_Survival.html](https://www.ntu.edu.sg/home/ehchua/programming/howto/CMD_Survival.html)

# Aprende el proceso de desarrollo i

- Cuando se trabaja con un IDE tan potente como Eclipse, hay muchos *detalles* importantes que pasan desapercibidos. Entre ellos el proceso de compilación de Java.
- Podeis empezar por este enlace y luego buscar algo más de información: <http://www.oracle.com/technetwork/java/compile-136656.html>
- Como habéis podido ver, los comandos a usar para compilar y ejecutar desde la línea de comandos son `javac` y `java`.
- Dichos comandos están disponibles en cualquier instalación moderna de Unix.

# Aprende el proceso de desarrollo ii

- Para poder utilizar java desde la línea de comandos en Windows es necesario añadir *Java* a la variable de entorno PATH:
  - Selecciona Start → Computer → System Properties → Advanced system settings → Environment Variables → System variables → PATH.
  - En Vista, selecciona Start → My Computer → Properties → Advanced → Environment Variables → System variables → PATH.
  - En Windows XP, Selecciona Start → Control Panel → System → Advanced → Environment Variables → System variables → PATH.
  - Algo más de información en <https://www.computerhope.com/issues/ch000549.htm>
  - Tienes que añadir el directorio `bin` del JDK al principio de la variable PATH
  - El directorio suele tener un nombre como este:  
`C:\ProgramFiles\Java\jdk1.8.0_144\bin`

# Aprende el proceso de desarrollo iii

- La existencia del directorio depende de que tengas instalado el JDK de Java (lo más probable es que esté instalado con DrJava o con Eclipse, pero siempre puedes instalarlo directamente: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
- Tendrías que añadir el directorio en cuestión al principio del PATH incluyendo un punto y coma, algo como esto:  
`C:\ProgramFiles\Java\jdk1.8.0_144\bin;`
- Asegúrate que se guarda (dale a OK varias veces ;)
- Comprueba que todo está ok ejecutando un *command prompt*:
  - All Programs → Accessories → Command Prompt
  - Tiene que aparecer algo como esto:  
`Microsoft Windows [Version 6.3.9600]  
(c) 2013 Microsoft Corporation. All rights reserved.  
C:\Users\username>`
  - Para confirmar que todo ha ido bien y que tienes disponible el compilador escribe:

# Aprende el proceso de desarrollo iv

```
C:\Users\username>javac -version
```

```
javac 1.8.0_144
```

```
C:\Users\username>
```

- Para confirmar que todo ha ido bien y que tienes disponible la máquina escribe:

```
C:\Users\username>java -version
```

```
java version "1.8.0_144"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_144)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

- Personaliza tu línea de comandos, la vas a usar frecuentemente:
  - Click con el botón derecho en la barra del command prompt
  - Seccionar Properties
  - Establecer Layout → Screen Buffer Size to 80 × 500.
  - Establecer Options → Edit Options → QuickEdit Mode.
  - Seleccionar Options → Edit Options → Insert Mode.

# Aprende el proceso de desarrollo v

- Para compilar desde la línea de comandos
  - Navega desde el command prompt al directorio donde tienes el código fuente (ficheros .java)
  - Por ejemplo

```
C:\Users\username>cd c:\users\username\sesion01
C:\Users\username\sesion01
```
  - username es tu usuario en windows
  - Compila usando javac:

```
C:\Users\username\sesion01>javac HolaMundo.java
C:\Users\username\sesion01>
```
- Para ejecutar desde la línea de comandos usamos el intérprete java:

```
C:\Users\username\sesion01>java HolaMundo
Hola mundo
C:\Users\username\sesion01>
```



# Aprende a usar un buen editor i

Te recomiendo iniciar una búsqueda de un buen editor de texto. De nuevo me atrevo con unas recomendaciones, por orden:

1. Emacs. Es más que un editor, es un compilador de Lisp, puedes editar ficheros de texto delegando bastante inteligencia en sus *modos*.
2. Vi. Editor clásico, configurable. Muchas herramientas en Unix siguen sus combinaciones de teclas (el manual, por ejemplo).
3. Nano. Editor muy básico, viene preinstalado con casi cualquier distribución de Linux.

# Aprende a usar un buen editor ii

4. Gedit. No es muy listo pero es muy socorrido ya que lo encontraréis instalado en cualquier Ubuntu.
5. Sublime Text. Editor ligero. Permite la instalación de plugins. Tuvo mucha fama hace unos años y muchos desarrolladores que siguen usándolo.
6. Atom. Editor creado por Github de código libre, multiplataforma (creado con tecnologías web usando el framework llamado Electron) y apto para casi cualquier lenguaje de programación. Es muy configurable y se le pueden instalar multitud de plugins, hasta el punto de consumirte toda la RAM si instalas muchos.

# Aprende a usar un buen editor iii

7. Visual Studio Code (VSCode). Editor de moda entre muchos desarrolladores. La competencia a Atom y Sublime Text de Microsoft. Usa la misma tecnología que Atom pero está mas optimizado. Permite la instalación de infinidad de plugins (extensiones), convirtiéndolo en un IDE muy completo para cualquier lenguaje.

Elige un editor, y dale tiempo. Emacs y vi puede que sean los más potentes. Dales más tiempo, ellos te lo devolverán dentro de un par de meses. Aquí tienes los editores mas usados este 2018 segun StackOverflow.

**String args[]?**

---



# Hola mundo

- Editar

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

- Compilar: `javac Hola.java`
- Ejecutar: `java Hola`

# Hola ...

- Vamos a modificar la implementación:

```
$ java Hola
```

```
Hola mundo
```

```
$ java Hola clase
```

```
Hola clase
```

```
$ java Hola angel
```

```
Hola angel
```

# Variables, objetos y referencias

---

# ¿Qué es una clase?



# ¿Qué es una clase?

Plantilla  
para fabricar objetos  
que tiene el mismo  
comportamiento

# ¿Qué es un objeto?

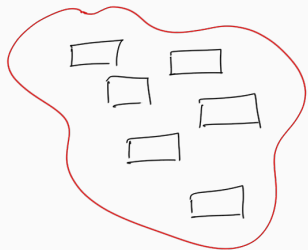
# ¿Qué es un objeto?

Agrupación de datos junto con  
operaciones capaces de  
consultarlos y modificarlos

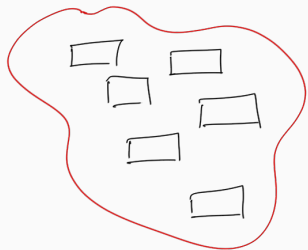
# ¿Qué es un objeto?

Agrupación de datos junto con  
operaciones capaces de  
consultarlos y modificarlos  
Los objetos viven en memoria

# ¿Qué es un objeto?



# ¿Qué es un objeto?





## Cascara.java y Magia.java

```
public class Cascara {  
    int x;  
}
```

```
public class Magia {  
    public static void main(String args[]) {  
        Cascara c;  
    }  
}
```

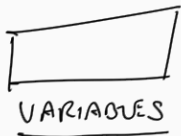
# Ejecutar java Magia

- No parece pasar *nada*
- Pero... ¿qué pasa realmente?

## Modelo de Ejecución




# Nuestro lenguaje



42 true

DAFOS  
PRIMITIVOS

null +   
REFERENCIAS