

TEMAS A ABORDAR:

- Características principales JAVA
- Historia de JAVA
- Herramientas de desarrollo y ejecución (JVM/JRE)
- Definición de JVM
- Entorno de desarrollo NETBEANS



Características Principales

- **LENGUAJE DE ALTO NIVEL:** Basado en el lenguaje natural
- **COMPILADO:** Se implementa mediante un compilador
- **ORIENTADO A OBJETOS:** Basado en el paradigma de programación orientado a objetos
- **DE FORMATO LIBRE:** No se requiere tabular ni indentar el código
- **MULTIPLATAFORMA:** El código JAVA se ejecuta sobre su maquina virtual.
- **TIPADO ESTATICAMENTE:** La comprobación de la tipificación se realiza durante el compilado

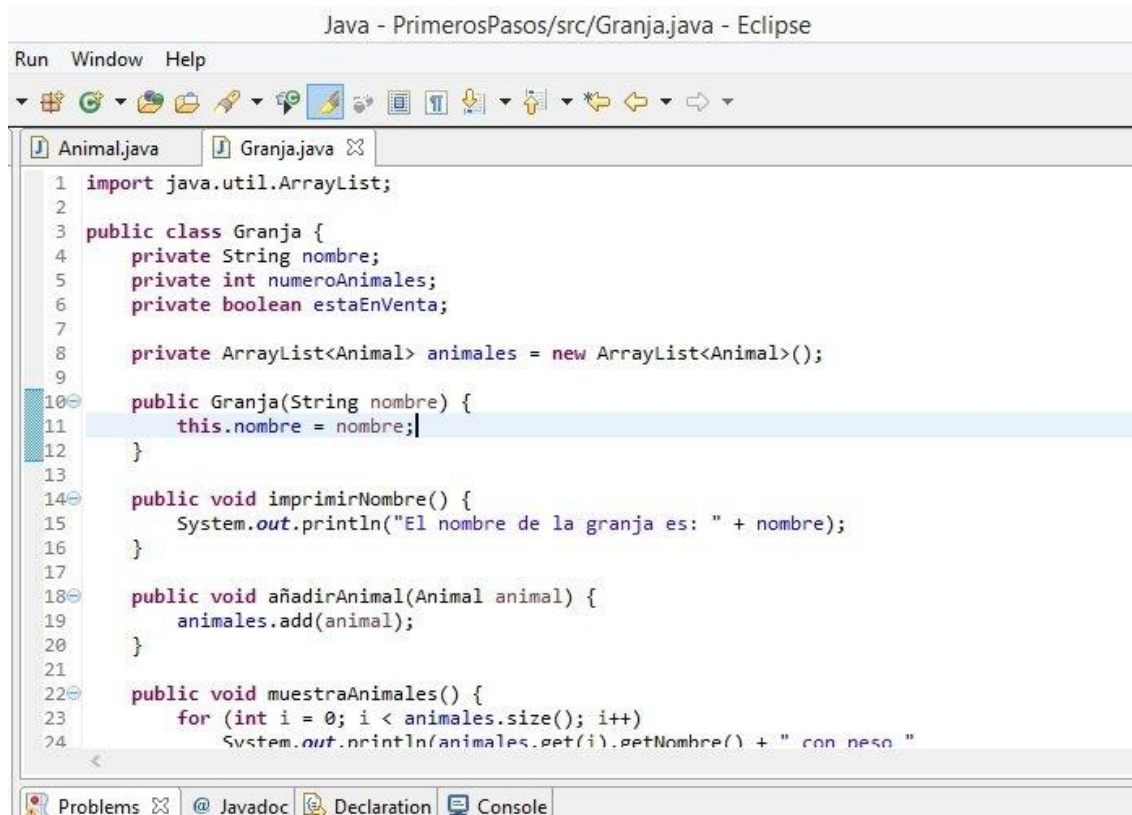


Historia:

- Creado en 1991 por “SUN Microsystems”
- Comercializado por primera vez en 1995
- Sus objetivos principales son:
 - Proporcionar un lenguaje que sea independiente de la plataforma
 - Basado en un entorno de ejecución seguro, ligero y gratuito
 - Con una sintaxis similar a la desplegada en C++

https://www.java.com/es/download/faq/whatis_java.xml

¿Por qué JAVA es un lenguaje de alto nivel?



```
Java - PrimerosPasos/src/Granja.java - Eclipse
Run Window Help
Animal.java Granja.java
1 import java.util.ArrayList;
2
3 public class Granja {
4     private String nombre;
5     private int numeroAnimales;
6     private boolean estaEnVenta;
7
8     private ArrayList<Animal> animales = new ArrayList<Animal>();
9
10    public Granja(String nombre) {
11        this.nombre = nombre;
12    }
13
14    public void imprimirNombre() {
15        System.out.println("El nombre de la granja es: " + nombre);
16    }
17
18    public void añadirAnimal(Animal animal) {
19        animales.add(animal);
20    }
21
22    public void muestraAnimales() {
23        for (int i = 0; i < animales.size(); i++)
24            System.out.println(animales.get(i).getNombre() + " con peso "
```

- Por que utiliza para definir sus instrucciones palabras del lenguaje natural (ingles)
- Ejemplos de ello son palabras como “CASE”, “IF”, “FOR”, “WHILE”
- La facilidad que implica la utilización de este tipo de palabras se contrapone con los lenguajes de bajo nivel (código binario)

¿Qué herramientas necesito para desarrollar o ejecutar código JAVA?

- **JDK: Java Development Kit (Kit de desarrollo JAVA)**

Librerías necesarias para desarrollar en JAVA + Máquina Virtual

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

- **JRE: Java Runtime Environment (Ambiente de ejecución de JAVA)**

<https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

Solo máquina virtual que permite ejecutar aplicaciones desarrolladas sobre JAVA

- **IDE: Integrated Development Environment (Ambiente de desarrollo Integrado)**

Herramientas para la edición y compilación de código

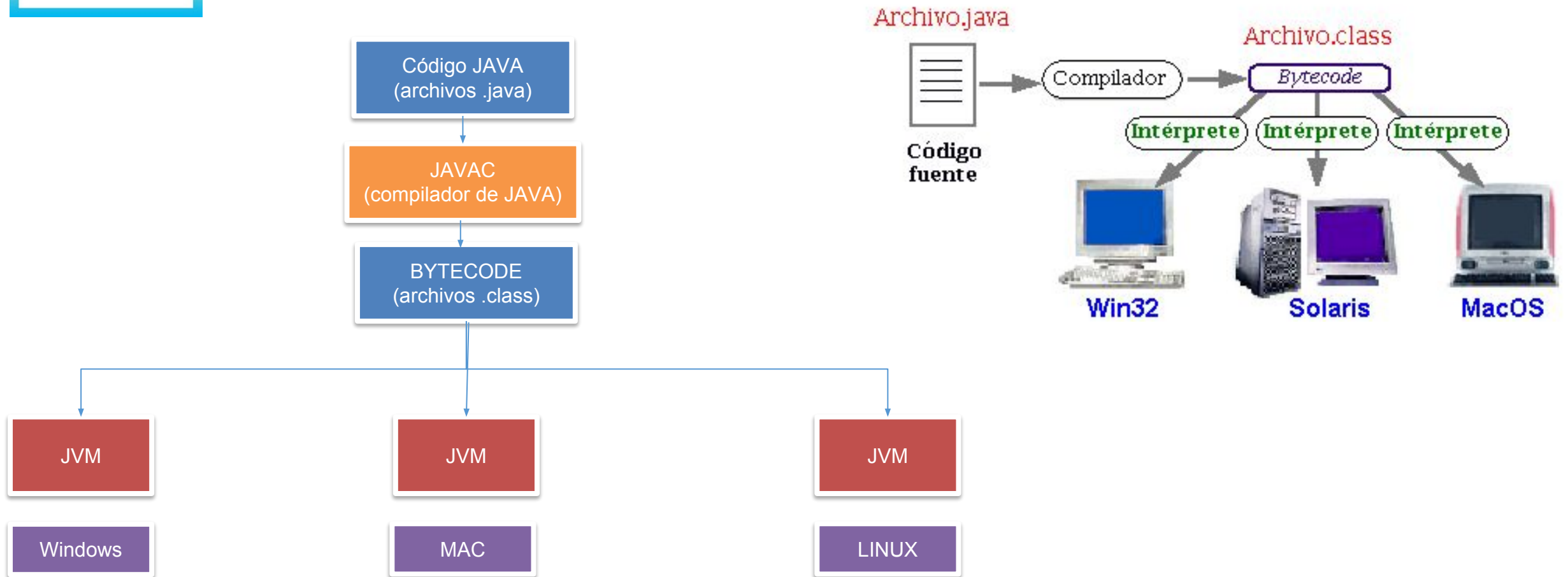
¿Qué es una maquina virtual? (“JVM”)

- Una máquina virtual Java (en inglés Java Virtual Machine, JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java

- En otras palabras, la JVM es una abstracción de una máquina real, que es capaz de entender el *Byte Code* creado por el compilador de Java y traducirlo en instrucciones nativas equivalente que a su vez el sistema operativo actual es capaz de entender, ejecutando realmente la aplicación.

¿Cómo funciona la ejecución de código JAVA?



¿Qué es un “IDE”?

- Es una aplicación que proporciona una interfaz de trabajo que contiene un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Ejemplos IDE's JAVA:

- **ECLIPSE:**
 - <https://www.eclipse.org/downloads/packages/>
- **NETBEANS:**
 - <https://netbeans.org/features/java/index.html>
- **IntelliJ IDEA:**
 - <https://www.jetbrains.com/idea/>

PRACTICA:

- Descargar e instalar la JDK de JAVA
- Descargar e instalar NETBEANS
- Crear un nuevo proyecto

Entorno de desarrollo NETBEANS:

- **Archivo JAVA:** Definición de la clase JAVA en lenguaje natural
 - <nombre proyecto>\src\ **archivo.java**
- **Compilar:** Archivo bytecode (binario) traducido por el compilador (lo que ejecuta la JVM)
 - <nombre proyecto>\build\classes\ **archivo.class**
- **Generación de “build”:** Archivo ejecutable de JAVA
 - <nombre proyecto>\dist\ **archivo.jar**
- **Debugger:** Verifica la sintaxis del código

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

- Todo programa es escrito como el método estático llamado `main` en una clase cualquiera
- Este método se empieza a ejecutar cuando se invoca el intérprete de java para una clase dada
- `args` es un arreglo de Strings que contiene los parámetros
- con los que fué invocado el programa.

Comentarios

Java ofrece tres tipos de comentarios: dos para comentarios regulares en el código fuente y uno para la documentación especial del sistema javadoc.

Comentarios de una sola línea.

Para comentar una sola línea se utiliza la doble diagonal `//`. El comentario se inicia cuando se encuentra la doble diagonal y continua hasta el final de la línea.

```
// Este es un comentario de una sola linea //Este es otro comentario
```

Comentarios de varias líneas.

Los comentarios de varias líneas se incluyen entre los símbolos `/*` y `*/`, como en C y C++.

```
/* Este es un ejemplo de un comentario de varias líneas. */
```

Tipos primitivos de datos en JAVA

- enteros: `int`, `long`, `short`, `byte`

Const. `1`, `-1`, `1024`, `1L`

- reales: `float`, `double`

Const. `1.0`, `-3.14159`, `1.5e4`, `1.0f`

- caracter: `char`

Const. `'a'`, `'X'`, `'@'`

- lógico: `boolean`

Const. `true`, `false`

Constantes de String: `"Hola"`, `"12 de Abril"`

Palabras clave en Java

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

- Palabras claves tienen significado especial y no pueden usarse como identificadores de variables ni clases ni métodos

Declaraciones

```
int i;  
int i = 1;  
double pi = 3.14159;  
char c = 'a';  
boolean estamos_bien = true;
```

Las declaraciones de variables pueden ir en cualquier parte del programa pero siempre antes de que la variable sea usada. Hay que tener cuidado con el rango de validez (scope) de la declaración

Expresiones y asignación

- Aritmeticas: `suma + 20 * c / (mod % 3)`
- Relacionales: `a > b`, `b >= c`, `c != 4`, `a == 0`
- De String: `"hola "+ nombre + " hoy es "+
dia + "de"+mes`
- Casts: `(int) pi` (`pi = 3.1`) `(int) (Math.random()*100)+1)`
- Otros: `a == 1 ? a+1 : a-1`
- Asignacion: `a = 1;`
- Asignación como operador: `a = b = c = d = 0;`
`boolean cero = (b = c - 10) == 0;`

Variables

Tipo	Descripción	Tamaño/Formato
<i>Números enteros</i>		
byte	Entero byte	8-bit 2's
short	Entero corto	16-bit 2's
int	Entero	32-bit 2's
long	Entero largo	64-bit 2's
<i>Números reales</i>		
float	Punto flotante	32-bit IEEE 754
double	Punto flotante de doble precisión	64-bit IEEE 754
<i>Otros tipos</i>		
char	Un solo carácter	16-bit caracteres Unicode
boolean	Un valor booleano	true o false