

Capítulo 1

Introducción a Java

Objetivos

Comprender los conceptos básicos de programas y algoritmos.

Describir la relación entre Java y la World Wide Web.

Entender el significado de la especificación de lenguaje Java, API, JDK e IDE.

Conocer y usar un IDE como NetBeans, Eclipse o IntelliJ IDEA.

Escribir un programa Java simple.

Mostrar la salida en la consola.

Explicar la sintaxis básica de un programa Java.

Crear, compilar y ejecutar programas Java.

Utilizar correctamente el estilo de programación Java y los programas de documentos.

Explicar las diferencias entre errores de sintaxis, errores de ejecución y errores lógicos.

Desarrollar programas Java.

Programas

Los programas informáticos, conocidos como software, son instrucciones para el ordenador. Podemos decirle a qué hacer a través de los programas. Sin programas, un ordenador es una máquina vacía. Los ordenadores no entienden los lenguajes humanos, por lo que necesitamos utilizar los lenguajes del ordenador para comunicarnos con ellos.

Los programas se escriben utilizando lenguajes de programación.

El ordenador solo sabe realizar las siguientes operaciones.

- Sumar, restar, multiplicar y dividir dos valores numéricos.
- Comparar dos valores: comprobar si son iguales, si el primero es mayor o si el segundo es mayor.
- Almacenar o recuperar información, en la memoria o en dispositivos externos.

Combinando estas operaciones realizamos los programas.

Algoritmo

Se denomina algoritmo a una secuencia ordenada y finita de las operaciones a ejecutar para conseguir ciertos resultados determinados o conjunto ordenado de pasos que hay que realizar para llevar a cabo una acción.

Ejemplo: Realización de una tortilla

- ☐ 1.Cascar los dos huevos en el plato hondo.
- ☐ 2.Añadirles la sal a gusto.
- ☐ 3.Remover los huevos enérgicamente con el tenedor hasta que queden perfectamente batidos.
- ☐ 4.Encender el fuego de la cocina.
- ☐ 5.Echar el aceite de oliva en la sartén.
- ☐ 6.Poner la sartén con el aceite a calentar.
- ☐ 7.Verter, cuando el aceite esté caliente, el contenido del plato hondo en la sartén.
- ☐ 8.Esperar unos segundos mientras se fríe la base de los huevos batidos.
- ☐ 9.Doblar al centro la base de los huevos batidos.
- ☐ 10.Esperar unos segundos hasta que se termine de freír.
- ☐ 11.Sacar la tortilla de la sartén y servir.

Programa

- Para que un algoritmo pueda ser utilizado en la computadora, debe convertirse primero en un programa
- Programa: Se escribe en un determinado lenguaje de programación
- Existen muchos lenguajes de programación como: BASIC, COBOL, Pascal, FORTRAN, C, Java, C++, Ada, Clipper, etc.

Partes de un programa de ordenador

- Descripción de las acciones que deben ejecutarse mediante instrucciones
- Descripción de los datos que son manipulados por esas instrucciones mediante declaraciones y definiciones.

Características de los algoritmos

- Finito: El algoritmo debe finalizar su ejecución en algún punto.
- Legibilidad: Debe estar escrito de tal forma que sea fácil de leer y entender.
- Modificabilidad: Se debe poder realizar fácilmente modificaciones y actualizaciones del programa, para adaptarse a nuevos requisitos.
- Eficiencia: Debe minimizar el espacio de memoria usado y el tiempo de ejecución empleado.
- Modularidad: el programa, llamado programa principal, puede estar subdividido en módulos o programa más pequeños llamados subprogramas, cada uno de los cuales realiza una parte del problema.
- Estructuración: comprende todas las características anteriores. Como consecuencia de una mayor estructuración, resulta más fácil:
 - Leerlo
 - Modificarlo
 - Eliminar las partes del programa que se puedan repetir

Fases básicas del diseño y puesta a punto de un programa

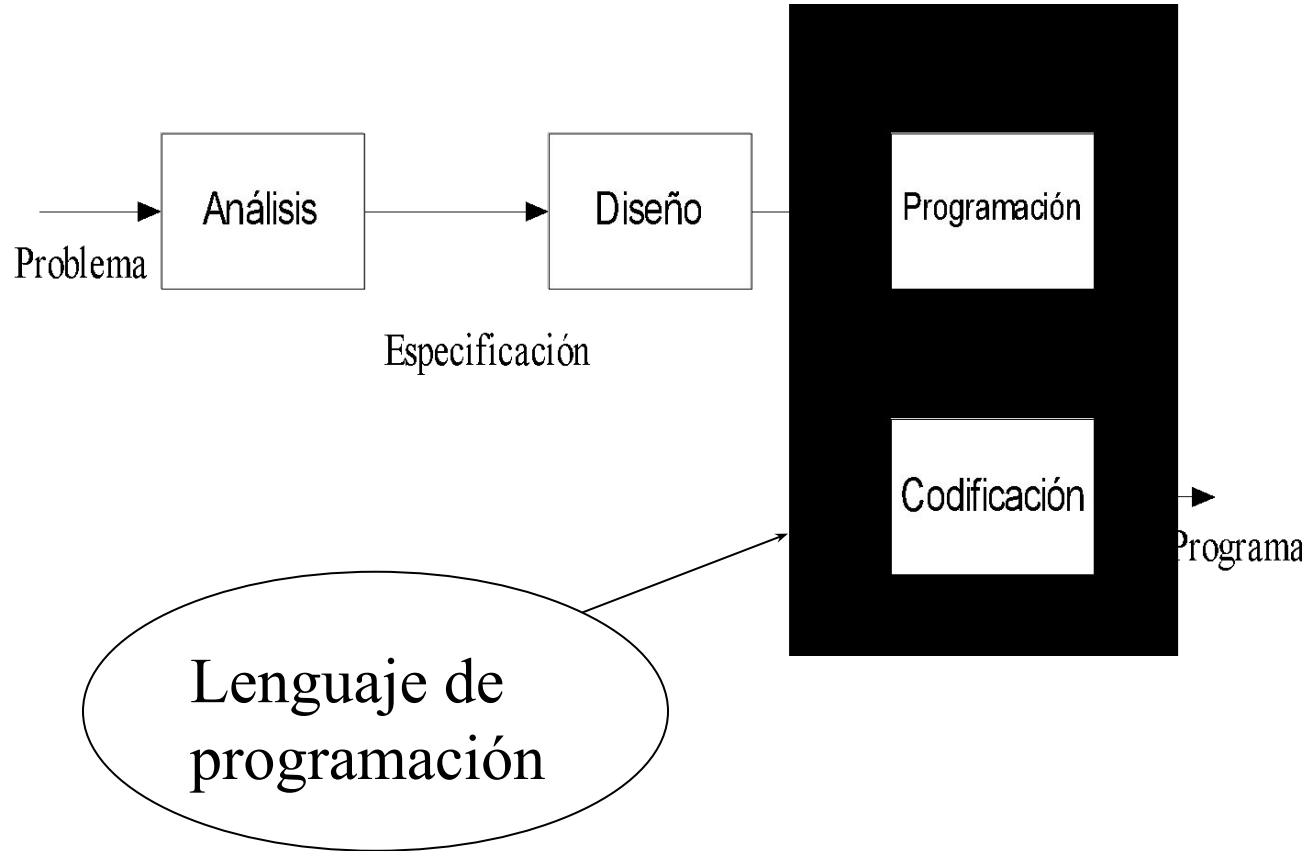
■ **Diseño del programa**

- Fase de análisis
- Fase de diseño
- Fase de programación
- Fase de codificación

■ **Puesta a punto del programa**

- Fase de Prueba
- Fase de Implantación
- Fase de Mantenimiento

Diseño del programa



Lenguajes de programación

Lenguaje Máquina Lenguaje Ensamblador Lenguaje de Alto-Nivel

El lenguaje de máquina es un conjunto de instrucciones primitivas incorporadas en cada computadora. Las instrucciones están en forma de código binario, por lo que tiene que introducir códigos binarios para varias instrucciones. Un programa en lenguaje de máquina nativo es un proceso tedioso. Además, los programas son muy difíciles de leer y modificar.

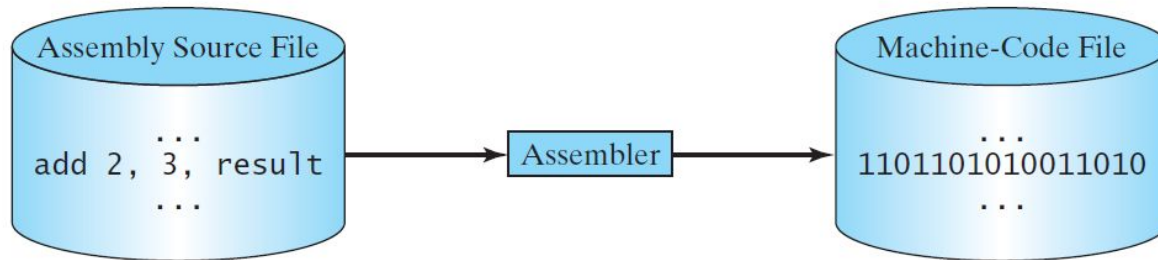
Por ejemplo, para agregar dos números, puede escribir una instrucción en binario de la siguiente manera:

```
1101101010011010
```

Lenguajes de programación

Lenguaje Máquina **Lenguaje Ensamblador** Lenguaje de Alto-Nivel

Se desarrollaron lenguajes ensamblador para facilitar la programación. Dado que la computadora no puede entender el lenguaje ensamblador, sin embargo, un programa llamado ensamblador se utiliza para convertir programas de lenguaje ensamblador en código de máquina. Por ejemplo, para agregar dos números, puede escribir una instrucción en código ensamblador de la siguiente manera: `ADDF3 R1, R2, R3`



Lenguajes de programación

Lenguaje Máquina Lenguaje Ensamblador **Lenguaje de Alto-Nivel**

Los lenguajes de alto nivel son normalmente en inglés y fáciles de aprender y programar. Por ejemplo, la siguiente es una instrucción de lenguaje de alto nivel que calcula el área de un círculo con radio 5:

```
area = 5 * 5 * 3.1415;
```

Lenguajes de Alto Nivel más conocidos

Language	Description
Ada	Llamado como Ada Lovelace, la primera programadora de la historia. El lenguaje Ada fue desarrollado para el Departamento de Defensa y se utiliza principalmente en proyectos de defensa.
BASIC	Beginner's All-purpose Symbolic Instruction Code Fue diseñado para ser aprendido y utilizado fácilmente por los principiantes.
C	Desarrollado en los Laboratorios Bell. C combina la potencia de un lenguaje ensamblador con la facilidad de uso y la portabilidad de un lenguaje de alto nivel.
C++	C++ es un lenguaje orientado a objetos, basado en C.
C#	Se pronuncia "C Sharp". Es un híbrido de Java y C ++ y fue desarrollado por Microsoft.
COBOL	COMmon Business Oriented Language. Se utiliza para aplicaciones empresariales.
FORTRAN	FORmula TRANslation. Muy utilizado para aplicaciones científicas y matemáticas.
Java	Desarrollado por Sun Microsystems, ahora parte de Oracle. Es ampliamente utilizado para desarrollar aplicaciones de Internet independientes de la plataforma.
Pascal	Es un lenguaje simple, estructurado, de uso general, principalmente para la enseñanza de la programación.
Python	Un lenguaje de scripting, de uso general bueno para escribir programas cortos.
Visual Basic	Fue desarrollado por Microsoft y permite a los programadores desarrollar rápidamente interfaces gráficas de usuario.

Interpretación / Compilación de código fuente

Un programa escrito en un lenguaje de alto nivel se llama un programa fuente o código fuente.

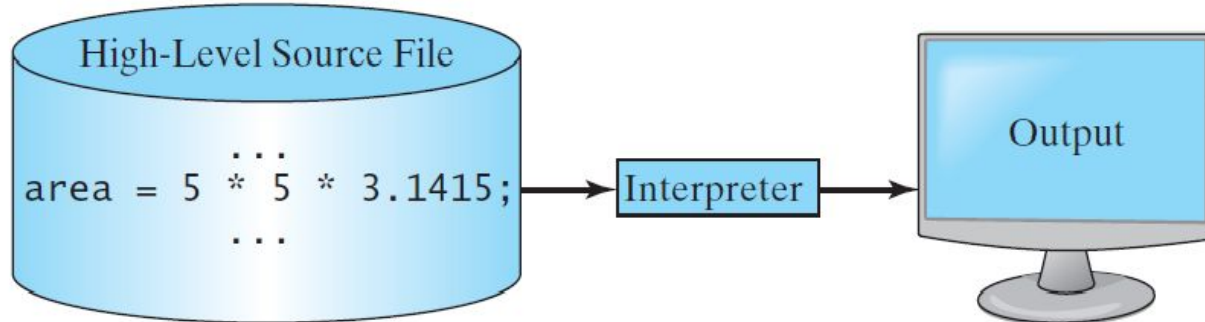
Debido a que un equipo no puede entender un programa fuente, un programa fuente debe traducirse en código de máquina para su ejecución.

La traducción puede hacerse utilizando otra herramienta de programación llamada intérprete o compilador.

Interpretación de código fuente

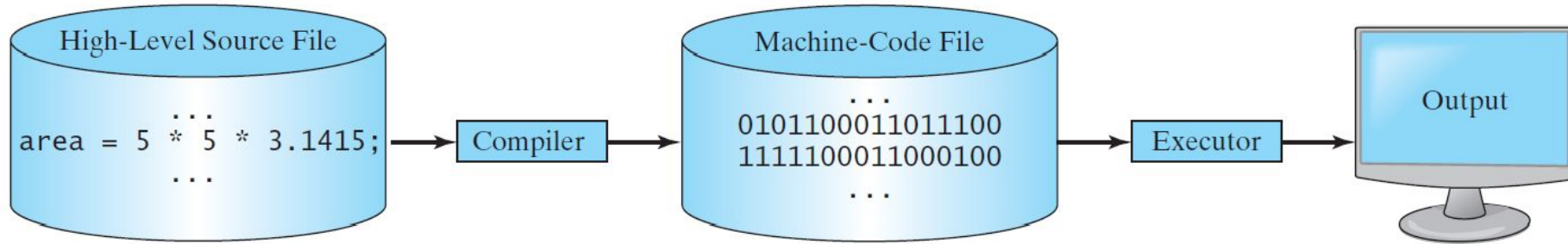
Un intérprete lee una instrucción del código fuente, la traduce al código máquina o al código de la máquina virtual y luego lo ejecuta de inmediato, como se muestra en la siguiente figura.

Tenga en cuenta que una instrucción del código fuente puede traducirse a varias instrucciones de la máquina.



Compilando código fuente

Un compilador traduce todo el código fuente en un archivo de código máquina y el archivo de código máquina se ejecuta, como se muestra en la siguiente figura.



Por qué Java?

La respuesta es que Java permite a los usuarios desarrollar e implementar aplicaciones en Internet para servidores, computadoras de escritorio y pequeños dispositivos portátiles.

El futuro de la informática está siendo profundamente influenciado por Internet, y Java promete seguir siendo una gran parte de ese futuro.

- Java es un lenguaje de programación de propósito general.
- Java es el lenguaje de programación de Internet.
- Java se puede utilizar para desarrollar aplicaciones independientes.
- Java se puede utilizar para desarrollar aplicaciones que se ejecutan desde un navegador.
- Java también se puede utilizar para desarrollar aplicaciones para dispositivos portátiles.
- Java se puede utilizar para desarrollar aplicaciones para servidores Web.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es intepretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Java está parcialmente modelado en C ++, pero muy simplificado y mejorado. Algunas personas se refieren a Java como "C++ -" porque es como C++, pero con más funcionalidad y menos aspectos negativos.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Java es inherentemente orientado a objetos. Aunque muchos lenguajes orientados a objetos comenzaron estrictamente como lenguajes procedurales, Java fue diseñado desde el principio para ser orientado a objetos. Programación orientada a objetos (OOP) es un enfoque de programación popular que está reemplazando las técnicas de programación de procedimientos tradicionales.

Uno de los temas centrales en el desarrollo de software es cómo reutilizar código. La programación orientada a objetos proporciona gran flexibilidad, modularidad, claridad y reutilización a través de la encapsulación, la herencia y el polimorfismo.

Características de Java

- Java es simple
- Java está orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

La computación distribuida implica que varios equipos trabajen juntos en una red. Java está diseñado para facilitar la computación distribuida. Dado que la capacidad de conexión en red está inherentemente integrada en Java, escribir programas de red es como enviar y recibir datos desde y hacia un archivo.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Necesitamos un intérprete para ejecutar programas Java. Los programas se compilan en el código de la Máquina Virtual de Java, llamado bytecode. El bytecode es independiente de la máquina y puede ejecutarse en cualquier máquina que tenga un intérprete de Java, que es parte de la Máquina Virtual de Java (JVM).

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Los compiladores de Java pueden detectar muchos problemas que aparecerían por primera vez en el tiempo de ejecución en otros lenguajes.

Java ha eliminado ciertos tipos de construcciones de programación propensas a errores que se encuentran en otros lenguajes.

Java tiene una característica de manejo de excepciones en tiempo de ejecución para proporcionar soporte de programación para incrementar la robustez.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Java implementa varios mecanismos de seguridad para proteger su sistema contra los daños causados por programas de fuentes desconocidas.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Escribe una vez, corre en cualquier lugar

Con una máquina virtual Java (JVM), puede escribir un programa que se ejecutará en cualquier plataforma.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es intepretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Debido a que Java es neutral en arquitectura, los programas Java son portátiles. Pueden ejecutarse en cualquier plataforma sin ser recompilados.

Características de Java

- Java es simple
- Java es orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Rendimiento de Java Debido a que Java es neutral en arquitectura, los programas Java son portátiles. Pueden ejecutarse en cualquier plataforma sin ser recompilados.

Características de Java

- Java es simple
- Java está orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

La programación multiproceso se integra perfectamente en Java, mientras que en otros lenguajes hay que llamar a procedimientos específicos del sistema operativo para habilitar el multiprocesamiento.

Características de Java

- Java es simple
- Java está orientado a objetos
- Java es distribuido
- Java es interpretado
- Java es robusto
- Java es seguro
- Java es Arquitectura-Neutral
- Java es portátil
- Rendimiento de Java
- Java es multiproceso
- Java es dinámico

Java fue diseñado para adaptarse a un entorno en evolución. El nuevo código se puede cargar sobre la marcha sin recompilación. No es necesario que los desarrolladores creen y que los usuarios instalen nuevas versiones de software. Las nuevas características se pueden incorporar de forma transparente según sea necesario.

Versiones de JDK

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) a. k. a. JDK 5 or Java 5
- JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- JDK 1.8 (2014) a. k. a. JDK 8 or Java 8
- JDK 9 (2017) ...
- JDK 21 (2023) (actualmente)

Ediciones JDK

- Java Standard Edition (J2SE)
 - J2SE puede usarse para desarrollar aplicaciones o applets independientes del lado del cliente.
- Java Enterprise Edition (J2EE)
 - J2EE se puede utilizar para desarrollar aplicaciones del lado del servidor como servlets Java, Java ServerPages y Java ServerFaces.
- Java Micro Edition (J2ME).
 - J2ME se puede utilizar para desarrollar aplicaciones para dispositivos móviles como teléfonos y tablets.

Este curso utiliza J2SE para introducir la programación Java.

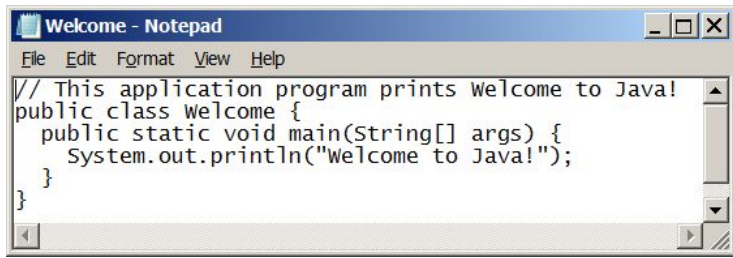
Instalación y uso de JDK

¿Qué hace falta para crear un programa en Java?

Existen diversas herramientas que nos permitirán crear programas en Java. La propia Oracle suministra un kit de desarrollo oficial que se conoce como JDK (Java Development Kit). Es de libre distribución y se puede conseguir en la propia página Web de Oracle.

El inconveniente del JDK es que no incluye un editor para crear nuestros programas, sólo las herramientas para generar el programa ejecutable y para probarlo, para ello usaremos un IDE de los siguientes:

- NetBeans
- Eclipse
- IntelliJ IDEA



```
// This application program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Source code (developed by the programmer)

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

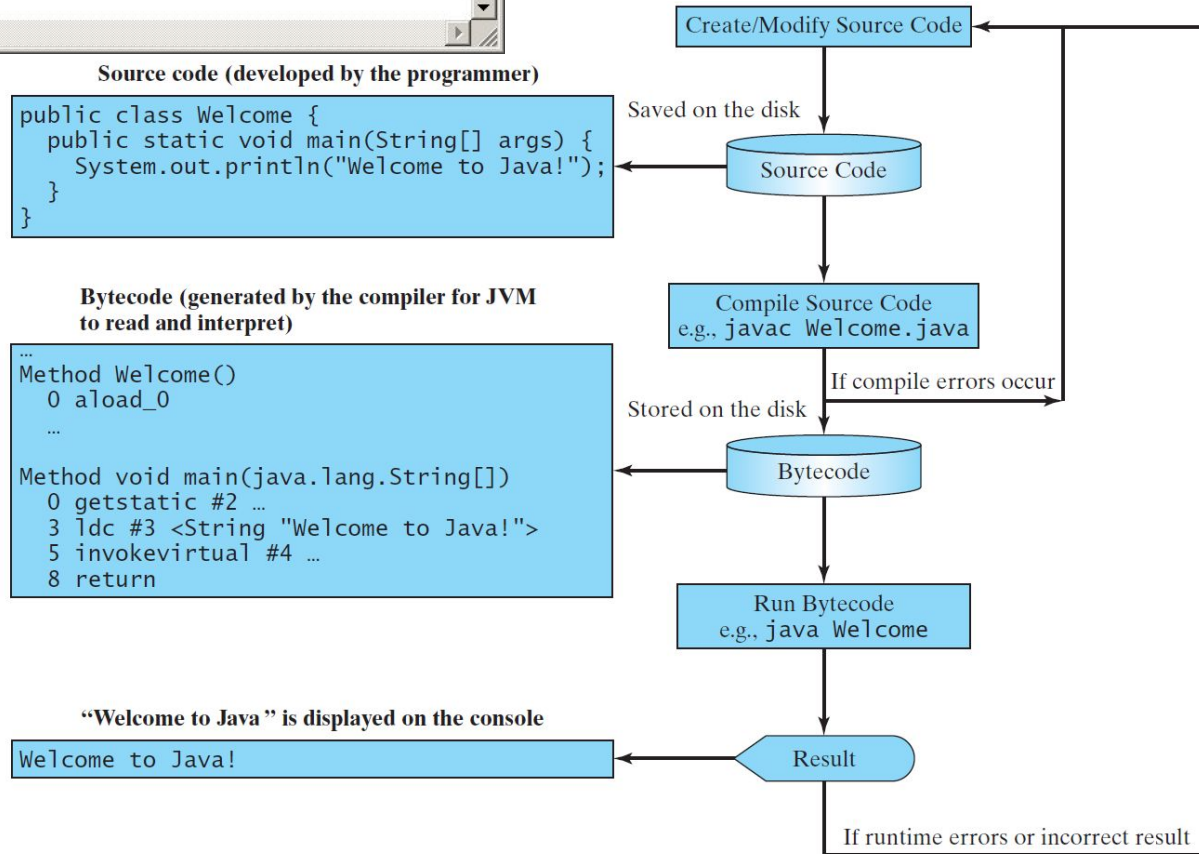
```
...
Method Welcome()
  0 aload_0
  ...

Method void main(java.lang.String[])
  0 getstatic #2 ...
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 ...
  8 return
```

"Welcome to Java" is displayed on the console

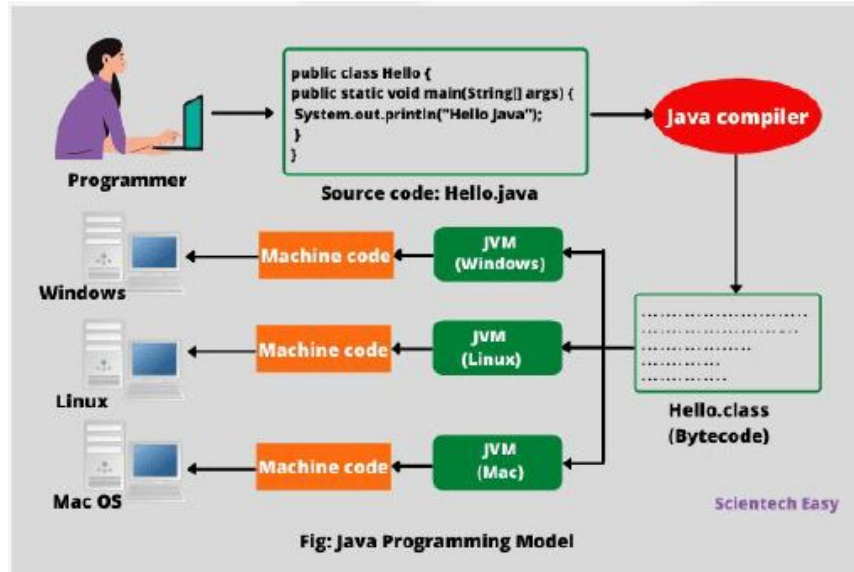
Welcome to Java!

Creación, compilación y ejecución de un programa Java muy sencillo



Compilando código fuente Java

Java fue diseñado para ejecutar programas objeto en cualquier plataforma. Con Java, escribes el programa una vez y compilas el programa fuente en un tipo especial de código objeto, conocido como bytecode. El bytecode puede ejecutarse en cualquier computadora con una Máquina Virtual Java, como se muestra a continuación. Java Virtual Machine es un software que interpreta el bytecode de Java.



Seguimiento de la ejecución de un programa

Inicia el método
principal

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Seguimiento de la ejecución de un programa

Ejecuta la sentencia

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

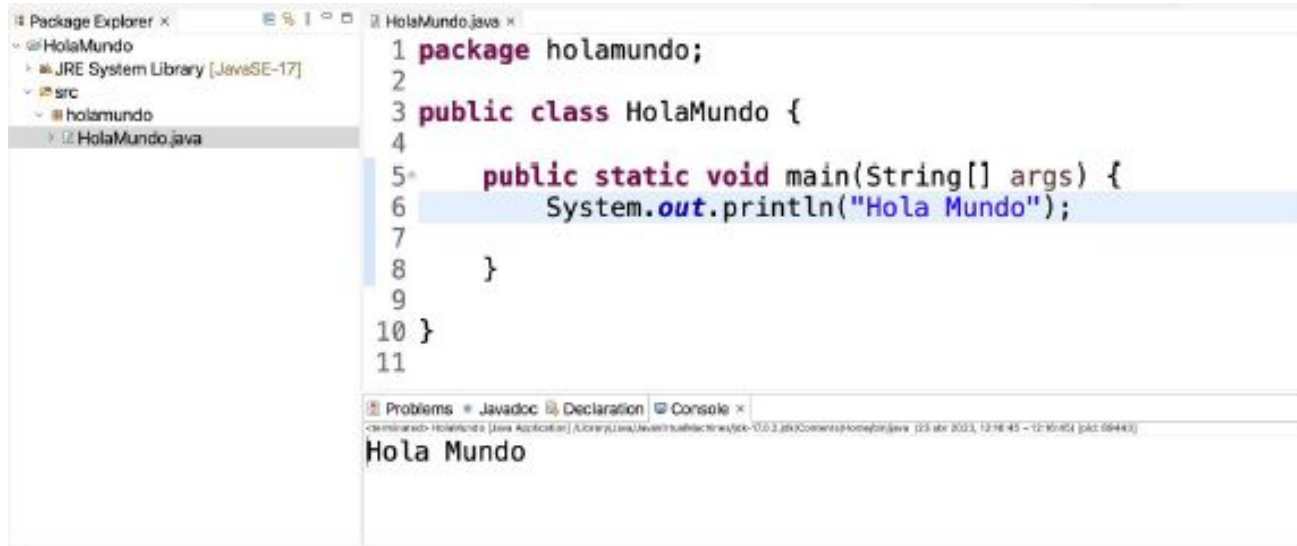
Seguimiento de la ejecución de un programa

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



imprime un mensaje en la
consola

Seguimiento de la ejecución de un programa



```
1 package holamundo;
2
3 public class HolaMundo {
4
5     public static void main(String[] args) {
6         System.out.println("Hola Mundo");
7     }
8 }
9
10 }
11
```

Hola Mundo

Anatomía de un programa Java

- Nombre de la clase
- Método principal
- Declaraciones
- Terminador de declaración
- Palabras reservadas
- Comentarios
- Bloques

Nombre de clase

Cada programa Java debe tener al menos una clase. Cada clase tiene un nombre. Por convención, los nombres de las clases comienzan con una letra mayúscula. En este ejemplo, el nombre de la clase es Welcome.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Método principal

La línea 2 define el método principal. Para ejecutar una clase, la clase debe contener un método llamado main. El programa se ejecuta desde el método principal.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Finalizador de instrucción

Una **sentencia** representa una acción o una secuencia de acciones.

La sentencia `System.out.println("Welcome to Java!")`.

Cada instrucción/declaración/sentencia en Java termina con un punto y coma (;).

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Palabras reservadas

Las palabras reservadas o las palabras clave son palabras que tienen un significado específico para el compilador y no pueden utilizarse para otros fines en el programa.

Por ejemplo, cuando el compilador ve la palabra **class**, entiende que la palabra después de **class** es el nombre de la clase.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Símbolos especiales

Carácter	Nombre	Descripción
{ }	Apertura y cierre de llaves.	Indica un bloque para incluir declaraciones.
()	Apertura y cierre de paréntesis.	Se utiliza con métodos.
[]	Corchetes de apertura y cierre.	Indica una matriz.
//	Doble barra	Precede una línea de comentario.
" "	Apertura y cierre de comillas	Incluyendo una cadena (es decir, secuencia de caracteres).
;	Punto y coma	Marca el final de una declaración.

Estilo de programación y documentación

- Comentarios apropiados

Incluya un resumen al principio del programa para explicar lo que hace el programa, sus características clave, sus estructuras de datos de apoyo y cualquier técnica única que use.

- Convenciones de nombres

Elija nombres significativos y descriptivos.

Nombres de las clases: Capitalizar la primera letra de cada palabra en el nombre. Por ejemplo, el nombre de clase `ComputeExpression`.

- Indentación adecuada y líneas de espaciado

- Indentar dos espacios.

- Utilizar una línea en blanco para separar segmentos del código.

Estilo de programación y documentación

- Estilos de bloques

Utilice el “end-of-line” para las llaves

*Next-line
style*

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

*End-of-line
style*

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

Errores de programación

- Errores de sintaxis: Detectado por el compilador

```
public class ShowSyntaxErrors {  
    public static main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

- Errores de tiempo de ejecución: Hace que el programa aborte

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        System.out.println(1 / 0);  
    }  
}
```

- Errores lógicos: Produce un resultado incorrecto

```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.println("35 grados Celsius son en Fahrenheit: ");  
        System.out.println((9 / 5) * 35 + 32); //se obtiene multiplicando la temperatura por 1,8 y sumando 32  
    }  
}
```

Formato de salida

Use la instrucción printf.

```
System.out.printf(format, items);
```

Donde format es una cadena que puede consistir en subcadenas y especificadores de formato.

Un especificador de formato determina cómo se debe mostrar un elemento. Un elemento puede ser un valor numérico, un carácter, un valor booleano o una cadena. Cada especificador comienza con un signo de porcentaje.

Los especificadores más usados

Specifier Output

%b un valor booleano

%c un caracter

%d un entero decimal

%f numero de punto flotante

%e un numero en notación científica

%s una cadena

Example

true or false

'a'

200

45.460000

4.556000e+01

"Java is cool"

```
int count = 5;  
double amount = 45.56;  
System.out.printf("count is %d and amount is %f", count, amount);
```

Diagram illustrating the mapping of variables to format specifiers in the printf statement:

- count is mapped to %d
- amount is mapped to %f
- The entire argument list (count, amount) is grouped under the label "items".

display

count is 5 and amount is 45.560000

Formato de salida: color del texto

```
public class TablaDeColores {  
    public static void main(String[] args) {  
        System.out.println(" ");  
        System.out.println(" | Código | Color | Código | Color | ");  
        System.out.println(" |-----|-----|-----|-----| ");  
        System.out.print(" | 30 | \033[30m negro \033[39;49m | ");  
        System.out.println(" | 90 | \033[90m negro claro \033[39;49m | ");  
        System.out.print(" | 31 | \033[31m rojo \033[39;49m | ");  
        System.out.println(" | 91 | \033[91m rojo claro \033[39;49m | ");  
        System.out.print(" | 32 | \033[32m verde \033[39;49m | ");  
        System.out.println(" | 92 | \033[92m verde claro \033[39;49m | ");  
        System.out.print(" | 33 | \033[33m amarillo \033[39;49m | ");  
        System.out.println(" | 93 | \033[93m amarillo claro \033[39;49m | ");  
        System.out.print(" | 34 | \033[34m azul \033[39;49m | ");  
        System.out.println(" | 94 | \033[94m azul claro \033[39;49m | ");  
        System.out.print(" | 35 | \033[35m morado \033[39;49m | ");  
        System.out.println(" | 95 | \033[95m morado claro \033[39;49m | ");  
        System.out.print(" | 36 | \033[36m cian \033[39;49m | ");  
        System.out.println(" | 96 | \033[96m cian claro \033[39;49m | ");  
        System.out.print(" | 37 | \033[37m blanco \033[39;49m | ");  
        System.out.println(" | 97 | \033[97m blanco claro \033[39;49m | ");  
        System.out.println(" |-----|-----|-----|-----| ");  
    }  
}
```

Formato de salida: color del texto

Usando la plantilla `\033[XXm`, basta cambiar XX por código correspondiente al color deseado. Estos códigos están establecidos en el estándar ANSI: https://en.wikipedia.org/wiki/ANSI_escape_code.

A terminal window titled "Terminal" with a dark purple background. It displays a table of ANSI color codes. The table has four columns: "Código", "Color", "Código", and "Color". The first two columns show codes 30-37 with their corresponding colors. The next two columns show codes 90-97, which are the same colors as the first two columns but with a lighter intensity. The text in the table is color-coded to match the descriptions.

Código	Color	Código	Color
30	negro	90	negro claro
31	rojo	91	rojo claro
32	verde	92	verde claro
33	amarillo	93	amarillo claro
34	azul	94	azul claro
35	morado	95	morado claro
36	cian	96	cian claro
37	blanco	97	blanco claro

Enlaces de interés

- Cursos/Tutoriales Java

- Aprende a Programar Java:

<https://aprendeaprogramar.com/cursos/verApartado.php?id=5001>

- Aprende Java con w3school:

<https://www.w3schools.com/java/default.asp>

- Tutorial Java

<https://guru99.es/java-tutorial/>

- Software

- Descarga JDK:

<https://www.oracle.com/java/technologies/downloads/?er=221886#java21>

- Instalar Eclipse:

<https://www.eclipse.org/downloads/packages/release/2024-06/r/eclipse-ide-java-developers>

<https://jarroba.com/instalar-bien-eclipse-un-ide-de-muchos/>