Programación

PLG
Planning and Learning Group

Universidad Carlos III de Madrid

Tema 3: Introdución a Java

En este tema

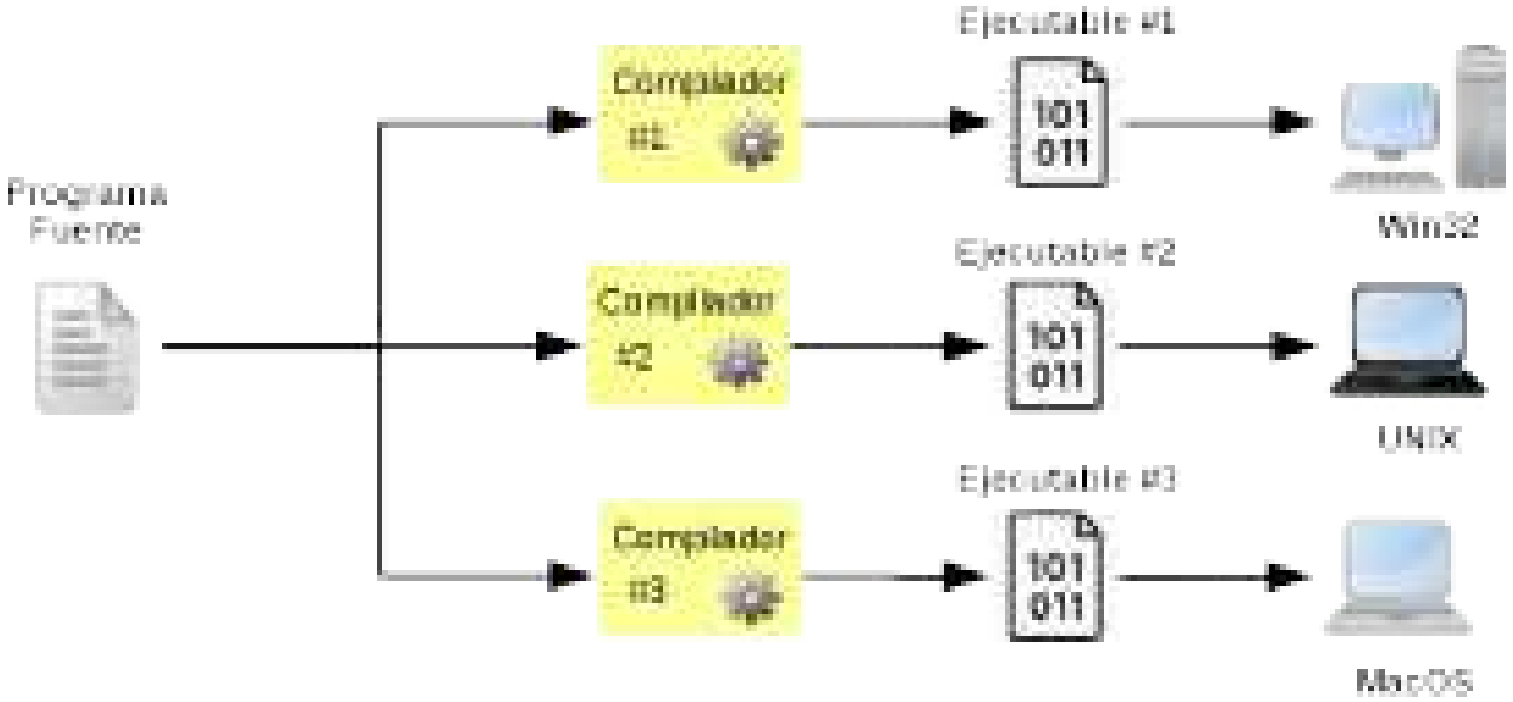
Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

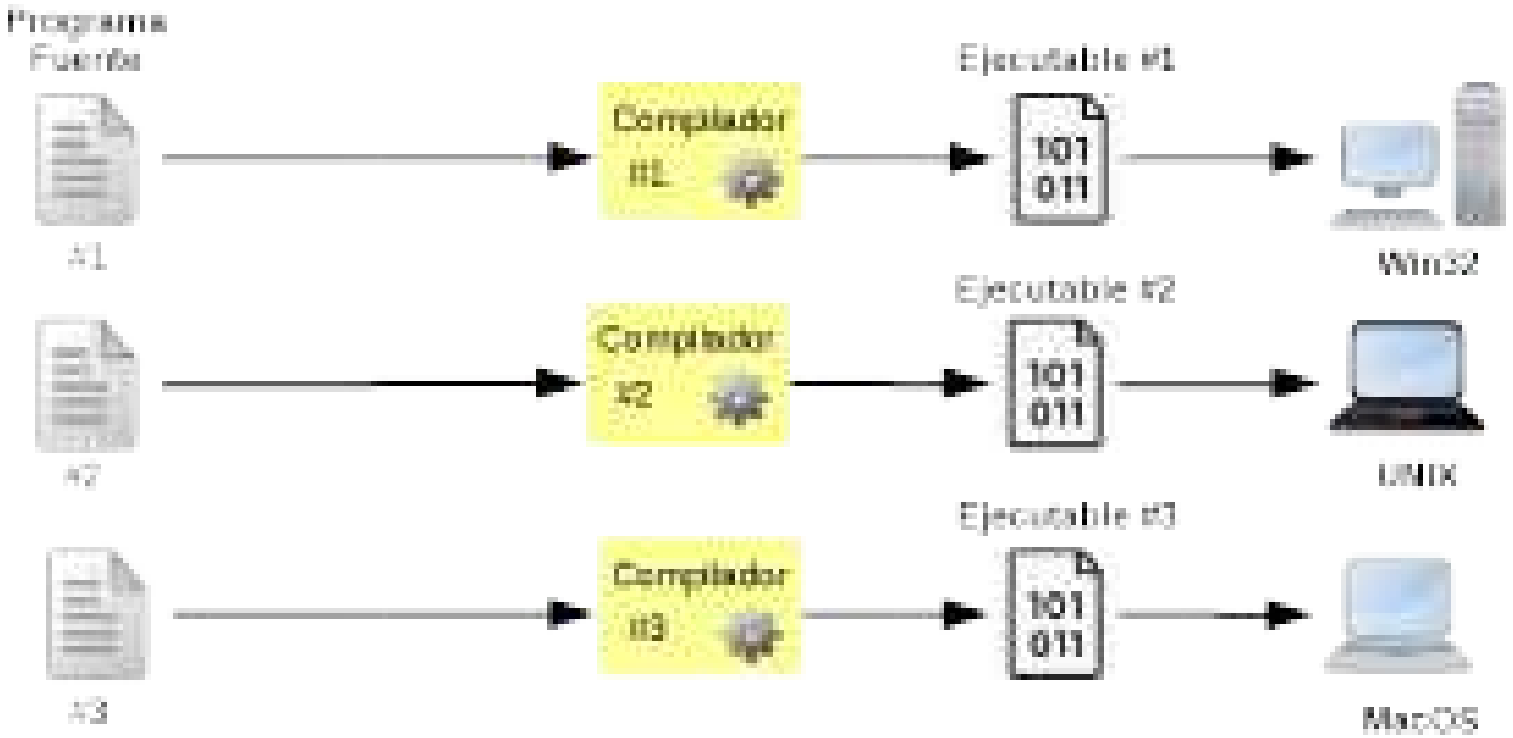
¿Qué es Java?

- Lenguaje de programación de alto nivel orientado a objetos
- Es también una plataforma de desarrollo
- 1991: Sun Microsystems diseña un lenguaje para sistemas embebidos, (set-top-boxes), electrodomésticos
  - Lenguaje sencillo, pequeño, neutro
  - Necesidad de un nuevo lenguaje: orientado a objetos, multiplataforma
  - Incialmente ninguna empresa muestra interés por el lenguaje
- Java: tipo de café?

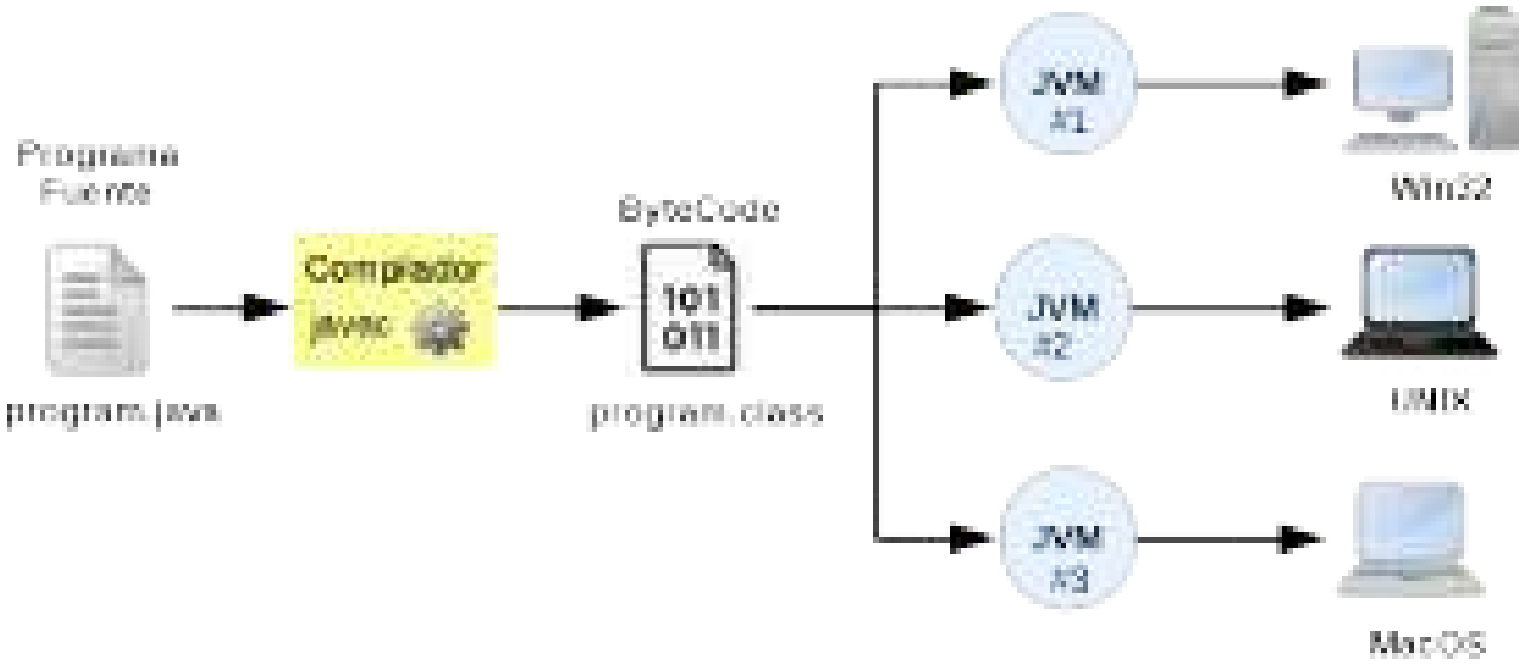
Historia de Java: otros lenguajes



Historia de Java: otros lenguajes



Historia de Java: Java



Historia de Java

- 1995: Java se presenta como lenguaje para Internet
- Netscape 2.0 introduce la primera JVM en un navegador web
- Filosofía Java: *Write once, run everywhere*
- 1997: Aparece Java 1.1. Muchas mejoras respecto a 1.0
- 1998: Java 1.2 (Java 2). Plataforma muy madura
- Apoyado por grandes empresas: IBM, Oracle, Inprise, Hewlett-Packard, Netscape, Sun
- 1999: Java Enterprise Edition. Revoluciona la programación en el lado servidor

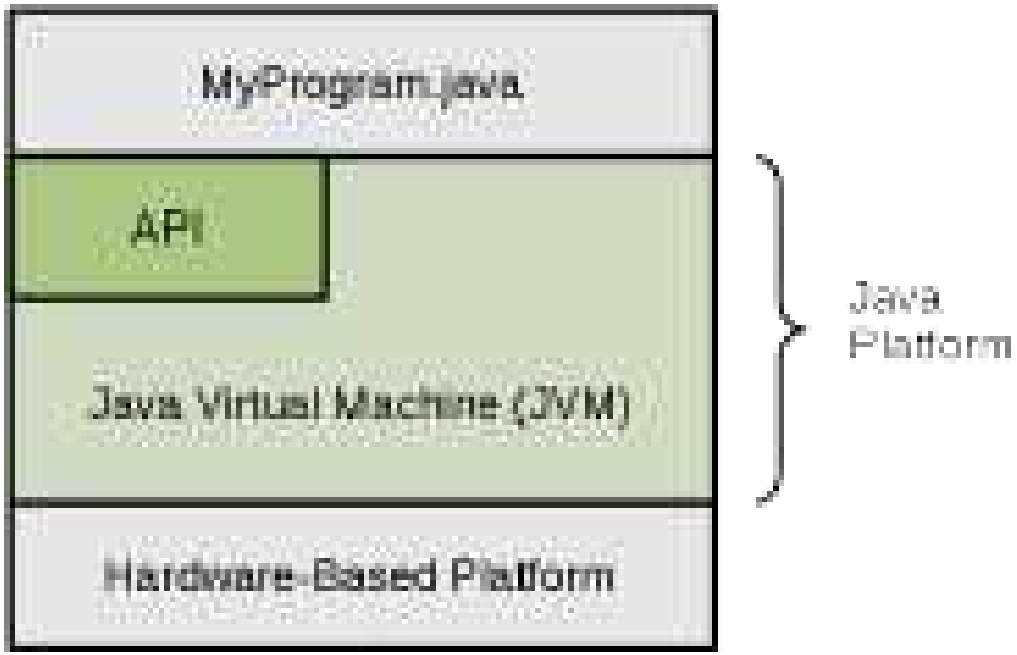
Características principales de Java

- Orientado a Objetos
- Totalmente Portable
- Lenguaje Interpretado (compilado a código intermedio, no a código máquina)
  - Java Virtual Machine (JVM)
  - ByteCode: Independiente de la máquina
- Gestión Automática de Memoria Dinámica. Recolector de basura (Garbage Collector)
- Sensible a Mayúsculas / Minúsculas
- Distribuido
- ¿Seguro?
- ¿Lento?

Versiones

- 1.0 (1996) – 1.1 (1997)- 1.2 (Java2) (1998) – 1.3 (2000) -1.4 (2002) – 1.5 (Java5.0) (2004) – Java 6 (2006) – Java 7 (2011) – Java 8 (Marzo-2014)
- Múltiples especificaciones
  - J8ME (Java 8 Micro Edition)
  - J8SE (Java 8 Standard Edition)
  - J8EE (Java 8 Enterprise Edition)

Plataforma Java



JDK (Java Development Kit)

- Compilador: javac
- Intérprete: java
- Plataforma de ejecución: JRE (Java Runtime Environment). Incluye JVM
- Plataforma de desarrollo: Java JDK (Java Software Development Kit):
  - Incluye Compilador, etc.
  - Incluye JRE

Entornos RAD (Rapid Application Development) o IDE (Integrated Development Environment)

- Productividad
- Modelado visual
- Depuración
- Rapidez de desarrollo
- Eclipse, Netbeans, Jbuilder, Symantec Café, Oracle Jdeveloper, Sun Java Workshop, IBM VisualAge, ...
- Prácticas:
  - J8SE (Java8 Standard Edition). Gratuito: `http://www.java.com/download`
  - Eclipse
    - Gratuito: `http://www.eclipse.org`
    - Versiones para Windows, Linux, etc.

En este tema

Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

Programa

Programa = datos + instrucciones

- Datos: `3.5`, `a`, `María`, ...
- Instrucciones: operan con los datos
  - Operadores: sumar, restar, etc.
  - Control de flujo: condicionales, bucles, etc.
  - Entrada/Salida (E/S): leer, escribir, imprimir datos

Variables

- Una variable es un espacio de memoria que se utiliza para guardar un dato
- Los datos son de un tipo
  - *3* es un número entero y *3.5* es un número real
  - *verdadero* es un valor lógico
  - *'a'* es un caracter
  - *María* es una palabra
- Una variable se define por un nombre y por el tipo del dato que almacena

Tipos básicos o primitivos en Java

- Números
  - Enteros: dependiendo del rango hay 4 subtipos
  - Reales: 2 subtipos
- Lógicos: 1 tipo
- Letras: 1 tipo

Java es un lenguaje fuertemente tipado: al declarar una variable hay que decir el tipo

Números enteros

	Tipo	Tamaño	Rango
Enteros	byte	8 bits	-128 a 127
	short	16 bits	-32.768 a 32.767
	int	32 bits	-2.147.483.648 a 2.147.483.647 ($\sim 2 * 10^9$)
	long	64 bits	-9.223.372.036.854.775.808L a 9.223.372.036.854.775.807L ($\sim 9 * 10^{18}$)

- Siempre con signo
- Rango independiente de la plataforma
- Por defecto son de tipo int
- Los *long* se escriben con una "L" al final: 989493849859L, -284829848L

Números Reales (coma flotante)

	Tipo	Tamaño	Rango
Reales	float	32 bits (1 signo, 8 exponente, 23 mantisa)	± 1.4E-45F ± 3,40282347E+38F
	double	64 bits (1 signo, 11 exponente, 52 mantisa)	±4.9E-324 ±1,79769313486231570E+308

- Por defecto son de tipo double, pero *float* es más rápido y ocupa menos memoria
- Los *float* se escriben con una “F” al final
  - 3.45E+21F
  - -284829848F
- Rango mayor que enteros. *double*: 15-16 cifras, *float*: 8-9 cifras. Más allá se trunca.
- Para notación científica se puede poner 22e-5 o 22E-5 (e$< x >$=$10^{<x>}$ por lo que 1e2=100)

Tipos lógicos

- Tipo: boolean
  - *true*
  - *false*

Caracteres

- Tipo: char (16 bits – 65536 caracteres)
  - Entre comillas simples: 'a'
  - En memoria se codifican numéricamente: *Unicode* ('a' = 97)
  - Caracteres especiales (secuencias de escape)

Caracteres especiales

Secuencia	Descripción
\b	Retroceso *(backspace)*
\t	Tabulador *(tab)*
\r	Retorno de carro *(return)*
\n	Nueva Línea *(line feed)*
\’	Comilla simple *(single quote)*
\”	Comilla doble (la del 2) *(double quote)*
\\	Barra invertida *(backslash)*

Cadenas de caracteres

- Son secuencias de caracteres o palabras
- NO son un tipo básico, se utiliza la clase String
- Pero se pueden tratar como datos de tipo básico
- Se denotan entre comillas: “hola”

En este tema

Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

Nuestro Primer Programa en Java

- Programas en Java
  - ficheros de texto con extensión .java (ej. `Hola.java`)
  - Hay que crear una clase y guardarla en el fichero. Una clase es un programa, y un programa es una clase.

Nuestro Primer Programa en Java

```
public class Hola
{
    public static void main(String[] args)
    {
        System.out.println("Mi primer programa en Java");
    }
}
```

Nuestro Primer Programa en Java

- El código va dentro del método main
- Primero los datos, luego las instrucciones.
- Las instrucciones terminan en “;”

En este tema

Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

Nombres e identificadores válidos

- Un identificador sirve para nombrar ficheros, variables, constantes, etc.
  - Un identificador empieza por _, $ o una letra.
  - y luego continua con _, o una letra o un número.
  - Ejemplos: `$var1, _var2, Variable`
- Sensible a las diferencias de mayúsculas y minúsculas
- No se pueden utilizar palabras reservadas
  class, public, static, int, float, true, . . .

Convenciones para identificadores

- Conjunto de recomendaciones para facilitar la compresión del código.
- CamelCase convención que escribe un conjunto de palabras unidas sin espacio y con la letra inicial de cada palabra en mayúscula.
  - Upper CamelCase: la primera letra en mayúscula
    `Num1, MiVariableContador`
  - Lower CamelCase: la primera letra en minúsculas
    `num1, miVariableContador`
- Los nombres de clases deben escribirse en Upper CamelCase
- Los nombres de variables deben escribirse en lower CamelCase

Uso de variables (Almacenamiento en Memoria)

- Declarar la variable
  - Reservar memoria, declarando el tipo
  - Dar un nombre a la dicha posición de memoria (identificador de la variable)

```
int numero1;
```

Uso de variables (Almacenamiento en Memoria)

- Declarar la variable
  - Reservar memoria, declarando el tipo
  - Dar un nombre a la dicha posición de memoria (identificador de la variable)

```
int numero1;
```

- Inicializar la variable
  - Dar el valor inicial del dato
  - Java no da valores por defecto a las variables

```
numero1 = 3;
```

Uso de variables (Almacenamiento en Memoria)

- Declarar la variable
  - Reservar memoria, declarando el tipo
  - Dar un nombre a la dicha posición de memoria (identificador de la variable)

```
int numero1;
```

- Inicializar la variable
  - Dar el valor inicial del dato
  - Java no da valores por defecto a las variables

```
numero1 = 3;
```

- Ambos pasos a la vez

```
int numero1 = 3;
```

Uso de variables (Almacenamiento en Memoria)

- Declaración de varias variables a la vez
  - separando por comas
  - algunas pueden estar inicializadas

```
int numero1 = 3, numero2 = 1, i, j;
float x, y, real1 = 3.5f;
```

Uso de constantes

- Las constantes son variables especiales: una vez les hemos asignado el valor, éste no puede cambiarse
- Se declaran igual que las variables pero con la palabra `final` delante
- Por convención se escriben en mayúsculas. Si son varias palabras se separan con _

```
final float PI = 3.141593f;
final int ALTURA_MAXIMA = 100;
```

Salida por pantalla

- Para mostrar en la pantalla el contenido de variables

```
int n = 5, m = 6;
System.out.println(n);
System.out.print(m);
System.out.println(n + " ," + m);
```

- También se puede imprimir un dato directamente

```
System.out.println(10.5);
```

- O concatenar varias variables o datos

```
int n = 5, m = 6;
System.out.println("Valores: " + n + " ," + m);
```

En este tema

Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

Instrucciones en Java

- Instrucciones
  - Operaciones
  - Control de flujo (más adelante)
  - Entrada/Salida
- Tipos de Operadores
  - de asignación
  - aritméticos
  - lógicos

Operador de Asignación

- El operador = lo utilizamos para copiar en una variable un dato, el contenido de otra variable o el resultado de una operación
- Visto en la inicialización de variables
- Nos permite cambiar los valores de las variables

```
int a = 4, b,c;
b = a;
a = 6;
```

Operadores Aritméticos

Operador	Descripción	int a=2, b=3, c	Valor en c
+	Suma	c = a + b	5
-	Resta	c = a - b	-1
*	Multiplicación	c = a * b	6
/	División	c = a / b	0
%	Módulo	c = a % b	3

Operadores Aritméticos

- El resultado de las operaciones con byte y short pasa a int automáticamente.
- Para el resto de tipos, el resultado de las operaciones es del mismo tipo

```
byte b1 = 5, b2 = 6;
int i1 = 3, i2 = 4, suma, producto;
double d1 = 3.0, d2 = 2, resultado;

suma = b1 + b2;
producto= i1 * i2;
resultado = d1/d2;

System.out.println(suma);
System.out.println(producto);
System.out.println(resultado);
```

Operadores de Autoincremento y Autodecremento

- Forma abreviada de sumar o restar 1 a una variable
  - `++` : suma uno a la variable
  - `--` : resta uno a la variable
- Prefijos: operador antes de la variable
  - se hace el incremento (o decremento) y se genera el valor
- Postfijos: operador después de la variable
  - se genera el valor y luego se hace el incremento (o decremento)

```
int i=0, j=0, x, y;
i++;
++j;
x = i++;
y = ++j;
```

Operadores de Autoincremento y Autodecremento

- Forma abreviada de sumar o restar 1 a una variable
  - `++` : suma uno a la variable
  - `--` : resta uno a la variable
- Prefijos: operador antes de la variable
  - se hace el incremento (o decremento) y se genera el valor
- Postfijos: operador después de la variable
  - se genera el valor y luego se hace el incremento (o decremento)

`int i=0, j=0, x, y;`	declaraciones
`i++;`	i vale 1
`++j;`	j vale 1
`x = i++;`	x vale 1, i vale 2
`y = ++j;`	y vale 2, j vale 2

Operaciones Aritméticas (char)

- Las operaciones artiméticas entre variables char pasan a int automáticamente.
- Los auto-incrementos o auto-decrementos con char mantienen el mismo tipo

```
char letra = 'a';
int letraNum;

letraNum = letra + 1;
letra++;
```

Operaciones Fuera de Rango

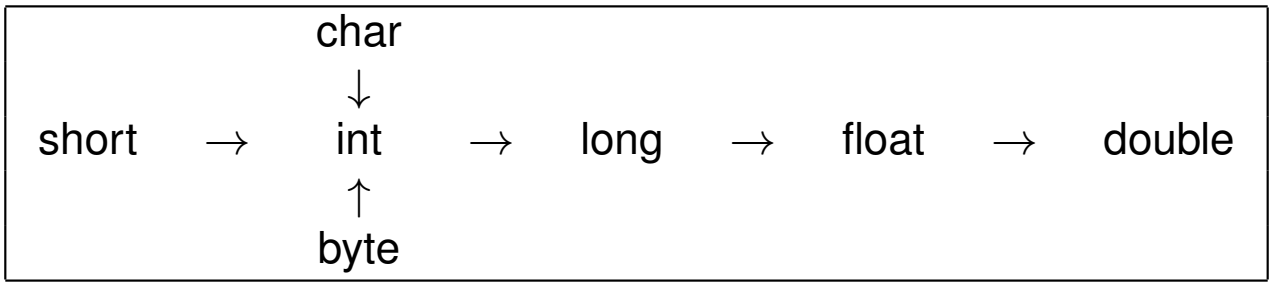
- ¿Qué ocurre si al operar nos salimos del rango?

Operaciones Fuera de Rango

- ¿Qué ocurre si al operar nos salimos del rango?
- Entre enteros
  - Resultado no coherente.
  - Desborda bits más significativos y error de signo
- Entre reales
  - Toma el mayor valor posible (`Infinity` o `-Infinity`)
- Casos Especiales
  - `Infinity`: Infinito positivo. Ejemplo `1/0`
  - `NaN`: Not a Number. Ejemplo Infinity-Infinity, Math.sqrt(-1).
  - No se puede asignar directamente `Infinity` o `NaN` a una variable

Mezcla de Tipos en Operaciones Aritméticas

- Reglas de compatibilidad
  - Todos los tipos numéricos son compatibles entre si. char también, pasa a int
  - boolean no es compatible con ningún otro tipo
  - String no es compatible con char ni con ningún otro tipo,
- Al realizar operaciones con tipos combinados, el resultado es del tipo de mayor capacidad.



Asignación de datos entre distintos tipos de variable

- Automático cuando:
  - Los tipos son compatibles, y
  - El tipo destino es mayor que el origen (no se pierde información)

Asignación de datos entre distintos tipos de variable

- Automático cuando:
  - Los tipos son compatibles, y
  - El tipo destino es mayor que el origen (no se pierde información)

```
int a = 3;
double d;
d = a;
```

Casting: conversión explícita entre tipos

- Cuando no se cumple lo anterior hay que forzar la conversión: Casting

```
double d = 3.5;
int a;
a = (int) d;
```

Casting: conversión explícita entre tipos

- Cuando no se cumple lo anterior hay que forzar la conversión: Casting

```
double d = 3.5;
int a;
a = (int) d;
```

- De real a entero: la parte decimal se trunca
- Si se sale de rango en `byte` y `short` resultado no coherente
- Se guarda el máximo valor posible

Operador de concatenación de *String*

- Operador + para concatenar `String`

```
String s1 = "Hola ";
String s2 = " mundo";

String s3 = s1 + s2 ;
String s4 = "Hola otra vez " + s2;
```

Operador de concatenación de *String*

- Operador + para concatenar `String`

```
String s1 = "Hola ";
String s2 = " mundo";

String s3 = s1 + s2 ;
String s4 = "Hola otra vez " + s2;
```

- Conversión automática si se opera con otro tipo

```
String s5 = "Hola otra vez " + 3;
```

Operadores relacionales

Operador	Descripción
==	Igual
!=	Distinto
>	Mayor
<	Menor
>=	Mayor ó igual
<=	Menor ó igual

```
3==5, 3.0 == 3
3!=5
'a' > 'c'
'a' < 'c'
'a' >= 60
'a' >= 60
```

Operadores relacionales

Operador	Descripción	
==	Igual	`3==5, 3.0 == 3`
!=	Distinto	`3!=5`
>	Mayor	`'a' > 'c'`
<	Menor	`'a' < 'c'`
>=	Mayor ó igual	`'a' >= 60`
<=	Menor ó igual	`'a' >= 60`

- Para tipos básicos (`boolean`, sólo `==` y `!=`)
- Resultado de tipo boolean
- Mezcla de tipos
- ¡Cuidado con la igualdad y los números reales!
- Reales especiales: `Infinity` == `Infinity`, pero `NaN` != `Nan`

Operadores lógicos

Operador	Descripción
&	AND
|	OR
^	XOR
!	NOT
&&	AND (evaluación perezosa – cortocircuito)
||	OR (evaluación perezosa – cortocircuito)

Operadores abreviados: operación + asignación

- El único operador de asignación es =
- Pero hay abreviaturas que permiten operar y asignar en una sola instrucción.

Operador	Descripción
+=	`a += <expresión>` significa `a = a + <expresión>`

Operadores abreviados: operación + asignación

- El único operador de asignación es =
- Pero hay abreviaturas que permiten operar y asignar en una sola instrucción.

Operador	Descripción
+=	`a += <expresión>` significa `a = a + <expresión>`

- Igual con otros operadores aritméticos: -=, *=, /=
- Y también lógicos: &=, |=, ^=

Precedencia de operadores

Se evalúan de izqda a dcha, excepto en *Asignación* que es de dcha a izqda

1. Paréntesis: ()
2. Unarios: !, $\sim$, ++, $--$, (*cast*)
3. Multiplicativos: $*$, /, %
4. Aditivos: +, $-$
5. Rotación: $>>$, $<<$
6. Relacional: $>$, $>=$, $<$, $<=$
7. Igualdad: ==, ! =
8. Lógico: &
9. Lógico: ˆ
10. Lógico: |
11. Lógico: &&
12. Lógico: ||
13. Condicional: ? :
14. Asignación: =, + = (*etc.*), & = (*etc.*)
15. Coma: ,

Precedencia de operadores

Se evalúan de izqda a dcha, excepto en *Asignación* que es de dcha a izqda

1. Paréntesis: ()
2. Unarios: !, $\sim$, ++, $--$, (*cast*)
3. Multiplicativos: $*$, /, %
4. Aditivos: +, $-$
5. Rotación: $>>$, $<<$
6. Relacional: $>$, $>=$, $<$, $<=$
7. Igualdad: ==, ! =
8. Lógico: &
9. Lógico: ˆ
10. Lógico: |
11. Lógico: &&
12. Lógico: ||
13. Condicional: ? :
14. Asignación: =, + = (*etc.*), & = (*etc.*)
15. Coma: ,

¡Usad paréntesis en caso de duda!

En este tema

Tema 3: Introdución a Java

- La JVM (*Java Virtual Machine*)
- Tipos básicos o tipos primitivos
- Creación de programas en Java
- Variables y Constantes
- Operadores
- Otras cuestiones: datos, comentarios y errores

Origen de los datos del programa

- Preprogramados
- Argumentos del programa
- Entrada *Standard*
- Fichero
- Aleatorios

Entrada *Standard* y fichero: clase `Scanner`

```
import java.util.Scanner; // se importa la clase Scanner


 class Ejemplo {


 public static void main(String arg[]){

      String variableString;

      Scanner entrada=new Scanner(System.in);

      System.out.print("Introduce un texto: ");

      variableString =  entrada.next();

      System.out.println("El texto es: "+variableString);

 }

}
```

Entrada *Standard* y fichero: clase `Scanner`

- Estos métodos leen un dato. Los datos se separan por espacios.
- Para leer un `String` se utiliza: `entrada.next()`
- Para leer enteros (`int`) se utiliza: `entrada.nextInt()`
- Existen más métodos de la clase `Scanner`

Entrada *Standard* y fichero: clase `Scanner`

- Estos métodos leen un dato. Los datos se separan por espacios.
- Para leer un `String` se utiliza: `entrada.next()`
- Para leer enteros (`int`) se utiliza: `entrada.nextInt()`
- Existen más métodos de la clase `Scanner`
- Ejemplo
  - Programa que lee un entero y un real en doble precisión (*double*) y los suma en un real en precisión simple (*float*)

Datos aleatorios

- Se utiliza `Math.random()`
- Este método devuelve un `double` mayor o igual que 0.0 y menor que 1.0

Comentarios

- El código **debe** tener comentarios
- Comentario de línea: //
- Comentario de bloque: /* ... */
- Comentario para *Javadoc*: /** ... */
- En la cabecera de cada fichero fuente se suele incluir:
  - Datos del creador
  - Versión
  - Fecha

Errores de programación

- Errores de compilación: por ejemplo errores de sintaxis, variables no declaradas/inicializadas, etc.,
- Errores de ejecución: por ejemplo memoria insuficiente, índices u operaciones fuera de rango, etc.
- No hay errores como tal, pero la lógica del programa es incorrecta