Universidad
Carlos III de Madrid

Tema 3

Introducción a Java

Programación
2015-2016

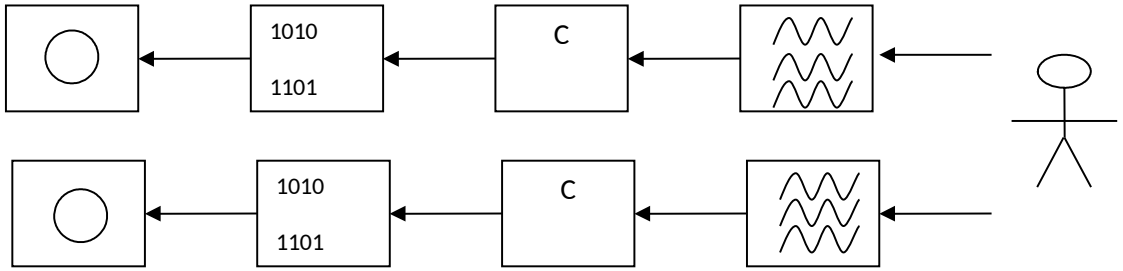
Agenda

- **Programación orientada a objetos: Java**
- Tipos de Datos
- Nuestro primer programa Java
- Operadores
- Resumen y Referencias

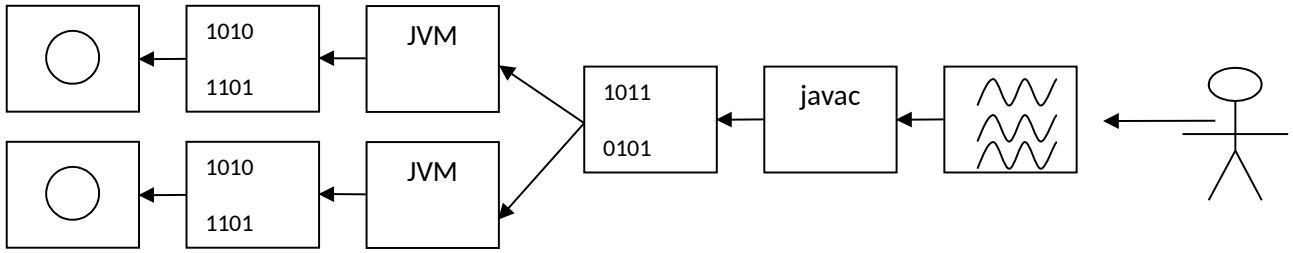
¿Qué es Java?

- 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
  - Ninguna empresa muestra interés por el lenguaje
- Java: tipo de café

Historia de Java (I)

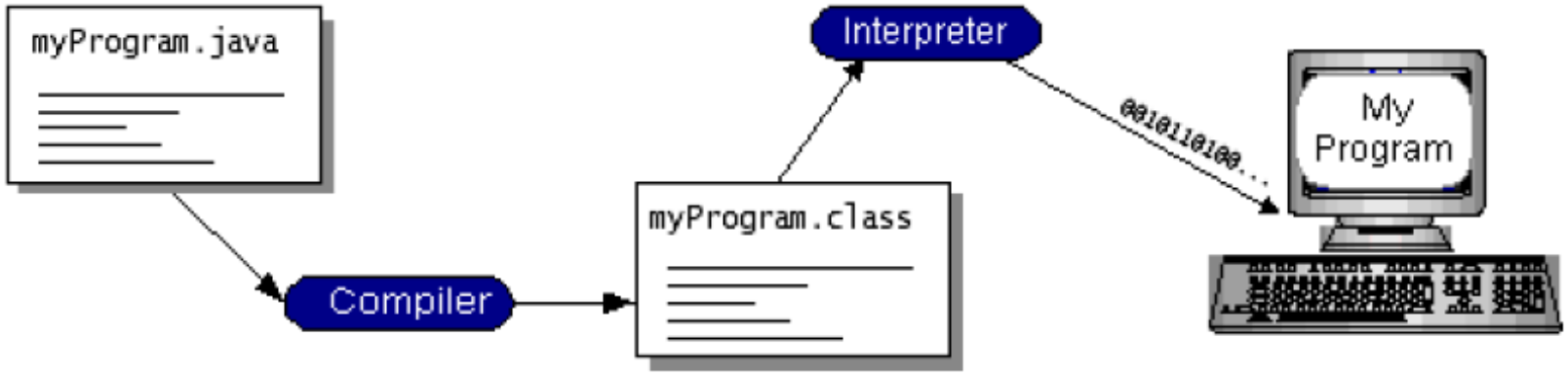


- Programas ligeramente distintos para distintas lavadoras
- Solución: lavadora virtual ␣ javalavadora
- Ganancia: un solo programa, aunque haya que hacer tres cosas (2 JVM y un compilador)



Compilado e interpretado

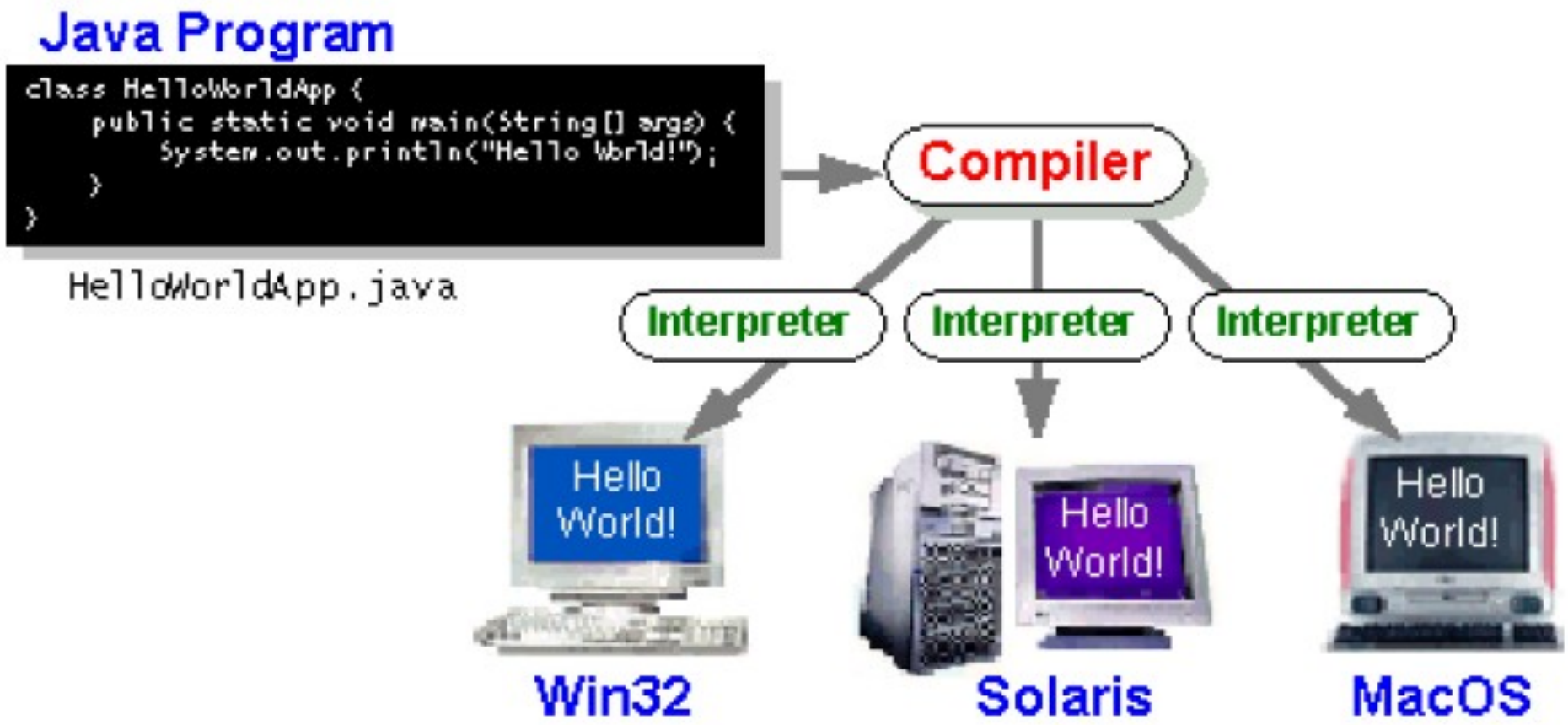
El lenguaje de programación Java es inusual por el hecho de que un programa a la vez se compila e interpreta



Con el compilador, un programa se traduce a un lenguaje intermedio llamado *Java bytecodes* —*estos códigos son independientes de la plataforma*— que será interpretado por el intérprete en la plataforma Java

Compilado e interpretado

"Escribir una vez, ejecutar en cualquier sitio". Un programa ***.java*** puede compilarse en cualquier plataforma que tenga un compilador Java. El fichero con los códigos de bytes ***.class*** puede entonces ejecutarse en cualquier implementación de la VM de Java.



Historia de Java (II)

- 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
- Robusto
- ¿Seguro?
- ¿Lento?

Versiones

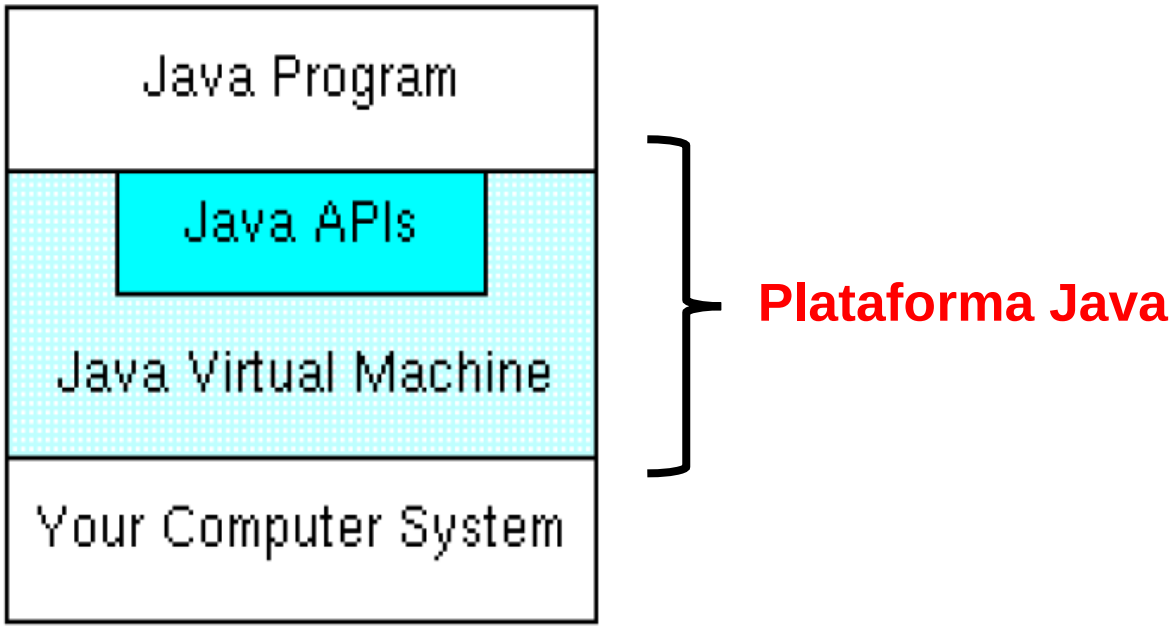
- 1.0 (1996) – 1.1 (1997)- 1.2 (Java2) (1998) – 1.3 (2000) -1.4 (2002) – 1.5 (Java5.0) (2004) – Java 6 (2006) – Java 7 (2011) – Java 8 (Marzo-2014)

- Múltiples Especificaciones:
  - J8ME (Java 8 Micro Edition)
  - J8SE (Java 8 Standard Edition)
  - J8EE (Java 8 Enterprise Edition)

Plataforma Java

La plataforma Java tiene dos componentes:

- La *Java Virtual Machine (Java VM)*
- La *Interfaz de Programación de Aplicaciones Java (Java API)*



Como muestra la figura, Java API y la máquina virtual (virtual machine) aíslan al programa del hardware

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

- Productividad
- Modelado visual
- Depuración
- Rapidez de desarrollo
- Eclipse, Netbeans, Jbuilder, Symantec Café, Oracle Jdeveloper, Sun Java Workshop, IBM VisualAge, ...

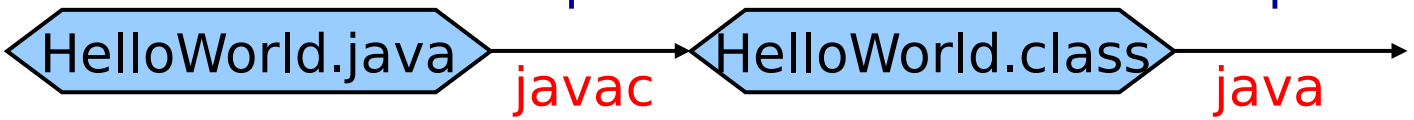
- Prácticas:
  - J8SE (Java8 Standard Edition)
    - Gratuito: http://www.java.com/download
  - Eclipse
    - Gratuito: http://www.eclipse.org
    - Versiones para Windows, Linux, etc.

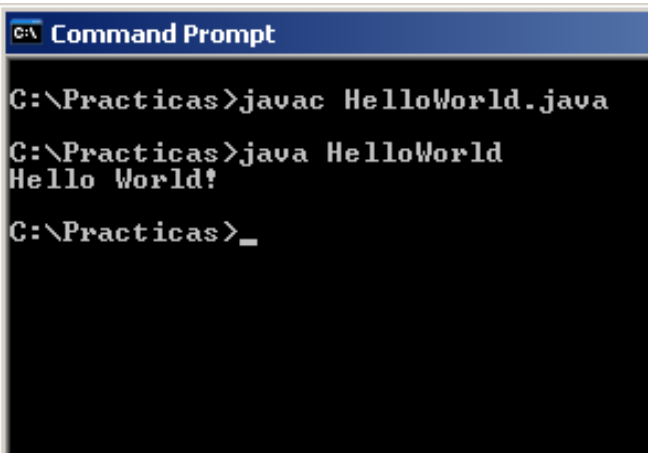
Ciclo de ejecución

Código Fuente

ByteCode

Ejecución





Programa = datos + instrucciones

- Instrucciones: operadores + control de flujo (control flow statements).
- Primero veremos tipos de datos, luego cómo se proporcionan a Java esos datos (cómo se ponen en la memoria), luego cómo operar con ellos y por último como ver el resultado.
- Ejemplo con un programa que suma dos números (en comparación con cómo se haría en una calculadora)

Agenda

- Programación orientada a objetos: Java
- **Tipos de Datos**
- Nuestro primer programa Java
- Operadores
- Resumen y Referencias

Tipos Básicos

Java es un lenguaje fuertemente tipado .

- Es necesario declarar el tipo de las variables.

Java dispone de ocho tipos básicos:

- Enteros. Son cuatro tipos para números enteros.
- Coma flotante. Son dos tipos para datos reales.
- Caracteres. Un tipo para caracteres de cualquier idioma mundial.
- Lógicos. Un tipo para los valores lógicos.

Enteros

- Los números enteros en Java son siempre con signo
- Tienen siempre las mismas capacidades de almacenamiento, independientemente del entorno.
- Por defecto las constantes enteras son de tipo int
- Se pueden poner guiones bajos para mejorar la legibilidad: 3_123 (Java 7)
- Para long hay que añadir una “L” al final

Nombre	Tamaño	Rango
long	64 bits	-9.233.372.036.854.775.808L a 9.233.372.036.854.775.807L
int	32 bits	-2.147.483.648 a 2.147.483.647
short	16 bits	-32.768 a 32767
byte	8 bits	-128 a 127

Coma Flotante

- Dos tipos: *float* y *double*
- Por defecto son de tipo *double*.
- Para *float* se añade una “F” al final del número.
- Hay tres valores especiales: infinito positivo Infinity, infinito negativo -Infinity y NaN (Not a number)

Nombre	Tamaño	Rango
float	32 bits	± 3.40282347E+38F
double	64 bits	± 1.79769313486231570E+308

- El número puede tener 15-16 cifras como máximo para el double (2^52) y 8-9 para el float (2^23), más allá se descarta (se trunca).

Lógicos

- Para valores lógicos: el tipo *boolean*
- Sólo toma dos valores: verdadero *true* y falso *false*
- Se emplea en las estructuras condicionales
- Resulta en operaciones con operadores relacionales
- Es distinto de los demás e incompatible con el resto
- Ejemplo:

```
boolean a;
a=true;
if (a)
   System.out.println("Es verdadero");
```

Caracteres

- Los caracteres se codifican en UNICODE ocupando 16 bits (65536 caracteres).
- Se pueden representar los caracteres de cualquier lenguaje.
- Se representan entre comillas simples. 'A' '\101'
- Unicode '\u0041' Hexadecimal '\0x41'
- Secuencias de escape:

Secuencia	Descripción
\b	Retroceso
\t	Tabulador
\r	Retorno de carro
\n	Nueva línea
\’	Comilla simple
\"	Comilla doble
\\	Barra invertida

Cadenas de caracteres

- En Java **no** hay un tipo básico para cadenas de caracteres.
- Se utiliza la clase *String*
- Se escriben entre comillas dobles ”
  (Todo en la misma línea, no se pueden cortar).
- Se pueden concatenar con el operador +
- Se pueden usar las secuencias de escape con String.
- Se pueden declarar constantes de tipo String
- Ejemplo:

```
String a,b;
a="Buenos";
b=" días";
String c=a+b;
System.out.println(c);
```

Agenda

- Programación orientada a objetos: Java
- Tipos de Datos
- **Nuestro primer programa Java**
- Operadores
- Resumen y Referencias

Creando nuestro primer programa en Java

- Dar el nombre al programa
- Guardar los datos en la memoria
- Mostrar por pantalla

Dar nombre y crear el fichero

- Hay que crear una clase y guardarla en un fichero.
  - Un programa es una clase y una clase es un programa.
- Nombres válidos

Identificador empieza por:	**Continúa por:**
_ (guión bajo) $ (símbolo del dólar) Letra	_ (guión bajo) $ (símbolo del dólar) letra número

- No se pueden usar palabras protegidas (int…, true, class, public)
- Se recomienda usar nombres explicativos, ni muy largos ni muy cortos.
  - El fichero se llama <nombreClase>.java
  - ¡Mayúsculas y minúsculas!
- Por convención el nombre de un programa debería empezar por mayúsculas.
- Todo el código va dentro del método **`main`**, primero los datos y luego las instrucciones.

Programa Java

```
public class Ejemplo {

   public static void main(String[] args) {
      System.out.println(“Ejemplo”);
   }

}
```

Variables

- Se usan para guardar datos e información.
- En Java antes de usarlas es preciso declararlas.

  tipo identificador[=valor][,identificador[=valor]...];

- Ejemplo:

```
int var = 3456, otravar = 2143;
double db = 9876.34;
float fl = 876.234F;
```

Votación

S2-Votacion 0: Declaracion de variables

Variables

- Ámbito es el bloque en el que está declarada
- Bloque: porción de código delimitado por dos llaves ({ y })

```
1. {
2.    int a;
3.    a=9;
4.                {
5.             int b=a+1;
6.                }
7.       a=10;
8. }
```

- Tiempo de vida de una variable es el tiempo (código) que va de la declaración de la variable hasta su destrucción

Inicialización de variables

- Las variables locales no son inicializadas por el compilador. Por lo tanto es nuestra obligación inicializarlas, de lo contrario el compilador visualizará un mensaje de error en todas las sentencias que hagan referencia a esas variables.

- En cambio las variables miembro de una clase son inicializadas por omisión por el compilador para cada objeto que se declare de la misma.
  - números enteros: 0
  - números reales: 0.0
  - booleanos: false;
  - caracteres: \u0000 // carácter nulo
  - referencias: null

- También si queremos pueden ser inicializadas explícitamente.

Mostrar por pantalla

- Para comprobar que la memoria guarda lo que hemos puesto, lo imprimimos por pantalla:
  - System.out.println(variable);
  - System.out.print(variable);
- También se puede imprimir directamente un dato:
  - System.out.println(23);

Constantes

- Se usan para declarar “variables” con un valor inmutable.

```
final int maximo= 100;
final float velocidadLuz= 300000.0;
```

- El valor de las constantes se proporciona en la declaración o la primera vez que se asignan

- Error: Si son atributos de clase, tienen valor por defecto

```
final int numAlumnos; // ¡fija el valor a 0!
numAlumnos= 540; // no se puede cambiar
```

Identificadores

- Se utilizan para nombrar cualquier cosa que el programador necesite usar o identificar: clases, métodos y variables.
- Un identificador es válido si comienza por:
  - Una letra
  - Un subrayado (_)
  - Símbolo de dólar ($)
- Los siguientes caracteres pueden ser letras, dígitos y caracteres subrayado y dólar (no hay longitud máxima)

  NOTA: distingue entre letras mayúsculas y minúsculas

Ejemplos Válidos:
MiPrograma Mi_Programa
_miprograma $mi_programa
mi_primer_programa_en_java

Ejemplos Incorrectos:
2Programas Mi-Programa
“miprograma” $mi/programa
mi primer programa en java

Agenda

- Programación orientada a objetos: Java
- Tipos de Datos
- Nuestro primer programa Java
- **Operadores**
- Resumen y Referencias

Operadores Aritméticos

```
public class Operadores {
    public static void main(String [] args){
        int a=9;
        int b=9;
        System.out.println(a++);
        System.out.println(++b);
    }
}
```

```
9
10
```

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
++	Incremento
- -	Decremento

Votación

S2-Votacion 1: Operadores aritméticos

Conversión de tipos

Cuando en una expresión se mezclan datos con distintos tipos, se realizan conversiones de tipo

Cuando es posible se realiza de forma automática.

```
char c = 'a';
int i = c;              // CORRECTO
short s = c;            // INCORRECTO, por el signo de s
s = 678;
```

En otros casos, el programador puede forzar la conversión: casting

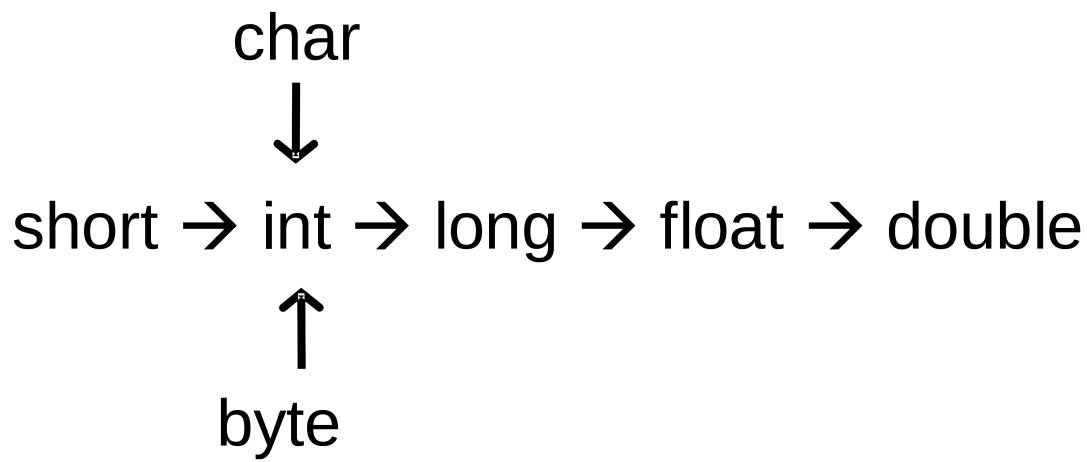
```
double db = 123.67;
int destino = (int) db;     // CORRECTO, lo trunca
```

Conversión automática

Los dos tipos son compatibles.

- Todos los tipos numéricos son compatibles entre si, sin importar que sean enteros o reales.
- El tipo *char* es compatible con *int*.
- El tipo *boolean* no es compatible con ningún otro tipo.

El tipo destino es más grande que el origen



Conversión explícita. Casting

La conversión sigue unas reglas:

- Entre números enteros, si el destino es menor que el origen, el valor resultado será el resto (módulo) de la división entera del valor entre el rango del destino.
- Si el origen es un número real y el destino un entero, la parte decimal se trunca, y si la parte entera no cabe en el destino, se aplica en criterio del módulo.
- Entre números reales, se guarda el máximo valor posible.

Ejercicio

S2-Clase: Variables y aritméticos

Promoción en expresiones

Al evaluar una expresión, se produce la conversión de tipos necesaria.

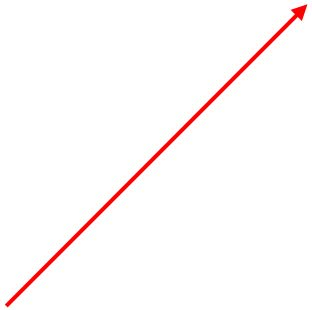
Java convierte los operandos al tipo mayor de los de la expresión y después evalúa la expresión

Las reglas que se aplican son las siguientes:

- ***byte*** y ***short*** se promocionan a ***int***.
- Si un valor es ***long*** la expresión se promociona a ***long***.
- Si un valor es ***float*** la expresión se promociona a ***float***.
- Si un valor es ***double*** la expresión se promociona a ***double***.
- Un ***char*** en una expresión numérica se promociona a ***int***.

Promoción

```
Class Promocionar{
  public static void main (String args[]){
    byte b = 42;
    char c = 'a';
    short s = 1024;
    int i = 200;
    float f = 5.67F;
    double d = .2324;
    double result =          (f*b)+(i/c)-(d*s);
    System.out.println("resultado = "+result);
  }
}
```



Promociones

f*b → b promociona a float
i/c → c promociona a int
d*s → s promociona a double
(float)+(int) → promociona a float
(float)-(double) → promociona a double
result → es de tipo double

Votación

S3-Votacion 2: Casting

Operaciones con String

Operador + para cadenas String

Podemos convertir las cosas a String sumándoselas a una variable String:

```
char c='a'; int b=3;
String cad2= "hola "+c+b;
System.out.println(cad2);
```

Hay una forma más ortodoxa ya la veremos: String.valueOf(tipo)

Operadores Relacionales

Sirven para hacer comparaciones.

El resultado es tipo boolean

Para comparar Strings hay que usar equals

```
public class OperRelac {
public static void main(String [] args){
  String s=”cadena”;
  String p=”otra cadena”;
  boolean res = "cadena".equals(s);  // True
  boolean res2 = s.equals(p);        // False
  System.out.println(res);
  System.out.println(res2);
  }
}
```

```
true
false
```

Operador	Descripción
==	Igual
!=	Distinto
>	Mayor que
<	Menor que
>=	Mayor o igual
<=	Menor o igual

Votación

S3-Votacion 3: Operadores Relacionales

Operadores Lógicos

Se aplican sobre operandos *boolean*

Como resultado se obtiene un valor *boolean.*

En cortocircuito, la expresión se evaluará hasta que se conozca el valor seguro del resultado

Operador	Descripción
&	AND
|	OR
^	XOR
&&	AND en cortocircuito
||	OR en cortocircuito
!	NOT

Votación

S3-Votacion 4: Operadores Lógicos

Operadores de Asignación

- Asigna la expresión de la derecha a la variable de la izquierda
- Existe la posibilidad de escribir en forma reducida operaciones

Operador	Descripción
+=	Suma y asignación
-=	Resta y asignación
*=	Multiplicación y asignación
/=	División y asignación
%=	Módulo y asignación
&=	AND y asignación
|=	OR y asignación
^=	XOR y asignación

Operadores de Asignación

```
public class OperadoresDeAsignacion {
   public static void main(String[] args) {
      int x=5,y=2,z=-3;
      y+=x;
      System.out.println("x es:" +x);
      System.out.println("y es:" +y);
      z-=x;
      System.out.println("z es:" +z);
      y/=x;
      System.out.println("y es:" +y);
      y%=x;
      System.out.println("y es:" +y);
   }
}
```

```
x es:5
y es:7
z es:-8
y es:1
y es:1
```

Paréntesis

Usamos paréntesis cuando:

- Exista ambigüedad sobre qué operador aplicar antes que otro
- Se quiera dar más precedencia a unos operadores que a otros
- Se quiera hacer el código más legible/ entendible

```
public class Parentesis {
    public static void main(String[] args) {
        int a = 5 * 3 + 2;
        int b = (5 * 3) + 2;
        int c = 5 * (3 + 2);
        int d = 5 % 5 + 1;
        int e = 5 % (5 + 1);
        int f = (5 % 5) + 1;
        int g = 2 + (2 * (2 +1));
        System.out.println("a=" + a + ", b=" + b + ", c=" + c);
        System.out.println("d=" + d + ", e=" + e + ", f=" + f);
        System.out.println("g=" + g);
    }
}
```

Precedencia

()
! ~ ++ -- (delante)
(cast)
** / %*
+ -
>> << >>>
> >= <= > instanceof
== !=
&
^
|
&&
||
Ternario
asignación

Ejemplos de utilización de expresiones

1. Escribir una expresión en java para determinar si un carácter es alfanumérico o no

2. Escribir una expresión en java para determinar si un año es bisiesto o no (múltiplo de 4, no múltiplo de 100, pero sí múltiplo de 400)

3. Escribir expresiones que a partir de un tiempo transcurrido en segundos escriban el número de días, horas, minutos y segundos.

Ejemplos de utilización de expresiones. Solución

1. Escribir una expresión en java para determinar si un carácter es alfanumérico o no

```
boolean alfan = (c>='a'&&c<='z')||(c>='A'&&c<='Z')||(c>='0'&&c<='9');
```

2. Escribir una expresión en java para determinar si un año es bisiesto o no (múltiplo de 4, no múltiplo de 100, pero sí múltiplo de 400)

```
boolean bisiesto= a%4==0&&(a%100!=0||a%400==0);
```

3. Escribir expresiones que a partir de un tiempo transcurrido en segundos escriban el número de días, horas, minutos y segundos.

```
int segundos=(s%60);
int minutos=(s%(3600))/60;
int horas=(s%(3600*24))/3600;
int dias=s/(3600*24);
```

Ejemplo precedencia operadores

```
class EjOperadores1{
        public static void main (String [] args){
            int a=5,b=2,s;
            System.out.println("a es:"+a);
            System.out.println("b es:"+b);
            // evaluar la siguiente expresión
            s = a+++b*5%3+a;
            System.out.println("a es:"+a);
            System.out.println("b es:"+b);
            System.out.println("a+++b*5%3+a="+s);
            }
}
```

```
a es:5
b es:2
a es:6
b es:2
a+++b*5%3+a=12
```

Salida: a+++b*5%3+a=12
equivale a: (a++) + ((b*5) % 3) + a
// resultado 12, porque la segunda a se ha incrementado
// cuando se usa su valor

Comentarios

Es muy importante **comentar bien el código**:

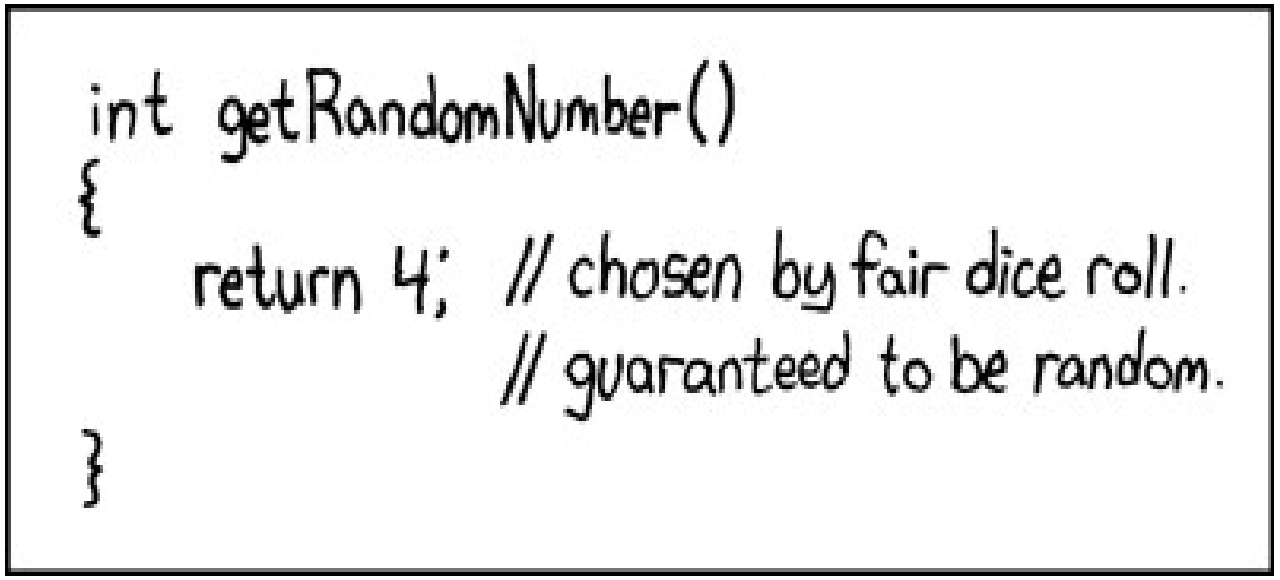
- Hace el código **legible y entendible**
- Aunque ahora sepamos perfectamente lo que hace, **puede que dentro de años volvamos a utilizarlo**.
- Puede que **otros programadores** reutilicen nuestro código y necesitan entenderlo
- Es una buena práctica introducir un **comentario al principio de cada archivo** con la información principal sobre éste

Comentarios

En java, tres formas de escribir comentarios:

- Comentarios de **una sola línea**:
  - Se utilizan los caracteres “//”
  - Todo lo que se escriba **a la derecha** es un comentario, será ignorado por el compilador
- Comentarios de **varias líneas**:
  - Se utilizan los caracteres “/*” para el principio del comentario, y “*/” para el final
  - Todo lo que se escriba **dentro** es un comentario, será ignorado por el compilador
- Comentario para descripción del elemento, generando automáticamente una documentación cuando se utiliza **javadoc**
  - ***/**** comentario de documentación, una o más líneas ***/***

Generación de datos aleatorios



http://xkcd.com/221/

Números aleatorios

- Método interesante de la clase Math:

```
static double random()
```

- Devuelve un valor aleatorio **double** mayor o igual que 0.0 y menor que 1.0

- Por ejemplo para simular el lanzamiento de un dado (buscamos un entero del 1 al 6)

```
Math.round(Math.random()*5)+1
```

Entrada por teclado

A partir de JDK 5.0 aparece la clase Scanner para obtener valores introducidos mediante el teclado.

```
import java.util.Scanner;
...
Scanner lee = new Scanner(System.in);
int b=0;
b= lee.nextInt();
System.out.println(b);
```

```
nextBoolean(): obtiene un boolean
nextByte(): obtiene un byte
nextShort(): obtiene un short
nextInt(): obtiene un número entero
nextLong(): obtiene un número entero long
nextFloat(): obtiene un número real float
nextDouble(): obtiene un número real double
next(): obtiene una cadena de caracteres
```

Ejercicio

S3-Clase: Casting y operadores

Agenda

- ¿Qué es programar?
- Arquitectura básica de un ordenador
- Breve introducción histórica a la programación
- Compilación vs. interpretación de programas
- Paradigmas de programación
- Programación orientada a objetos: Java
- **Resumen y Referencias**

Resumen

- Programación Orientada a Objetos: Java
  - Qué es Java - Historia
  - Características principales
- Tipos de Datos
  - Tipos Básicos
  - Cadenas de Caracteres
- Programando en Java
  - Variables
  - Conversión de Tipos

- Operadores
  - Aritméticos
  - Relacionales
  - Lógicos
  - De asignación
  - Paréntesis
- Precedencia de Operadores
- Comentarios
- Datos aleatorios
- Entrada por teclado
- Resumen

Bibliografía y referencias web

- Variables:
  - https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
- Operadores
  - https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html