



## TEMA 2

# Fundamentos de JAVA

V1.3

Manuel Pereira González



## Agenda

- **Introducción**
  - Historia de Java
  - Características Principales
  - Hello World
- Tipos
- Operadores
- Control de Flujo
- E/S básica
- Atributos y Métodos
- Resumen



# Introducción: Historia de Java



- Sun Microsystems (Patrick Naughton y James Gosling)
- **Java**: Marca de Café
- Necesaria portabilidad de código
- Inicialmente orientado a Internet
- Impulsado por compatibilidad con Netscape Navigator (navegador de Internet)

# Introducción: Historia de Java



- Versiones
  - 1996: Java v1.0
  - 1998: Java2 (v1.2). Gran paso adelante.
  - Actual: J2SE v1.6
- Prácticas
  - J2SE (Java 2 Standard Edition) v1.5
    - Gratuito: <http://java.sun.com>
  - Eclipse v3.1.1
    - Gratuito: <http://www.eclipse.org>
    - Versiones para Windows, Linux, etc.

## Introducción: Características Principales



- Orientado a Objetos
- Totalmente Portable
- Lenguaje Interpretado (compilado a código intermedio, no a código máquina)
  - Java Virtual Machine (**JVM**)
  - ByteCode: Independiente de la maquina
- Gestión Automática de Memoria Dinámica
  - Recolector de basura (Garbage Collector)
- Case Sensitive (Sensible a Mayús. / Minus.)

## Introducción: Características Principales



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

# Introducción: Hello World



```
/**
 * Nombre:      HelloWorld.java
 * Descripción: Esta es mi primera clase escrita en Java
 * Autor:       Manuel Pereira
 */

public class HelloWorld {

    /**
     * @param args Argumentos recibidos por línea de parámetros
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

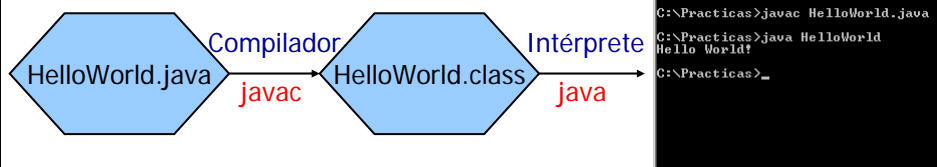
# Introducción: Hello World



Código Fuente

ByteCode

Ejecución



# Agenda



- Introducción
- **Tipos**
  - Tipos Básicos
  - Variables
  - Conversión de tipos
  - Cadenas de Caracteres
  - Arrays
- Operadores
- Control de Flujo
- E/S básica
- Atributos y Métodos
- Resumen



## Tipos: Tipos Básicos



- Enteros
  - long
  - int
  - short
  - byte
- Coma Flotante
  - float
  - double
- Caracteres
- Lógicos

TIPO	TAMAÑO	EJ:
long	64 bits	-85738593L , 8593854L
int	32 bits	-28392858 , 592934
short	16 bits	-30000 , 8438 , -4923
byte	8 bits	-32 , 123 , 39
float	32 bits	-3.56E+30F , 8.234
double	64 bits	-2.49E+300 , 3.95E+200
char	16 bits	'a', 'D', '\n', '\\', '\\''
boolean	1 bit	true , false

## Tipos: Tipos Básicos: Enteros



- Enteros
  - Siempre **con signo**
  - Cuatro tipos: **byte**, **short**, **int**, **long**
  - Rango independiente de la plataforma
  - Enteros por defecto son tipo "**int**"
  - Para long añadir "**L**" al final
    - 989493849859L
    - -284829848L

## Tipos: Tipos Básicos: Coma Flotante



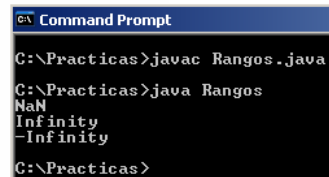
- Coma Flotante
  - Dos tipos: **float**, **double**
  - Flotantes por defecto son tipo "**double**"
  - Para float añadir "**F**" al final
    - 3.45E+21F
    - -284829848F

## Tipos: Tipos Básicos: Coma Flotante



- Valores especiales de **float** y **double**: Infinity, -Infinity, NaN (Not a Number)

```
/**
 * Nombre: Rangos.java
 * Descripción: Muestra valores especiales de números en coma flotante
 * Autor: Manuel Pereira
 */
public class Rangos {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        System.out.println(Math.sqrt(-1));
        System.out.println(1.1e200*1.1e200);
        System.out.println(-1.1e200*1.1e200);
    }
}
```



```
C:\Practicas>javac Rangos.java
C:\Practicas>java Rangos
NaN
Infinity
-Infinity
C:\Practicas>
```

## Tipos: Tipos Básicos: Caracteres



- Caracteres
  - 16 bits -> **UNICODE**
  - Entre comillas simples: 'a', 'A', 'b'
  - Secuencias de escape: '\b', '\t', '\r', '\n', '\\', '\"', '\'', '\'
  - Mediante código unicode: '\u0041'
  - Hexadecimal: '\x41'

## Tipos: Tipos Básicos: Envoltorios



- En Java, todo excepto los tipos básicos son clases y objetos (heredan de Object)
- Existen objetos que envuelven a los tipos básicos
- Estos objetos tienen métodos útiles para tratar con los tipos básicos

Tipo	Envoltorio
int	Integer
long	Long
float	Float
double	Double
short	Short
byte	Byte
char	Character
boolean	Boolean
void	Void

## Tipos: Tipos Básicos: Envoltorios



```
/**
 * Nombre:      Envoltorios.java
 * Descripción: Muestra uso de envoltorios de tipos básicos
 * Autor:       Manuel Pereira
 */
public class Envoltorios {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        int var = 65;
        Integer envoltorioDeI = new Integer(var);
        Integer otroEnvoltorio = Integer.valueOf("4859");
        int otraVar = Integer.parseInt("949");
        System.out.println("Envoltorio de i: " + envoltorioDeI.toString());
        System.out.println("Otro envoltorio: " + otroEnvoltorio.toString());
    }
}
```

```
C:\Practicas>javac Envoltorios.java
C:\Practicas>java Envoltorios
Envoltorio de i: 65
Otro envoltorio: 4859
C:\Practicas>_
```



## Tipos: Variables



- Necesario declarar la variable antes de utilizarla

```
{  
    int a;  
    a = 9;  
    {  
        int b=a+1;  
    }  
    a = 10;  
}
```

- Ámbito: el bloque de código en el que está declarada (delimitado por llaves {})

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

```
int vent = 3423, otraVar = 7382;  
boolean b1 = false, b2 = true;  
double db = 3948.493;  
float fl = 932.543F;
```

## Tipos: Conversión de Tipos



- Cuando es posible, se realiza de forma automática

```
char c = 'a';  
int i = c;      // CORRECTO  
short s = c;    // INCORRECTO: char no cabe en short por signo  
s = 678;        // CORRECTO
```

- En otros casos, forzado por el programador: castings

```
double dou = 123.67;  
int destino = (int) dou; // CORRECTO, coge parte entera
```

## Tipos: Cadenas de caracteres



- Cadenas de Caracteres
  - Entre comillas dobles
  - Clase de utilidad **String**
  - Operador de concatenación **+**

```
public class Strings {  
    /**  
     * @param args Argumentos de línea de parámetros  
     */  
    public static void main(String[] args) {  
        String cad1 = "El numero";  
        String cad2 = " es el siguiente: ";  
        String cad3 = String.valueOf(9483);  
        String cad4 = cad1 + cad2 + cad3;  
        System.out.println(cad4);  
    }  
}
```

```
C:\Practicas>javac Strings.java  
C:\Practicas>java Strings  
El numero es el siguiente: 9483  
C:\Practicas>_
```

## Tipos: Arrays



- Conjunto de datos de un tipo determinado

```
int[] a;  
a = new int[4];  
long[] b = new long[34];  
short[] c = {32, 434, -193};  
short d = c[0];           // d contiene 32  
short e = c[2];           // e contiene -193
```

- Copia de Arrays (System.arraycopy)
- Arrays Multidimensionales

# Tipos: Arrays



```
/**
 * Nombre: EjemploArrays1
 * Descripción: Ejemplo de Arrays y excepción de fuera de rango
 * Autor: Manuel Pereira
 */
public class EjemploArrays1 {
    /**
     * @param args Argumentos recibidos por línea de parámetros
     */
    public static void main(String[] args) {
        int[] miArray;
        miArray = new int[2];
        miArray[0] = 24;
        miArray[1] = 33;
        // Funciona
        System.out.println("Valor 0: " + miArray[0]);
        System.out.println("Valor 1: " + miArray[1]);
        // Falla porque se sale del rango
        miArray[2] = 45;
        System.out.println("Valor 2: " + miArray[2]);
    }
}
```

```
Simbolo del sistema
C:\Practicas\lecciones>javac EjemploArrays1.java
C:\Practicas\lecciones>java EjemploArrays1
Valor 0: 24
Valor 1: 33
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 2
    at EjemploArrays1.main(EjemploArrays1.java:20)
C:\Practicas\lecciones>
```

# Tipos: Arrays



```
public class InicializaArray {
    public static void main(String[] args) {
        int[] unaDim;

        /*
        unaDim = new int[3];
        unaDim[0] = 2;
        unaDim[1] = 4;
        unaDim[2] = 6;
        */
        unaDim = new int[] {2, 4, 6};

        int[][] dosDim;
        /*
        dosDim = new int[2][2];
        dosDim[0][0] = 3;
        dosDim[0][1] = 5;
        dosDim[1][0] = 12;
        dosDim[1][1] = 23;
        */
        dosDim = new int[][] {
            {3, 5},
            {12, 23}
        };
    }
}
```

## Tipos: Arrays



- Para acceder a la longitud (tamaño) de un array: **length**

```
/**
 * Nombre: EjemploArraysLength
 * Descripción: Ejemplo de Arrays y excepción de fuera de rango
 * Autor: Manuel Pereira
 */
public class EjemploArraysLength {
    /**
     * @param args Argumentos recibidos por línea de parámetros
     */
    public static void main(String[] args) {
        int[] a = new int[] {1, 3, 5};
        System.out.println("La longitud es: " + a.length);
    }
}
```

## Agenda



- Introducción
- Tipos
- **Operadores**
  - Aritméticos
  - Relacionales
  - Lógicos
  - A nivel de bit
  - De asignación
  - Precedencia de Operadores
- Control de Flujo
- E/S básica
- Atributos y Métodos
- Resumen



# Operadores: Aritméticos



- Operadores Aritméticos: **+, -, \*, /, %, ++, --**
- ++** y **--** son operadores de incremento, pueden ir como prefijo o sufijo, y tienen distinta precedencia

```
public class Incrementos {  
  
    public static void main (String[] args) throws Exception {  
        int j = 5;  
        int k = j++;  
        System.out.println("j vale: " + j + ", k vale: " + k);  
        int l = ++j;  
        System.out.println("j vale: " + j + ", l vale: " + l);  
    }  
}
```

ex Símbolo del sistema

```
C:\Practicas\lecciones>javac Incrementos.java  
C:\Practicas\lecciones>java Incrementos  
j vale: 6, k vale: 5  
j vale: 7, l vale: 7  
C:\Practicas\lecciones>
```

# Operadores: Aritméticos



```
public class NumerosPares {  
  
    public static void main(String[] args) {  
  
        for(int i=1; i<=20; i++) {  
            if(i%2 == 0) {  
                System.out.println("Número par: " + i);  
            }  
            else {  
                System.out.println("Número impar: " + i);  
            }  
        }  
    }  
}
```

ex Símbolo del sistema

```
C:\Practicas\lecciones>javac NumerosPares.java  
C:\Practicas\lecciones>java NumerosPares  
N-mero impar: 1  
N-mero par: 2  
N-mero impar: 3  
N-mero par: 4  
N-mero impar: 5  
N-mero par: 6  
N-mero impar: 7  
N-mero par: 8  
N-mero impar: 9  
N-mero par: 10  
N-mero impar: 11  
N-mero par: 12  
N-mero impar: 13  
N-mero par: 14  
N-mero impar: 15  
N-mero par: 16  
N-mero impar: 17  
N-mero par: 18  
N-mero impar: 19  
N-mero par: 20  
C:\Practicas\lecciones>
```

# Operadores: Relacionales



- Sirven para comparaciones
- Operadores relacionales: **==, !=, >, <, >=, <=**
- Comparación de Strings: Usar equals

```
/**
 * Nombre: OperadoresRelacionales.java
 * Descripción: Muestra uso de operadores relacionales
 * Autor: Manuel Pereira
 */
public class OperadoresRelacionales {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        String s = "cadena";
        String p = "otra cadena"; // True
        boolean res = "cadena".equals(s); // False
        boolean res2 = s.equals(p);
        System.out.println("res vale: " + res + ", y res2 vale: " + res2);
    }
}
```

```
Command Prompt
C:\Practicas>javac OperadoresRelacionales.java
C:\Practicas>java OperadoresRelacionales
res vale: true, y res2 vale: false
C:\Practicas>
```

# Operadores: Lógicos



- Operadores lógicos: **&, |, &&, ||, !**
- &: And, &&: And en cortocircuito
- |: Or, ||: Or en cortocircuito
- Operadores en cortocircuito dejan de evaluar cuando se conoce a ciencia cierta el resultado

```
public static void main(String[] args) {
    int a, b;
    boolean r;
    a = 3;
    b = 8;
    r = a!=0 | b>a; // Se evalúan las dos aunque a!=0 es cierto

    int num = 0;
    if (num<0 && 3/num==0) { // No se produce división por cero porque
        System.out.println("uno"); // la primera expresión no se cumple así que
    } else { // la segunda no se evalúa
        System.out.println("dos");
    }
}
```

# Operadores: Lógicos



```
public class ParesDivisiblesPorCinco {  
    public static void main(String[] args) {  
  
        for(int i=1; i<=30; i++) {  
            if(i%2==0 && i%5==0) {  
                System.out.println("Número par y divisible por 5: " + i);  
            }  
            else {  
                System.out.println("Número de otro tipo: " + i);  
            }  
        }  
    }  
}
```

```
Símbolo del sistema  
C:\Practicas\lecciones>javac ParesDivisiblesPorCinco.java  
C:\Practicas\lecciones>java ParesDivisiblesPorCinco  
N numero de otro tipo: 1  
N numero de otro tipo: 2  
N numero de otro tipo: 3  
N numero de otro tipo: 4  
N numero de otro tipo: 5  
N numero de otro tipo: 6  
N numero de otro tipo: 7  
N numero de otro tipo: 8  
N numero de otro tipo: 9  
N numero par y divisible por 5: 10  
N numero de otro tipo: 11  
N numero de otro tipo: 12  
N numero de otro tipo: 13  
N numero de otro tipo: 14  
N numero de otro tipo: 15  
N numero de otro tipo: 16  
N numero de otro tipo: 17  
N numero de otro tipo: 18  
N numero de otro tipo: 19  
N numero par y divisible por 5: 20  
N numero de otro tipo: 21  
N numero de otro tipo: 22  
N numero de otro tipo: 23  
N numero de otro tipo: 24  
N numero de otro tipo: 25  
N numero de otro tipo: 26  
N numero de otro tipo: 27  
N numero de otro tipo: 28  
N numero de otro tipo: 29  
N numero par y divisible por 5: 30  
C:\Practicas\lecciones>
```

# Operadores: Lógicos



```
public class Cortocircuito {  
    public static void main(String[] args) {  
        int k=1, m=2;  
        // Operador NO en cortocircuito |  
        // m++ se evalua, por lo tanto m valdrá 3  
        if(k == 1 | m++ == 3) {  
            System.out.println("Condicion satisfecha");  
        }  
        System.out.println("m vale: " + m);  
  
        int j=1, i=2;  
        // Operador en cortocircuito ||  
        // i++ NO se evalua, por lo tanto i valdrá 3  
        if(j == 1 || i++ == 3) {  
            System.out.println("Condicion satisfecha");  
        }  
        System.out.println("i vale: " + i);  
    }  
}
```

```
Símbolo del sistema  
C:\Practicas\lecciones>javac Cortocircuito.java  
C:\Practicas\lecciones>java Cortocircuito  
Condicion satisfecha  
m vale: 3  
Condicion satisfecha  
i vale: 2  
C:\Practicas\lecciones>
```

## Operadores: A nivel de Bit



- Aplicables a enteros (int, long, short, char, byte)
- Operadores a nivel de bit: `~, &, |, ^, >>, >>>, <<`
- Los valores **byte** y **short** promocionan a **int**

```
/**
 * Nombre: OperadoresBit.java
 * Descripción: Muestra uso de operadores a nivel de bit
 * Autor: Manuel Pereira
 */
public class OperadoresBit {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        byte b = 64, a;
        int i;
        i = b << 2;
        a = (byte) (b << 2);
        System.out.println("Valor de b:" + b);
        System.out.println("Valor de i:" + i);
        System.out.println("Valor de a:" + a);
    }
}
```

```
Command Prompt
C:\Practicas>javac OperadoresBit.java
C:\Practicas>java OperadoresBit
Valor de b:64
Valor de i:256
Valor de a:0
C:\Practicas>
```

## Operadores: Asignación



- Operadores de Asignación: `~, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=`
- `a = a + 5;`      `a += 5;`
- `boolean b = a > 5 ? true : false;`

```
public class NumerosPares2 {
    public static void main(String[] args) {
        for(int i=1; i<=20; i++) {
            String s = (i%2 == 0)? "Es par" : "Es impar";
            System.out.println("El número " + i + s);
        }
    }
}
```

```
Símbolo del sistema
C:\Practicas\lecciones>javac NumerosPares2.java
C:\Practicas\lecciones>java NumerosPares2
El número 1Es impar
El número 2Es par
El número 3Es impar
El número 4Es par
El número 5Es impar
El número 6Es par
El número 7Es impar
El número 8Es par
El número 9Es impar
El número 10Es par
El número 11Es impar
El número 12Es par
El número 13Es impar
El número 14Es par
El número 15Es impar
El número 16Es par
El número 17Es impar
El número 18Es par
El número 19Es impar
El número 20Es par
C:\Practicas\lecciones>
```



# Operadores: Precedencia



MAYOR  
PRECEDENCIA

MENOR  
PRECEDENCIA

Operador	Tipo
[] . () expr++ expr--	Operadores posfijos
++expr --expr +expr -expr ~ !	Operadores unarios
(cast) new	Creación o conversión
* / %	Multiplicación
+ -	Suma
>> >>> <<	Desplazamiento
> >= <= < instanceof	Comparación
== !=	Igualdad
&	AND a nivel de bit
^	XOR a nivel de bit
	OR a nivel de bit
&&	AND lógico
	OR lógico
?:	Condicional
= += -= *= /= %= &=  = =	Asignación
<<= >>= >>>=	

# Agenda



- Introducción
- Tipos
- Operadores
- **Control de Flujo**
  - Condicionales
    - If-else
    - switch
  - Bucles
    - while
    - for
    - do-while
- E/S básica
- Atributos y Métodos
- Resumen



## Control de Flujo: if-else



```
/**
 * Nombre:      Condicionales.java
 * Descripción: Muestra uso de condicionales
 * Autor:       Manuel Pereira
 */
public class Condicionales {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        int resultado;
        int valor = 3;
        if(valor > 3) {
            resultado = 1;
        } else if (valor == 3) {
            resultado = 0;
        } else {
            resultado = -1;
        }
    }
}
```

## Control de Flujo: switch



```
int dia = 3;
switch(dia) {
    case 1:
        System.out.println("Lunes");
        break;
    case 2:
        System.out.println("Martes");
        break;
    case 3:
        System.out.println("Miercoles");
        break;
    case 4:
        System.out.println("Jueves");
        break;
    case 5:
        System.out.println("Viernes");
        break;
    case 6:
        System.out.println("Sabado");
        break;
    case 7:
        System.out.println("Domingo");
        break;
    default:
        System.out.println("El valor es erroneo");
}
```

## Control de Flujo: switch



```
public class EquivocoBreak {  
    public static void main(String[] args) {  
        int a = 3;  
        switch(a) {  
            case 2:  
                System.out.println("Hola");  
                break;  
            case 3:  
                System.out.println("Adios");  
            case 4:  
                System.out.println("Hola otra vez");  
                break;  
            default:  
                System.out.println("Ninguna de las anteriores");  
                break;  
        }  
    }  
}
```

```
cs Símbolo del sistema  
C:\Practicas\lecciones>javac EquivocoBreak.java  
C:\Practicas\lecciones>java EquivocoBreak  
Adios  
Hola otra vez  
C:\Practicas\lecciones>
```

## Control de Flujo: while



```
/**  
 * @param args Argumentos de línea de parámetros  
 */  
public static void main(String[] args) {  
    int valor = 250;  
    while(valor > 0) {  
        System.out.println("El valor es: " + valor);  
        valor--;  
    }  
}
```

## Control de Flujo: for



```
int i;  
for(i=0; i<100; i++) {  
    if((i % 2) == 0) {  
        System.out.println("El número " + i + " es divisible por dos");  
    }  
}
```

## Control de Flujo: do-while



```
int j = -1;  
do {  
    System.out.println();  
    j--;  
} while(j > 0);
```

## Control de Flujo: break y continue



- **break** sale de la ejecución del bucle
- **continue** salta a la siguiente ejecución del bucle

```
int i;
for(i=0; i<100; i++) {
    if(i == 50) {
        break;           // Sale del bucle a la mitad
    }
}

for(i=0; i<100; i++) {
    if(i < 50) {
        continue;       // No se imprimen los numeros menores que 50
    }
    System.out.println("Número: " + i);
}
```

## Agenda



- Introducción
- Tipos
- Operadores
- Control de Flujo
- **E/S básica**
  - Flujos de Datos
  - Entradas y Salidas Estándar
- Atributos y Métodos
- Resumen



## E/S Básica: Flujos de datos



- Utilización de flujos de datos (**streams**)
- Paquete `java.io`
- **`java.io.InputStream`**
- **`java.io.OutputStream`**
- `FileInputStream`,  
`BufferedInputStream`, ....

## E/S Básica: Entradas y Salidas estándar



- **`System.out`** -> Salida estándar
- **`System.err`** -> Salida con errores
- **`System.in`** -> Entrada estándar

```
import java.io.*;

/**
 * Nombre:      IOEstandar.java
 * Descripción: Muestra uso de condicionales
 * Autor:       Manuel Pereira
 */
public class IOEstandar {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) throws IOException {
        String cadena;
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Escribe una cadena: ");
        cadena = stdin.readLine();
        System.out.println("Has escrito: " + cadena);
    }
}
```

# Agenda



- Introducción
- Tipos
- Operadores
- Control de Flujo
- E/S básica
- **Atributos y Métodos**
  - Paso de Parámetros
- Resumen



## Atributos y Métodos: Paso de Parámetros



- Los objetos se pasan por referencia.
- Los tipos básicos se pasan por valor

```
Command Prompt
C:\Practicas>javac Primos.java
C:\Practicas>java Primos
Números primos hasta 25
2
3
5
7
11
13
17
19
23
Total: 0 primos
C:\Practicas>_
```

```
/**
 * Nombre:      Primos.java
 * Descripción: Imprime números primos
 * Autor:       Manuel Pereira
 */
public class Primos {
    /**
     * @param args Argumentos de línea de parámetros
     */
    public static void main(String[] args) {
        int maximo = 25, numero = 0;
        System.out.println("Números primos hasta " + maximo);
        // MAL, los tipos básicos se pasan por valor, no por referencia
        imprimirPrimos(maximo, numero);
        System.out.println("Total: " + numero + " primos"); // Mal, imprime cero
    }

    public static void imprimirPrimos(int max, int num) {
        num = 2;
        System.out.println("1");
        System.out.println("2");
        for(int i=3; i<=max; i++) {
            int divisor = 2;
            while(i%divisor!=0 && divisor<i-1) {
                divisor++;
            }
            if(divisor == i-1) {
                System.out.println(i);
                num++;
            }
        }
    }
}
```

# Atributos y Métodos: Paso de Parámetros



```
public class PrimosBueno {  
    /**  
     * Clase interna que recubre un entero  
     */  
    static class Envoltorio {  
        int dato;  
    }  
    /**  
     * @param args Argumentos de línea de parámetros  
     */  
    public static void main(String[] args) {  
        int maximo = 25;  
        Envoltorio numero = new Envoltorio();  
        numero.dato = 0;  
  
        System.out.println("Números primos hasta " + maximo);  
        // MAL, los tipos básicos se pasan por valor, no por referencia  
        imprimirPrimos(maximo, numero);  
        System.out.println("Total: " + numero.dato + " primos"); // Mal, imprime cero  
    }  
  
    public static void imprimirPrimos(int max, Envoltorio numero) {  
        numero.dato = 2;  
        System.out.println("1");  
        System.out.println("2");  
        for (int i=3; i<=max; i++) {  
            int divisor = 2;  
            while (i%divisor!=0 && divisor<=i-1) {  
                divisor++;  
            }  
            if (divisor == i-1) {  
                System.out.println(i);  
                numero.dato++;  
            }  
        }  
    }  
}
```

```
C:\Practicas>javac PrimosBueno.java  
C:\Practicas>java PrimosBueno  
Números primos hasta 25  
1  
2  
3  
5  
7  
11  
13  
17  
19  
23  
Total: 10 primos  
C:\Practicas>
```

## Agenda



- Introducción
- Tipos
- Operadores
- Control de Flujo
- E/S básica
- Atributos y Métodos
- **Resumen**





# Resumen



- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>▪ Introducción<ul style="list-style-type: none"><li>▪ Historia de Java</li><li>▪ Características Principales</li><li>▪ Hello World</li></ul></li><li>▪ Tipos<ul style="list-style-type: none"><li>▪ Tipos Básicos</li><li>▪ Variables</li><li>▪ Conversión de tipos</li><li>▪ Cadenas de Caracteres</li><li>▪ Arrays</li></ul></li><li>▪ Operadores<ul style="list-style-type: none"><li>▪ Aritméticos</li><li>▪ Relacionales</li><li>▪ Lógicos</li><li>▪ A nivel de bit</li><li>▪ De asignación</li><li>▪ Precedencia de Operadores</li></ul></li></ul> | <ul style="list-style-type: none"><li>▪ Control de Flujo<ul style="list-style-type: none"><li>▪ Condicionales<ul style="list-style-type: none"><li>▪ If-else</li><li>▪ switch</li></ul></li><li>▪ Bucles<ul style="list-style-type: none"><li>▪ while</li><li>▪ for</li><li>▪ do-while</li></ul></li></ul></li><li>▪ E/S básica<ul style="list-style-type: none"><li>▪ Flujos de Datos</li><li>▪ Entradas y Salidas estándar</li></ul></li><li>▪ Atributos y Métodos<ul style="list-style-type: none"><li>▪ Paso de Parámetros</li></ul></li></ul> |
|--|--|

## Resumen: Para más información



- **Página de Java en Sun (ver "The Java Tutorial")**
  - <http://java.sun.com>
- **Introducción a la sintaxis de Java en Español:**
  - [http://eees.ii.uam.es/alfonso/web\\_poo\\_04/teoria/material/sintaxis\\_java.pdf](http://eees.ii.uam.es/alfonso/web_poo_04/teoria/material/sintaxis_java.pdf)
  - <http://www.people.virginia.edu/~am2zb/cursos/java/aplicaciones/intro.htm>
  - <http://www.desarrolloweb.com/articulos/1670.php?manual=57>
  - <http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/Index.htm>