

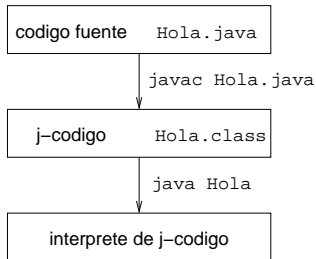
Java Básico

Entorno de Programación Emacs/JDEE

Luis Fernando Llana Díaz

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

9 de abril de 2007



- Sintaxis inspirada en C++.
- Lenguaje orientado a objetos.
- Código compilado independiente de la máquina **j-código**.
- Lenguaje **de moda**.
- Compilador e intérpretes gratuito. Libres según la versión.

Entornos de programación

- Consola+editor de textos.
- Editor de texto emacs + JDEE.
- Eclipse y NetBeans, entornos libres.
- JBuilder entorno propietario.

GNU-emacs (JDEE)

Ventajas

- Código libre.
- Editor potente (coloreado, sangrado automático, compilación integrada).
- Editor programable.

GNU-emacs (JDEE)

Ventajas

- Código libre.
- Editor potente (coloreado, sangrado automático, compilación integrada).
- Editor programable.

Desventajas

- Dificultad para entornos gráficos.
- Entorno desconocido.

Programa "Hola Mundo"

```
/* El primer programa
   como aparece en todos los
   libros */
1
2
3
4
/** Esto es un comentario
   * para ser procesado con 'javadoc'.
   * Clase "ejecutable" Hola mundo.
   */
5
6
7
8
public class HolaMundo {
9
   // Clase pública. Sólo puede haber una en cada fichero.
   // Además el fichero se debe llamar 'PrimerPrograma.java'
10
   /** El método estático "main" es el
11
    * que comienza a ejecutarse.
12
    * @author Luis Fernando Llana Díaz
13
    * @version 1.0
14
    * @since 5/oct/98
15
    * @param args son los parametros introducidos en la línea de comandos
16
    */
17
18
   public static void main (String[] args){
19
       System.out.println("Hola");
20
   }
21
22
}
23
```

Elementos básicos

- Instrucción de asignación.
- Instrucción condicional (`if`).
- Instrucción iterativa (`while`, `for`, `do-while`).

Tipos Básicos

Tipo	Que contiene	Rango
byte	entero de 8 bits	-128 a 128
short	entero de 16 bits	-32.768 a 32.767
int	entero de 32 bits	-2.147.483.648 a 2.147.483.647
long	entero de 64 bits	-2^{63} a $2^{63} - 1$
float	coma flotante de 32 bits	6 dígitos significativos (10^{-46} , 10^{38})
double	coma flotante de 64 bits	15 dígitos significativos (10^{-324} , 10^{308})
char	Carácter Unicode	'a', 'b', ... 'á', 'ñ', ... '0', '1', ...
boolean	valores booleanos	true y false

Literales

Enteros:

- Decimal: 37
- Hexadecimal: 0x25
- Octal: 045

Caracteres

- 'a', 'b', ... 'á', 'ñ', ..., '0', '1', ...
- Caracteres "escape": '\n', '\\', '\'', '\\"'.

Reales

Float: 1.0345F, 1.04E-12f, .0345f,
1.04e-13f

Double: 1.0345, 1.0345d, 5.6E-120D

Cadenas de caracteres: "hola\n".

Declaración de variables

- Estilo C++.
- Permite declaración de variables `al vuelo`.
- Java es sensible a mayúsculas y minúsculas.
- Notación:
 - Nombres de variables y clases largos.
 - Las variables todas en minúsculas, si una variable se compone de varias palabras, la segunda y siguientes palabras empiezan por mayúscula:

`numero, numeroPalabras`

- Las clases empiezan por mayúsculas y las demás en minúsculas:

`Lista, ListaEnlazada`

Declaración de variables

```
public class Variables {
    public static void main (String[] args)
    {
        int num1; // Declaración estándar
        double media=2.5; // Declaración e inicialización

        num1=6;
        System.out.println("num1: "+num1);

        int x, num2; // Declaración al vuelo
        x=2; num2=10;
        System.out.println("x: "+x+"\nnum2: "+num2);

        int num3=x*num2; // Declaración e inicialización al vuelo
        System.out.println("num3: "+num3);

        media=num1/num2;
        // media=0
        System.out.println("media: "+media);

        media=(double)num1/num2;
        // media=0.6
        System.out.println("media: "+media);
    }
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Operadores

Operadores aritméticos

- + suma para enteros y reales.
- - resta para enteros y reales.
- * producto para enteros y reales.
- / división entera entre enteros y con decimales para reales.
- % resto de la división entera entre enteros.

Operadores lógicos

Operadores lógicos

- conjunción `&&`, ¡ojo! no confundir con `&`.
- disyunción `||`, ¡ojo! no confundir con `|`.
- negación `!`.

Operadores relacionales

- igualdad `==`.
- desigualdad `!=`.
- comparación `<`, `<=`, `>`, `>=`.

Operadores lógicos

Expresión condicional

```
i%2 ? 1:0  
.....  
mes==2 && bisiestro ? 29:28
```

1
2
3

Operadores lógicos no estrictos

```
if (x!=0 && 1/x>3)  
.....  
while (i<=n && v[i]==0)
```

1
2
3

Constantes

```
public static final double CUALQUIERA=3.14159265;  
public static final int MAXIMO_NUMERO_ELEMENTOS=5000;
```

1
2

Cadenas de caracteres

```
public class PrString {  
    public static void main(String args[]) {  
        String s1 = "Hola";  
        String s2 = "Patata";  
        String s3 = s1+" "+s2;  
        System.out.println(s3.length());  
        for (int i = 0; i < s3.length(); i++) {  
            String mensaje = "Letra "+i+": "+s3.charAt(i)+":";  
            System.out.println(mensaje);  
        }  
    }  
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

String es una clase *especial* en Java. Sus objetos son inmutables.

Instrucción condicional

```
if (expresión booleana)
    instrucción
```

1

2

3

```
if (expresión booleana)
    instrucción
```

4

5

```
else
```

6

7

```
    instrucción
```

```
if (x==0)
    System.out.println("x vale 0");
else {
    System.out.print("x vale ");
    System.out.println(x);
}
```

1

2

3

4

5

6

Instrucción condicional

```
public class PruebaIf {  
    public static void main(String[] args) {  
        int x=0;  
  
        if (x!=0 && 1/x=0);  
            System.out.println(1/x);  
        else  
            System.out.print("x está ");  
            System.out.println("indefinido");  
    }  
}
```

1
2
3
4
5
6
7
8
9
10

Instrucciones iterativas

```
while (expresión booleana)  
    instrucción
```

1
2

```
do  
    instrucción  
while (expresión booleana)
```

1
2
3

```
for (inicialización; expresión booleana; incremento)  
    instrucción
```

1
2

Instrucciones iterativas

```
i=0;  
while (i<n && v[i]==0) i++;
```

```
1 for (int i=1; i<n; i++){  
2     int posMin=i;  
3     for (int j=i+1; j<n; j++){  
4         if (v[j]<v[posMin]) posMin=j;  
5     }  
6     intercambiar(v,i,posMin);  
7 }
```

Usos “prohibidos” de bucle for

Uso de bucle for como bucle while:

```
for (i=0; i<n && v[i]==0; i++);
```

1

Instrucciones iterativas

```
public class PruebaBucles {
    static void bucleWhile() {
        int i=1;
        boolean para = false;
        System.out.println("Bucle WHILE");
        while (i<=1000 && !para) {
            System.out.println("i: "+i);
            // Sale del bucle cuando se cumple la condición.
            if (i==10) {
                System.out.println("Me voy...");
                para = true;
            }
            i++;
        }
    }

    static void bucleFor() {
        System.out.println("Bucle FOR");
        for (int i=1; i<=10; i++)
            System.out.println("i: "+i);
    }
    // Después del bucle la variable 'i' no está definida;

    public static void main(String[] args) {
        bucleWhile();
        bucleFor();
    }
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

Distinción de casos

```
switch (n%7){  
  case 0:  
    System.out.println("Es lunes");  
    break;  
  case 1:  
    System.out.println("Es martes");  
    break;  
    ...  
    ...  
  case 6:  
    System.out.println("Es domingo");  
    break;  
  default:  
    System.out.println("Esto es imposible");  
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Distinción de casos

```
public class PruebaSwitch {
    static char prueba1(char c) {
        char letra;

        switch (c) {
            case 'a':
                letra='-';
                break;
            case 'e':
                letra='-';
                break;
            case 'i':
                letra='-';
                break;
            case 'o':
                letra='-';
                break;
            case 'u':
                letra='-';
                break;
            default: letra=c;
        }
        return letra;
    }
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

Distinción de casos

```
static void prueba2(int i) {  
    switch (i){  
        case 1:  
            System.out.println("Caso 1: "+i);  
        case 2:  
            System.out.println("Caso 2:"+i);  
        case 3:  
            System.out.println("Caso 3:"+i);  
        case 4:  
            System.out.println("Caso 4:"+i);  
        case 5:  
            System.out.println("Caso 5:"+i);  
        default:  
            System.out.println("Caso por defecto");  
    }  
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Distinción de casos

```
public static void main(String[] args) {  
    char letra=prueba1(args[0].charAt(0));  
    System.out.println("letra:"+letra+":");  
  
    System.out.println("llamada con 1");  
    prueba2(1);  
    System.out.println("llamada con 2");  
    prueba2(2);  
    System.out.println("llamada con 3");  
    prueba2(3);  
    System.out.println("llamada con 4");  
    prueba2(4);  
    System.out.println("llamada con 5");  
    prueba2(5);  
    System.out.println("llamada con 6");  
    prueba2(6);  
}  
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Métodos

Java no tiene procedimientos, sólo hay funciones (métodos).

- Existen funciones que devuelven nada (tipo `void`).
- Todos los parámetros son de entrada, pero cuidado con los punteros o referencias.
- Las funciones admiten sobrecarga.

Sobrecarga

```
public class Sobrecarga {
    final static double PI=3.1415926535897932;

    static double max(double a, double b){
        double m=a;
        if (b>a) m=b;
        return m;
    }
    static double max(double a){
        double m=PI;
        if (a>PI) m=a;
        return m;
    }
    static int max(int a, int b){
        int m=a;
        if (b>a) m=b;
        return m;
    }
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Sobrecarga

```
public static void main(String [] args){
    double x=Double.parseDouble(args[0]);
    double y=Double.parseDouble(args[1]);
    int a=Integer.parseInt(args[2]);
    int b=Integer.parseInt(args[3]);

    System.out.print("Máximo entre "+x+" y "+y+": ");
    System.out.println(max(x,y));

    System.out.print("Máximo entre "+x+" y "+PI+": ");
    System.out.println(max(x));

    System.out.print("Máximo entre "+a+" y "+b+": ");
    System.out.println(max(a,b));
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Arrays

```
int [] v; //definición de arrays
double [][] m; //definición de matrices
.....
int [] v = new int[10]; //definición y creación de array de 10 componentes
.....
int [] v = {1,2,3,4,5}; //definición y creación de array
.....
int [] m = new int[10][10];
.....
int [][] m = {{1,2,3},{4,5,6},{7,8,9}};
```

1
2
3
4
5
6
7
8
9
10

Todos los arrays empiezan a numerarse desde 0;

Arrays

```
import java.util.Random;
public class PrArray {
    private static final int N=10;
    private static void escribe(int [] v) {
        System.out.print("[");
        for (int i = 0; i<v.length ; i++) {
            if (i>0)
                System.out.print(",");
            System.out.print(v[i]);
        }
        System.out.println("]");
    }
    public static void main(String [] args) {
        int [] v = new int[N];
        genera(v);
        escribe(v);
        ordena(v);
        escribe(v);
        int [] u = v;
        u[0]=10000;
        escribe(u);
    }
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

Arrays

```
private static void genera(int [] v) {  
    Random r = new Random();  
    for (int i = 0; i<v.length ; i++)  
        v[i] = r.nextInt() % (3*N);  
}  
private static void ordena(int [] v) {  
    for (int i = 0; i < v.length ; i++) {  
        int pos = i;  
        for (int j = i+1; j<v.length ; j++)  
            if (v[j]<v[pos])  
                pos = j;  
        int aux = v[i];  
        v[i] = v[pos];  
        v[pos]=aux;  
    }  
}
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Hechos

- Cualquier programa puede tener varios cientos o miles de clases.
- La propia API de Java tiene más de 3000 clases.
- No es difícil escoger el nombre de una clase que ya esté cogido.

Paquetes

Hechos

- Cualquier programa puede tener varios cientos o miles de clases.
- La propia API de Java tiene más de 3000 clases.
- No es difícil escoger el nombre de una clase que ya esté cogido.

Paquete

Es una colección de clases con un mismo objetivo.

Clase soporte.Teclado

```
package soporte;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
public class Teclado {  
    ...  
    ...  
}
```

1
2
3
4
5
6
7
8
9

Esqueleto de proyecto Java

```
.
|-- build.xml
|-- prj.el
'-- src
    |-- pr1
    |   |-- Pr.java
    |-- pr2
    |   |-- Pr.java
'-- lib
'-- classes
    |-- pr1
    |   |-- Pr.class
    |-- pr2
    |   |-- Pr.class
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14