Unidad 3
Sintaxis básica de Java

Programación
1º D.A.M.

1

Contenido

1. Elementos del lenguaje
2. Primer programa en Java
3. Variables y constantes
4. Entrada / Salida estándar
5. Operadores y expresiones
6. Sentencias de control
7. Comentarios

2

1. Elementos del lenguaje

1. Conjunto de caracteres
2. Léxico vs. sintaxis
3. Elementos del programa
    1. Componentes léxicos o *tokens*
    2. Palabras reservadas
    3. Identificadores
    4. Tipos de datos
    5. Números
    6. Comentarios
    7. Expresiones
    8. Sentencias

3

1.1. Conjunto de caracteres

- Letras mayúsculas y minúsculas
    - `A – Z`
    - `a – z`
- Números
    - `0 – 9`
- Separadores
    - Espacio en blanco, tabulador
- Caracteres especiales
    - `(, ), {, }, [, ], <, >, _, ;, :, +, ^, ~, –, *, /, =, %, &, #, !, ?, ", ', ,.`

4

1.2. Léxico vs. Sintaxis

- Léxico del lenguaje
  - Elementos fundamentales a partir de los cuales se construyen los programas
- Sintaxis del lenguaje
  - Reglas para combinar e interpretar los elementos fundamentales del lenguaje

5

1.3. Elementos del programa

- Componente léxico (*token*)
  - Unidad del código fuente
  - `main, +, }, for, while, ...`
- Palabra reservada
  - Token con un significado para el compilador
  - `while, for, if, int, float, class, static, ...`
- Identificadores
  - Nombres de las variables, métodos, atributos, clases, objetos.

6

1.3. Elementos del programa

- Tipos de datos primitivos (básicos)
  - Indican cómo guardar datos en memoria
  - Posibilidades en Java
    - Caracteres
      - `char`
    - Números
      - Enteros: `byte, short, int, long`
      - Reales: `float, double`
    - Booleanos
      - `boolean`
- Números
  - Enteros
  - Reales

7

1.3. Elementos del programa

- Comentarios
  - Información adicional en el programa
    - Ayuda al programador a entender su código
  - Pueden generar documentación automática
  - Ignorados por el compilador
- Expresiones
  - Combinación de operandos y operadores
  - `3 + 4, a = 4 * 2, b < 9, ...`

8

1.3. Elementos del programa

- Sentencias
  - Especifican y controlan el flujo de ejecución
  - Expresión COMPLETA
  - Acabadas siempre en punto y coma
  - lvalue = rvalue
    - `8 + 3;` → Incompleta
    - `coches = 8 + 3;` → Completa
  - Sentencias compuestas o bloques de código

```
{
  Sentencia_1
  Sentencia_2
  Sentencia_3
}
```

9

1.3. Elementos del programa

- Operadores
  - Aritméticos: +, -, *, /, %, ++, --
  - Lógicos: ||, &&, !
  - Relacionales: >, <, >=, <=, ==, !=
  - De asignación: =, +=, -=, *=, /=, %=
  - Condicional: ?
  - A nivel de bit
    - De desplazamiento: >>, <<, >>>
    - Lógicos: &, |, ^, ~
  - Concatenación de cadenas: +

10

2. Primer programa en Java

1. Código fuente
2. Compilación y ejecución

11

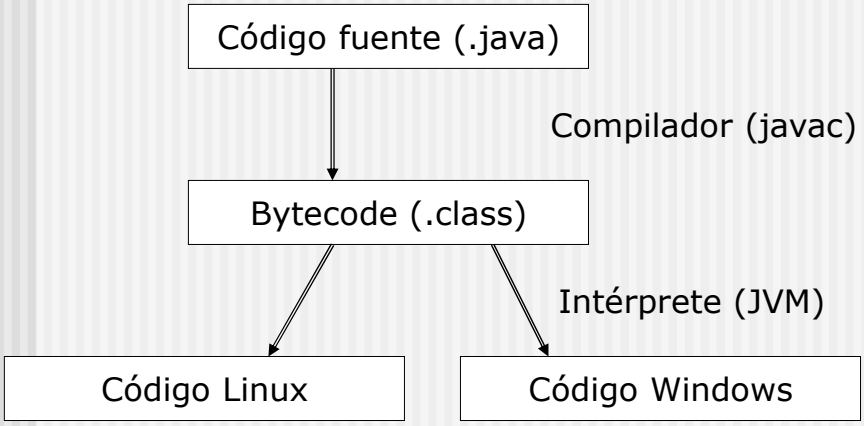
2. Primer programa en Java. Fuente

```
public class HolaMundo
{
  public static void main(String[] args)
  {
     System.out.println("HOLA, MUNDO !!");
  }
}
```

HolaMundo.java

12

2. Primer programa en Java. Compilación y ejecución



13

2. Primer programa en Java. Compilación y ejecución

- Compilación
  - `javac HolaMundo.java`
    - `javac *java`
  - Aparición del bytecode
    - HolaMundo.class
- Ejecución
  - `java HolaMundo`

14

3. Variables y constantes

1. Variables
   1. Tipos de datos básicos
   2. Declaración de variables
   3. Conversión de tipos
   4. Cadenas de caracteres
   5. Envoltorios
   6. Tipos compuestos
2. Constantes

15

3. Variables y constantes

- Variables vs. Constantes
  - Analogía
    - Posición de memoria que contiene datos
  - Diferencia
    - Variables
      - Los datos se pueden recuperar y modificar
    - Constantes
      - Los datos sólo se pueden recuperar, no modificar

16

3.1. Variables

- Declaración
  - Reserva del espacio en memoria
  - Partes
    - Tipo de datos
    - Nombre o identificador
      - Significativo
      - Letras, números, subrayado (_) y dólar ($)
        - Sin empezar por número
        - Sin ser palabras reservadas
        - *Case sensitive*
  - Ejemplos

```
float precio;
int num_Piezas;
char letra;
```

17

3.1.1. Tipos de datos básicos

- Enteros
  - Siempre con signo

Tipo Java	Tamaño	Número de valores	Rango de valores
`byte`	1 byte	$2^8 = 256$	-128..+127
`short`	2 bytes	$2^{16} = 65,536$	-32768..+32767
`int`	4 bytes	$2^{32} = 4,294,967,296$	-2147483648.. +2147483647
`long`	8 bytes	$2^{64} = 18.4 \times 10^{18}$	-9223372036854775808 .. +9223372036854775807

18

3.1.1. Tipos de datos básicos

- Reales (coma flotante)

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

19

3.1.1. Tipos de datos básicos

- Caracteres
  - Tipo char
  - Representados en Unicode
    - 16 bits / carácter
      - 65535 posibilidades
      - 256 primeras: ASCII/ANSI
  - Siempre entre comillas simples
  - Secuencias de escape
    - Caracteres con significado especial

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

20

3.1.1. Tipos de datos básicos

- Tipo lógico
  - `boolean`
  - Valores posibles
    - `true` → verdadero
    - `false` → falso

21

3.1.2. Declaración de variables

- Reserva de espacio en memoria
- Posibilidad de iniciar en la declaración
- Posibilidad de definir varias en una línea
- Sintaxis

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

22

3.1.2. Declaración de variables

- Ámbito de la variable
  - Zona de código en que puede ser accedida
  - Bloque de código en que se ha declarado
- Tiempo de vida de la variable
  - Tiempo entre su declaración y su destrucción
  - Suele coincidir con el ámbito

23

3.1.3. Conversión de tipos

- Conversión implícita de tipos
  - Problema
    - Mezcla operaciones con operandos de tipos distintos
  - Solución
    - Compilador → Convierte a un único tipo
  - Reglas
    - Promoción
      - Vble. de menor tamaño al tipo de la de mayor tamaño
    - Rangos de menor a mayor
      - byte<short<int<long<float<double

24

3.1.3. Conversión de tipos

- Conversión explícita de tipos
  - *Casting* o refundición de tipos
  - Modo

    `(tipo) expresión;`
  - Conversión de la expresión al tipo `tipo`.

25

3.1.4. Cadenas de caracteres

- Clase String
  - Permite crear cadenas de caracteres
  - Definición de una cadena
    - `String nombre = "valor de cadena";`
    - Ejemplos
      - `String nombre = "Sergio";`
      - `String saludo = "¿Hola cómo estás?";`

NOTA: *más adelante se estudiará el manejo detallado de las cadenas de caracteres.*

26

3.1.5. Envoltorios

- Clases para recubrir tipo básico
- El tipo básico se convierte en objeto
- No pueden modificar valor del tipo básico
  - Ha de destruir el objeto
- Incorporan funciones de conversión
  - `parseInt`
    - Conversión de cadena al tipo básico
    - `Integer.parseInt("123")`
  - `toString`
    - Conversión del tipo básico a cadena
    - `Integer.toString(123);`

27

3.1.5. Envoltorios

- 9 envoltorios, para 9 tipos básicos
  - `Byte`
  - `Short`
  - `Integer`
  - `Long`
  - `Float`
  - `Double`
  - `Character`
  - `Boolean`
  - `Void`

28

3.1.6. Tipos compuestos

- Albergan más de un dato
  - Homogéneos
    - Todos los elementos del mismo tipo
      - Colecciones estáticas
        - Arrays
      - Colecciones dinámicas
        - Vectores, listas, pilas, colas, mapas, …
  - Heterogéneos
    - Elementos de diferente tipo
      - Clases

29

3.1.6. Tipos compuestos

- Arrays
  - Colecciones homogéneas de datos
  - Valores en posiciones contiguas
  - Accedidas con un identificador único
    - Índice para distinguir el elemento concreto
  - Tipos
    - Unidimensionales
    - Multidimensionales

30

3.1.6. Tipos compuestos

- Arrays
  - **Definición** de un array unidimensional
    - `tipo_datos nombre[] = new tipo_datos[n_elementos];`
    - `tipo_datos[] nombre = new tipo_datos[n_elementos];`
      - `tipo_datos`
        - Tipo de datos básico de los elementos
        - Clase a la que pertenecen los elementos
      - `nombre`
        - Nombre que se da a la variable compuesta
      - `n_elementos`
        - Número de elementos albergados

31

3.1.6. Tipos compuestos

- Arrays
  - Ejemplos de definición
    - `int edades[] = new int[30] ;`
      - Edad de 30 alumnos
    - `float temperaturas[] = new float[12];`
      - Tª media de cada mes
    - `Coche flota[] = new Coche[10];`
      - Flota de 10 coches

32

3.1.6. Tipos compuestos

- Arrays
  - **Acceso** a los campos
    - `nombre[índice]`
      - `nombre`
        - Identificador de la variable compuesta
      - `índice`
        - Valor entre 0 y (n_elementos – 1)
  - Ejemplos
    - `edades[0] = 19;  // Edad del primero`
    - `temperaturas[11] = 4.5f;// Tª de diciembre`
    - `temperaturas[12] = 2.5f;// ERROR!!!!!!!`

33

3.1.6. Tipos compuestos

- Clase
  - Contiene
    - Propiedades → Atributos
      - De tipos básicos
      - De otras clases
    - Comportamiento → Métodos
  - Permite crear objetos
    - Con esas propiedades
    - Con ese comportamiento

34

3.1.6. Tipos compuestos

- Clase
  - Definición de una clase con atributos

```
class MiClase {
  tipo1 atributo1;
  tipo2 atributo2;
  …
  }
```

NOTA: *más adelante se estudiarán las clases al completo, con sus métodos correspondientes (comportamiento)*

35

3.1.6. Tipos compuestos

- Clase
  - Creación de un objeto de la clase
    - `MiClase miObjeto = new MiClase();`
  - Acceso a los atributos de la clase
    - Operador .
      - `miObjeto.atributo1 = valor1;`
      - `miObjeto.atributo2 = valor2;`
      - …

36

3.1.6. Tipos compuestos

- Clase
  - Ejemplo
    - Definición →

```
class Coche{
   String marca;
   int potencia;
   float peso;
   }
```

    - Creación de un objeto
      - `Coche miCoche = new Coche();`
    - Acceso a los atributos (campos)
      - `miCoche.marca = "Seat";`
      - `miCoche.potencia = 140;`
      - `miCoche.peso = 1950.75f;`

37

3.2. Constantes

- Valor no modificable en ejecución
- Tipo de datos
  - De cualquier tipo de datos básico
    - Si es carácter : comillas simples
  - Cadena de caracteres
    - Comillas dobles
- Facilitan el mantenimiento
- Sintaxis

```
[static] final <tipo_datos> identificador = valor;
```

38

4. Entrada / salida estándar

1. Salida estándar
2. Entrada estándar

39

4.1. Salida estándar

- Salida estándar
  - Pantalla (*stdout)*
  - Escritura en salida estándar
    - `System.out.print`
    - `System.out.println`
- Salida estándar de error
  - Pantalla (*stderr)*
  - Escritura en salida estándar de error
    - `System.err.print`
    - `System.err.println`

40

4.2. Entrada estándar

- Teclado (*stdin)*
- Lectura de un carácter

```
char c = System.in.read();
```

- Lectura de otros tipos de datos
  - Requiere `import java.util.Scanner;`

```
Scanner teclado  = new Scanner(System.in);
String cadena    = teclado.nextLine();
int entero       = teclado.nextInt();
float real       = teclado.nextFloat();
long largo       = teclado.nextLong();
double real2     = teclado.nextDouble();
boolean booleano = teclado.nextBoolean();
teclado.close();
```

41

5. Operadores y expresiones

1. Operadores
   1. Operadores aritméticos
   2. Operadores relacionales
   3. Operadores lógicos
   4. Operadores de incremento y decremento
   5. Operadores de asignación
   6. Operadores de manejo de bits
   7. Operador condicional
2. Precedencia de los operadores

42

5.1.1. Operadores aritméticos

- Operaciones aritméticas
- Operadores
  - Suma `+`
  - Resta `-`
  - Multiplicación `*`
  - División `/`
  - Módulo `%`
  - Signo `-`
- Precedencia (menor a mayor)
  - `(+, -)` → `(/, *, %)` → `(-)`
  - Uso de paréntesis

43

5.1.2. Operadores relacionales

- Relaciones entre valores
  - Resultado
    - false → falso
    - true → verdadero
- Operadores
  - Mayor `>`
  - Mayor o igual `>=`
  - Menor `<`
  - Menor o igual `<=`
  - Igual `==`
  - Distinto `!=`
- Precedencia (menor a mayor)
  - Relacionales → Aritméticos

44

5.1.3. Operadores lógicos

- Conectivas lógicas Y, O y NEGACIÓN
  - Resultado
    - Falso `false`
    - Verdadero `true`
- Operadores
  - Y (AND) `&&`
  - O (OR) `||`
  - Negación (NOT) `!`
- Evaluación
  - De izquierda a derecha
  - Interrumpida cuando el resultado está claro

45

5.1.4. Operadores de incremento y decremento

- Incrementan / decrementan variable
- Operadores
  - Incremento `++`
  - Decremento `--`
- Modalidades
  - Postincremento / Postdecremento
    - `A++ / A--`
  - Preincremento / Predecremento
    - `++A / --A`

46

5.1.5. Operadores de asignación

- Asignación de un valor
  - De derecha a izquierda
    - lvalue = rvalue
- Operador
  - Asignación =
- Variedades
  - Uso múltiple
    - lvalue1 = lvalue2 = lvalue3 = rvalue
  - Modo agrupado
    - `+=, -=, *=, /=, %=`

47

5.1.6. Operadores de manejo de bits

- Manipulación de variables bit a bit
- Operadores
  - AND a nivel de bit &
  - OR a nivel de bit |
  - XOR a nivel de bit ^
  - Desplazamiento a la izquierda <<
  - Desplazamiento a la derecha >>
  - Desplazamiento derecha sin el signo >>>
  - Complemento a uno ~

48

5.1.7. Operador condicional

- Equivalente al *if* de control de flujo
- Sintaxis

```
expresion1 ? expresion2 : expresion3
```

  - Evalúa expresion1
    - Si es cierta, evalúa expresion2
    - Si es falsa, evalúa expresion3

49

5.2. Precedencia de operadores

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

Mayor precedencia

Menor precedencia

50

6. Sentencias de control

1. Sentencias condicionales
   1. Sentencia *if – else*
   2. Sentencia *switch*
2. Sentencias de iteración o repetición
   1. Sentencia *while*
   2. Sentencia *for*
   3. Sentencia *do – while*
3. Sentencias de salto
   1. Sentencia *break*
   2. Sentencia *continue*

51

6. Sentencias de control

- Sentencia
  - Instrucción completa
  - El ordenador es capaz de interpretarla
- Tipos de sentencia
  - Declaración `int valor;`
  - Asignación `valor = 7;`
  - Invocación `System.out.println("Hola");`
  - Control `if(valor==7) valor++;`
  - Nula `/* Comentario */`

52

6.1. Sentencias condicionales

- Sentencia *if – else*

```
if (condición)
   sentencia_1;
[else
   sentencia_2;]
```

```
if (condición){
   sentencias_1;
}
else{
   sentencias_2;
}
```

53

6.1. Sentencias condicionales

- Sentencia *if – else*
  - Anidamiento

```
if (condición) sentencia_1;
else if(condición2)    sentencia_2;
else if(condición3)    sentencia_3;
else if(condición4)    sentencia_4;
else if(condición5)    sentencia_5;
else if(condición6)    sentencia_6;
else sentencia_7;
```

54

6.1. Sentencias condicionales

- Sentencia *switch*

```
switch (expresión)
{
    case valor_1:
        sentencia;
        sentencia;
        ...
        [break;]
    case valor_2:
        sentencia;
        ...
        [break;]
    [default:
        sentencia;
        sentencia;]
}
```

Expresión puede ser short, int, byte o char, y desde JDK7, también Enum, String y Wrapper

55

6.2. Sentencias de iteración

- Sentencia *while*

```
while (condición){
   sentencia_1;
   sentencia_2;
}
```

56

6.2. Sentencias de iteración

- Sentencia *for*

```
for(expresion_1;expresion_2;expresion_3)
{
   sentencia_1;
   sentencia_n;
}
```

  - Donde
    - expresion_1 : asignación inicial de variables
    - expresion_2 : condición que permite seguir el bucle
    - expresion_3 : asignación para actualización de vbles

57

6.2. Sentencias de iteración

- Sentencia *do – while*

```
do{
   sentencia_1;
   sentencia_2;
} while (condición);
```

58

6.3. Sentencias de salto

- Sentencia *break*
  - Interrupción de la ejecución de bucles

```
break;
```

- Sentencia *continue*
  - Paso de control al final del cuerpo del bucle

```
continue;
```

59

7. Comentarios

1. De una línea
2. Multilínea
3. Javadoc

60

7. Comentarios

- Comentarios de línea

```
// Esto es un comentario de línea
```

- Comentarios multilínea

```
/* Aquí comienza el comentario
Esto sigue siendo comentario
No será tenido en cuenta
Aquí finaliza el comentario */
```

61

7. Comentarios

- Comentarios *javadoc*
  - Permiten generación automática de código
  - Formato del resto del API

```
/** Aquí comienza el comentario
Esto sigue siendo comentario
Será tenido en cuenta por javadoc
Aquí finaliza el comentario */
```

62

Unidad 3
Sintaxis básica de Java

Programación
1º D.A.M.

63