

UD2 .Introducció a JAVA. Sintaxis

Verónica Mascarós

Curso 23-24



```
e = m(b, " ");  
-1 < e && b.splice(e, 1);  
e = m(b, void 0);  
-1 < e && b.splice(e, 1);  
e = m(b, "");  
-1 < e && b.splice(e, 1);  
for (c = 0; c < d && c < b.length; c++) {  
    a += b[c].b + ", ", n.push(b[c].b);  
}  
for (g = 0; g < f; g++) {  
    e = Math.floor(b.length * g / f);  
    d.c + "</span></li>"), b[e] = b[e] + " " + b[e];  
}  
for (; c < b.length; c++) {  
    void 0 !== b[c] && ("para" + b[c] + " ");  
}  
function(b);  
single").h("mode_9" + b[c] + " ");  
ent(
```

Principis bàsics

- Els programes escrits a Java els executa la màquina virtual Java (JVM, Java Virtual Machine).
- La JVM és un programa especial que sap executar programes escrits a Java.
- En Java tenim 2 principis bàsics:
 1. En el llenguatge de programació Java, cada ordre s'escriu en la seva pròpia línia i ha de portar un punt i coma al final.
 2. Un programa només pot contenir ordres.

Principis bàsics

- Intentem entendre l'estructura d'un programa en Java.
- Imagineu-vos una habitació d'un apartament. Una habitació no pot existir per si sola. Forma part d'un apartament. Un apartament tampoc no pot existir per si sol. Forma part d'un edifici.
- Així, podem dir que l'edifici es divideix en apartaments i que cada apartament es divideix en habitacions.
- Una **ordre** és com una habitació. En el llenguatge de programació Java, una ordre no pot existir per si sola. És part d'una **funció** (a Java, les "funcions" també s'anomenen "mètodes"). Alhora, un mètode o funció (seria com l'apartament) forma part d'una **classe** (que seria l'edifici d'apartaments).
- En altres paraules, una classe es divideix en mètodes i els mètodes es divideixen en ordres.

Principis bàsics

- Els programes de Java es componen de classes.
- Un programa mínim té una sola classe. Per a cada classe, es crea un fitxer independent, el nom del qual coincideix amb el de la pròpia classe.

Suposem que decideixes crear una classe que descriga una llar. Hauràs de crear una classe Llar que es desarà al fitxer Llar.java

```
public class Hogar  
{  
    CUERPO DE LA CLASE  
}
```

El nom de les classes comença per majúscula

Principis bàsics

- El cos de la classe pot contenir variables (també conegudes com a dades) i mètodes (o funcions).

NOTA: També poden existir classes que no tinguen variables i que no tinguen mètodes.

```
public class Hogar
```

```
{
```

```
    VARIABLE A
```

```
    VARIABLE Z
```

```
    MÉTODO 1
```

```
    MÉTODO N
```

```
}
```

Però un **programa mínim** (no té cap ordre) ha de contenir com a mínim una classe que ha d'incloure almenys un mètode o funció perquè el programa s'execute.

Principis bàsics

- Exemple cos d'una classe i d'un programa mínim

```
public class Hogar
{
    int a; ← Variables de tipus
    int b; ← enter

    public static void main(String[] args)
    {
        System.out.print("1");
    }

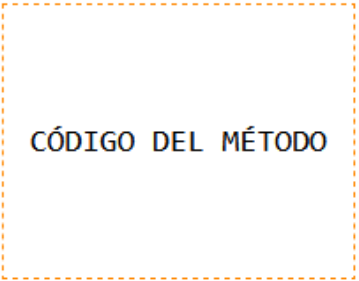
    public static double pi()
    {
        return 3.14;
    }
}
```

Mètodes

```
public class Hogar
{
    public static void main (String[] args)
    {
    }
}
```

Principis bàsics

- La classe que inicia el programa es pot anomenar de qualsevol manera, però el mètode "main" que s'utilitza per iniciar el programa sempre ha de tenir l'aspecte que es mostra a continuació:

```
public class Hogar
{
    // Parte inalterable
    public static void main(String[] args)
    {
        
        CÓDIGO DEL MÉTODO
    }
}
```

El cos d'un mètode es compon de comandes o **ordres**.

El llenguatge Java té ordres per a qualsevol ocasió, cadascuna de les quals descriu alguna acció. Al final de cada ordre, posem un punt i coma.

Principis bàsics

- Exemple: `System.out.println(1);`
- Aquesta ordre **mostra el número 1 a la pantalla**
- **Utilitzem parèntesis per passar arguments** a l'ordre. Depenent del valor d'aquests arguments, la mateixa ordre pot executar diferents accions.
- **Si es vol mostrar algun text** a la pantalla, **escriu-lo entre cometes dobles.**

Principis bàsics

- Alguns comandos tenen "versions". Per exemple: `System.out.println()` i `System.out.print()`
- En aquest cas, si utilitzeu l'ordre `System.out.println()` diverses vegades, veureu que els textos que passeu aniran mostrant-se en línies diferents. Però, si utilitzeu l'ordre `System.out.print()`, es mostraran tots a la mateixa línia.
- Concretament, `println()` imprimeix a la línia actual, però força que el text següent s'imprimisca en una altra línia.

Principis bàsics

- Les **variables** són entitats especials que es fan servir per emmagatzemar dades. Qualsevol dada. A Java, totes les dades s'emmagatzemen en variables.
- L'analogia més aproximada en aquest cas és una caixa. Suposem que escrivim el número 10 en un paper i el fem servir en una caixa. Ara podem dir que la caixa emmagatzema el valor 10.

Principis bàsics

- A Java, totes les variables tenen tres propietats importants: tipus, nom i valor.
 - Fem servir el nom per poder distingir una variable d'una altra. És com l'etiqueta d'una caixa.
 - D'altra banda, el tipus d'una variable determina quin tipus de valors o dades s'hi poden emmagatzemar. Posem ous a les caixes d'ous, bombons a les de bombons, etc.
 - Pel que fa al valor, és l'objecte, la dada o la informació concreta que s'emmagatzema a la variable.
- **Tot objecte a Java és un cert tipus.** Per exemple, n'hi ha numèrics sencers, numèrics decimals, de text, de tipus Gat, de tipus Llar, etc.
- Les variables també tenen un tipus i només poden emmagatzemar valors el tipus dels quals coincideix amb el seu propi.

Principis bàsics

- Per crear (o "declarar") una variable, fem servir el nom del tipus: `NomTipus nomVariable`
- Exemples:
- `int a;` --> Declara una variable de tipus *int* anomenada a
- `String s;` --> Declara una variable de tipus *String* anomenada s

Principis bàsics

- Recordem l'analogia de la caixa. Imagina que agafes un paper, escrius el número 42 i el fiques a la caixa. Ara la caixa emmagatzema el valor 42.
- Utilitzem una operació especial (anomenada **assignació**) per atorgar valors a les variables.
- L'assignació copia els valors d'una variable a una altra. No mou valors. Els copia. Com un fitxer que es copia en un disc.
- Per fer l'operació d'assignació, utilitzem el signe igual (=).

Principis bàsics

- El seu aspecte és com el que veus a continuació:

Código	Descripción
<pre>1 i = 3;</pre>	Asigna el valor 3 a la variable i .
<pre>1 a = 1; 2 b = a+1;</pre>	Asigna el valor 1 a la variable a . Asigna el valor 2 a la variable b .
<pre>1 x = 3; 2 x = x + 1;</pre>	Asigna el valor 3 a la variable x . En la siguiente línea, el valor de x aumenta en 1, por lo que es igual a 4.

Atenció! això no fa una comparació. Estem copiant el valor que hi ha a la dreta del signe igual a la variable que s'indica a l'esquerra. Per realitzar una comparació, Java utilitza un signe de doble signe igual (==)

Principis bàsics

- A tindre en compte:
- No pots crear dues variables amb noms idèntics dins del mateix mètode.
- Els noms de variables no poden contenir espais +, -, etc. El millor és fer servir només lletres i números per al nom de les variables.
- Java distingeix entre majúscules i minúscules. `int a` no és el mateix que `Int a`
- A Java pots crear una variable i assignar-li un valor alhora, per estalviar temps i espai. Exemples:

```
int a = 5;
int b = 6;
String s = "Soy Amigo";
int c = 7;
int d = c+1;
```