Apunts UD02 — Programació 1r DAM

1 – Introducció a Java

Títol

Inici Tema 2.

Programació 1r DAM · 2025-2026

Objectiu: entendre els fonaments de **Java** i practicar amb exemples i exercicis.

3. Introducció

UD02 · DAM · 2025

Què és Java

Java és un **llenguatge de programació** creat per *Sun Microsystems* (actualment **Oracle**) com a evolució de C++. Va nàixer orientat a dispositius però va esdevenir un llenguatge **multi-plataforma** gràcies a la **JVM (Java Virtual Machine)**.

Avantatge clau: un programa Java pot executar-se a qualsevol sistema on hi haja una JVM (Windows, macOS, Linux, Android…).

Per què usar Java

- **Portabilitat** (codi portable sense canvis).
- **Orientació a objectes** i **multitasca**.
- **Gestió d'errors** amb excepcions.
- Evita errors freqüents (no hi ha punters).
- Ecosistema ampli (biblioteques, IDEs, frameworks).

Quan no usar-lo

- Aplicacions que requerisquen **rendiment màxim nadiu** o on **no hi haja JVM** disponible.

Requisits

- Per **executar** programes → **JRE/JVM**.
- Per **programar** → **JDK (Java Development Kit)**.

2.Agenda

Contingut

Index del temari.

1. Introducció
2. Agenda
3. Introducció
4. Instal·lació del JDK
5. Instal·lació de Visual Studio Code
6. El primer programa: “Hola Món”
7. Variables i tipus
8. Comprovació de condicions
9. Bucles
10. Tipus de bucles en Java
11. Control de flux dins dels bucles
12. Arrays i cadenes
13. Dades introduïdes per l´usuari
14. Parse

Revisarem eines, sintaxi bàsica, tipus, E/S, condicions, bucles, funcions i col·leccions.

4. Instal·lació del JDK

UD02 · DAM · 2025

1. Descarrega i instal·la el **JDK** d'Oracle o *OpenJDK* segons el teu sistema.
2. Verifica a la terminal:

```
java -version
javac -version
```

3. Comprova que el compilador `javac` està en el PATH.

Visual Studio Code

IDE lleuger per a Java

Material docent – UD02

5. Instal·lació de Visual Studio Code

UD02 · DAM · 2025

- Descarrega **VS Code** i instal·la l'extensió **Java Extension Pack**.
- Recomanable revisar els vídeos d'introducció i activar el *Preview* de Markdown per a la documentació.

6. El primer programa: “Hola Món”

UD02 · DAM · 2025

Crea `HolaMundo.java`:

```
1   // Aplicació HolaMundo de exemple
2   class HolaMundo {
3       public static void main(String[] args) {
4           System.out.println("Hola Món!");
5       }
6   }
```

Compila i executa:

```
1   javac HolaMundo.java
2   java HolaMundo
```

4.1. Entendre el codi

- `class HolaMundo { … }` → defineix una **classe** (contenidor del codi).
- `public static void main(String[] args)` → punt d’entrada del programa.
- `System.out.println("…")` → escriu per pantalla i fa **salt de línia**.
- Cada **sentència** acaba amb `;`.

7. Variables i tipus

UD02 · DAM · 2025

7.1 Què són i com es declaren

```java
int primerNumero = 56;
int segonNumero = 23;
System.out.println(primerNumero + segonNumero); // 79
```

Regla: primer el **tipus**, després el **nom**. Es pot **inicialitzar** en la mateixa línia.

7.2 Tipus de dades primitives

Tipus	Decimals	Mida	Exemple
`byte`	No	1B	`byte edat = 25;`
`short`	No	2B	`short alt = 160;`
`int`	No	4B	`int suma = 100;`
`long`	No	8B	`long diners = 20000L;`
`float`	Sí	4B	`float preu = 2.5f;`
`double`	Sí	8B	`double pi = 3.1416;`
`char`	No	2B	`char lletra = 'A';`
`boolean`	No	1B	`boolean actiu = true;`

7.3 Operacions bàsiques i abreujades

- Matemàtiques: `+ - * / %`
- Relacionals: `> >= < <= == !=`

8. Comprovació de condicions

UD02 · DAM · 2025

Diagrama de flux

```
graph TD;
  A[Inici] --> B{Decisió}
  B -- Sí --> C[Acció]
  B -- No --> D[Altres accions]
  C --> E[Fins aquí]
  D --> E[Fins aquí]
```

if / else

```java
if (x == 3) {
    System.out.println("Correcte");
} else {
    System.out.println("Incorrecte");
}
```

switch

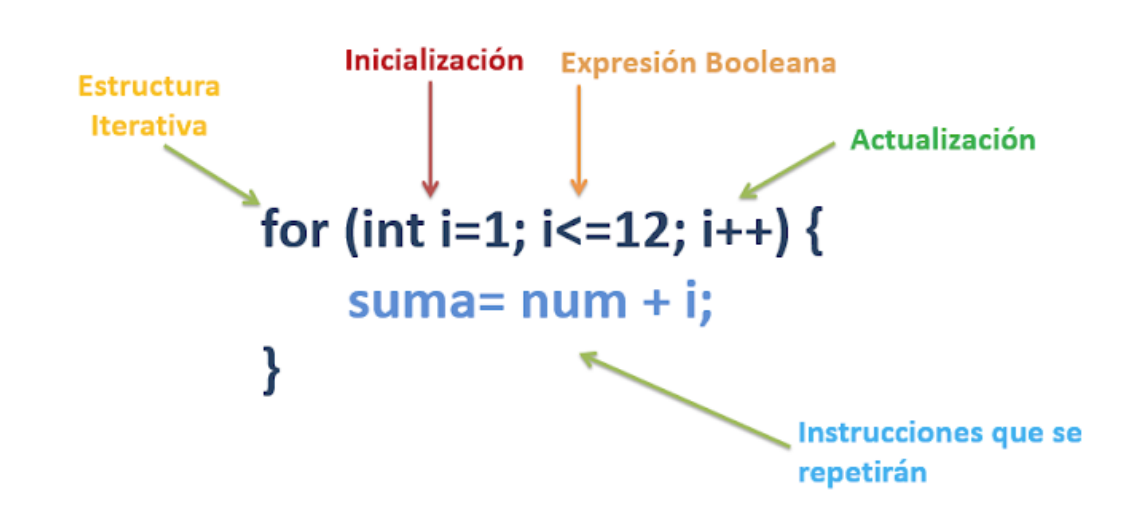
```java
switch (x) {
    case 1:
    case 2:
    case 3:
        System.out.println("Entre 1 i 3");
        break;
    case 4:
    case 5:
        System.out.println("4 o 5");
        break;
    default:
        System.out.println("Altres valors");
}
```

Operador condicional (ternari)

```java
int x = (a == 10) ? b * 2 : a;
```

9. Bucles

UD02 · DAM · 2025



while

```
int x = 5;
while (x > 0) {
    x--;
}
```

do...while

```
int op;
do {
    // mostra menú
    op = llegirOpcio();
} while (op != 0);
```

10.Tipus de bucles en Java

UD02 · DAM · 2025

1. `while` --- Bucle condicional

Executa el bloc **mentre la condició siga vertadera**.\ Primer avalua la condició i després executa el codi.

```
int i = 1;
while (i <= 5) {
    System.out.println("Iteración: " + i);
    i++;
}
```

Característiques: - La condició s'avalua **abans** de cada iteració.\ - Si la condició és falsa a l'inici, **no s'executa**.\ - S'utilitza quan **no sabem quantes vegades** es repetirà el procés.

---

2. `do...while` --- Bucle postcondicional

Executa el bloc **almenys una vegada**, i **després avalua la condició**.

```
int i = 1;
do {
    System.out.println("Iteración: " + i);
    i++;
} while (i <= 5);
```

Característiques: - La condición se evalúa **después** de cada ejecución.\ - Garantiza **al menos una iteración**.\ - Útil quan el bloc ha d'executar-se sempre una vegada (per exemple, demanar dades a l'usuari).

---

3. `for` --- Bucle controlat

Permet definir **inicialització**, **condició** i **actualització** en una mateixa línia.

11.Control del flux dins dels bucles

UD02 · DAM · 2025

`break`

Interromp el bucle immediatament, eixint d'ell.

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) break;
    System.out.println(i);
}
```

> El bucle s'atura quan `i` vale 5.

`continue`

Salta la iteració actual i **passa directament a la següent**.

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) continue;
    System.out.println(i);
}
```

> S'omet la iteració on `i == 3` .

Recomanacions

1. Evita els **bucles infinits** (`while(true)`) excepte que controles la seua eixida amb `break` .\
2. Utilitza **noms clars** para las variables de control (`i` , `contador` , `indice`).\
3. Assegura't d'**actualitzar la variable de control** dins del bucle per a evitar bloquejos.\
4. Practica amb diferents tipus de condicions per a entendre com canvia el flux.

12. Arrays i cadenes

UD02 · DAM · 2025

12.1 Arrays

```java
int[] a = {10, 20, 30};
System.out.println(a[1]); // 20

double[] dades = new double[1000];
dades[0] = 5.5;
```

Un **array** es una **estructura de datos** que almacena **varios valores del mismo tipo** bajo un solo nombre.\ Cada valor ocupa una **posición (índice)** que empieza en **0**.

Declaració i creació

```java
int[] numeros = new int[3];
String[] nombres = {"Ana", "Luis", "María"}; // Inicialització directa
```

- Els valors s´accedeixen per índex: `nombres[0]`, `nombres[1]`, etc.\
- Els arrays tenen **tamany fixe**.\
- Propietat útil: `array.length` torna el tamany.

12.2 Accés i modificació

```java
numeros[0] = 10;
System.out.println(numeros[0]);
```

13. Dades introduïdes per l'usuari

UD02 · DAM · 2025

13.1 Classe `Scanner`

```java
import java.util.Scanner;

Scanner entrada = new Scanner(System.in);
System.out.print("Nom: ");
String nom = entrada.nextLine();
System.out.println("Hola, " + nom);
```

13.2 Netejar el buffer

Quan llegim un enter i després un `String`, cal consumir el `\n` pendent:

```java
int n = entrada.nextInt();
entrada.nextLine(); // neteja el \n
String nom = entrada.nextLine();
```