

Activitat 1.2: Llenguatges interpretats vers compilats. Instal·lació de Java i Go en Windows

Descripció

- Versions de JAVA (Oracle, OpenJDK) i Go
- Mètodes d'instal·lació.
- Instal·lació i variables d'entorn del sistema
- Compilació i execució des de la línia d'ordres (PowerShell, Shell)

Objectius

- Saber instal·lar Java i Go en Windows
- Saber executar programes en Java i Go en l'entorn Windows
- Entendre l'execució de programes compilats i interpretats

REPÀS DE COMANDES DE SISTEMA

Entrar a la línia de comandos, a cerca aplicacions escribiu: **cmd** (o PowerShell)

Per veure tots els directoris i fitxers de la carpeta: **dir** (ls mac i linux)

Per entrar dins una carpeta és: **cd nom_carpeta**

Per tirar endarrere és: **cd ..**

- Visualitzar les variables d'entorn: **set**
- Veure el valor d'una variable específica: **echo %PATH%**
- Definir una variable d'entorn temporalment: **set MY_VARIABLE=HelloWorld**
- Eliminar una variable d'entorn, set sense assignar valor: **set MY_VARIABLE=**
- Modificar variables d'entorn de manera permanent: **setx MY_VARIABLE HelloWorld**
- Modificar el PATH des de la línia d'ordres:
 - Per afegir un nou directori al PATH temporalment:
set PATH=%PATH%;C:\nou_directori
 - Per afegir-lo permanentment: **setx PATH "%PATH%;C:\nou_directori"**

Més comandes:

[33 comandos básicos para dar tus primeros pasos en la consola de Windows \(CMD\)](#)

Generalitat de Catalunya Departament d'Educació Institut Caparrella	0487. Entorns de desenvolupament
	Activitat 1.2: Instal·lació de Java en Windows

APARTAT A: Instal·lació del JDK de Java a partir del codi font.

Instal·lació de JDK de Java d'Oracle

A la pàgina web de JAVA podem descarregar diferents versions de JDK. Per descarregar les versions més antigues com la 8 o 11 t'has de donar d'alta com usuari a Oracle.

<https://www.oracle.com/es/java/technologies/downloads/>

Vosaltres en instal·lar els IDE's normalment s'instal·la automàticament la versió més nova. Per fer proves instal·larem la [versió JDK 17](#).

Passos per la instal·lació:

- Anar al lloc de descàrrega, com podeu observar teniu 3 opcions
 - x64 Compressed Archive
 - [x64 Installer](#)
 - [x64 MSI Installer](#)
- (Aquest apartat no el farem)** Instal·lar a partir de x64 Installer (o MSI Installer) només si no teniu cap versió de JAVA instal·lada com a programa.
 - Descarregar el programa executable, per defecte JAVA s'instal·la a `C:\Program Files\Java`, serà el lloc on s'aniran instal·lant totes les versions.
 - Si obriu el CMD comproveu la versió **>java -version**, veureu que no s'executa caldrà modificar les variables d'entorn.
 - Creeu la variable d'entorn de sistema i afegiu l'adreça d'instal·lació
JAVA_HOME = C:\Program Files\Java\jdk-17
 - Afegiu a la variable **PATH** → **C:\Program Files\Java\jdk-17\bin**
 - Obriu un nou CMD i proveu d'executar **>java -version**
- Instal·lació a partir del Compressed Archive. Instal·lacions noves i en cas que vulgueu tenir més d'una versió de JDK.
 - Descarregar l'arxiu comprimit
 - Descomprimir-lo a **C:\Program Files\Java**
 - Modificar les variables d'entorn com el punt anterior

Cal afegir la variable de sistema

JAVA_HOME = C:\Program Files\Java\jdk-17

Afegir a la variable d'entorn PATH l'adreça dels executables

PATH → C:\Program Files\Java\jdk-17\bin

Generalitat de Catalunya Departament d'Educació Institut Caparrella	0487. Entorns de desenvolupament
	Activitat 1.2: Instal·lació de Java en Windows

APARTAT B: Instal·lació de Go a partir del codi font.

El compilador de Go l'instal·lareu a partir del paquet d'instal·lació per Windows msi. Llegiu la documentació de com instal·lar-lo i com saber la versió des de la línia d'ordres.

- Fixeu-vos en els paquets d'instal·lació per cada Sistema Operatiu

Web de Go: [The Go Programming Language](https://golang.org/)

APARTAT C: Proves d'execució d'un programa en Java i en Go

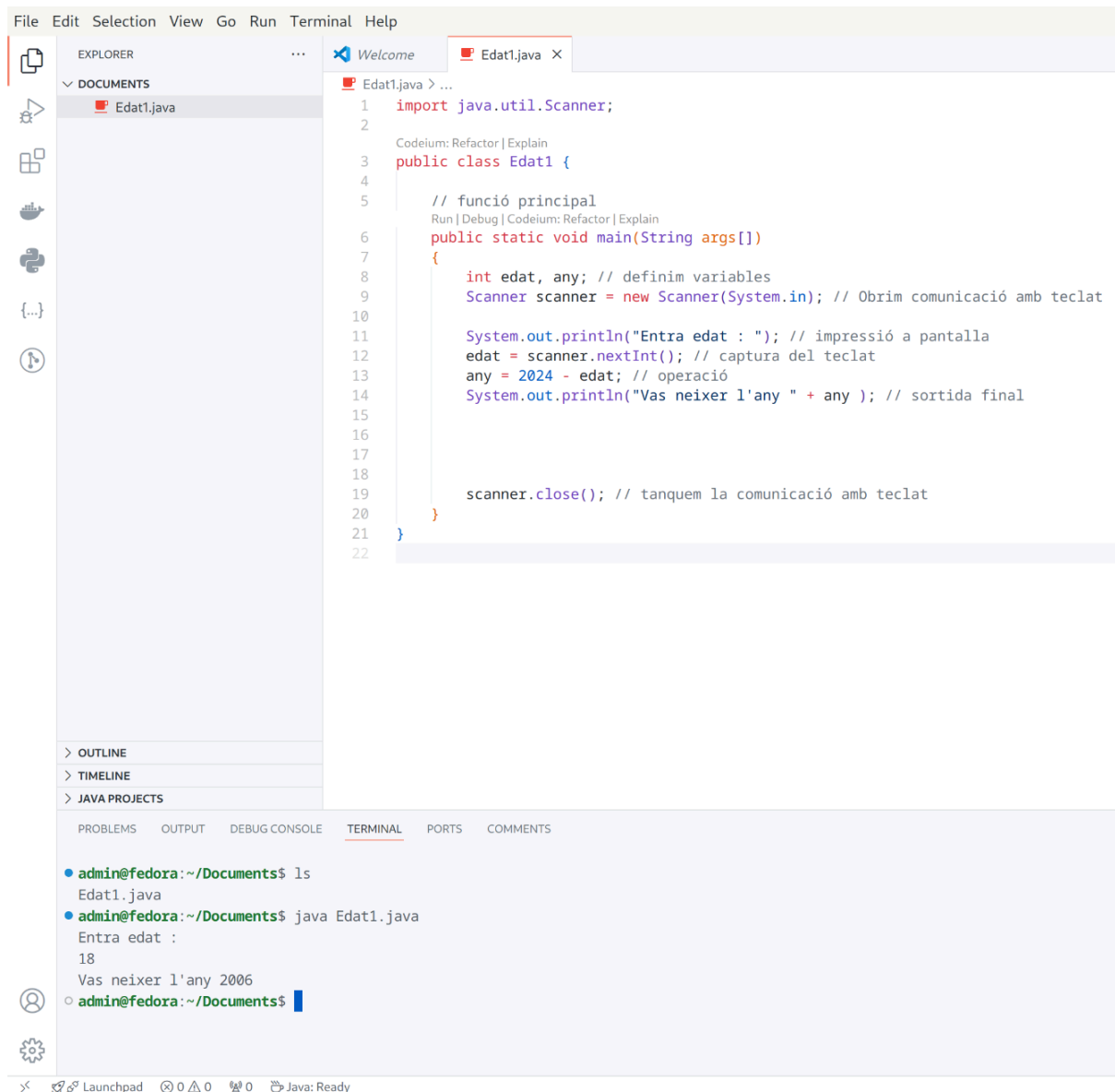
Per comprovar la velocitat d'execució d'un programa interpretat en Java i un compilat en Go, ho farem programant un bucle.

Programa en JAVA

1. Interpretar el codi directament a la MV

Copieu el codi a Geany i guardeu-lo com a **Edat1.java**

Interpreta i executa el programa de la següent forma: **C:> java Edat1.java**



The screenshot shows an IDE with the following components:

- EXPLORER:** Shows a project structure with a folder 'DOCUMENTS' containing a file 'Edat1.java'.
- EDITOR:** Displays the code for 'Edat1.java'. The code is as follows:

```
1 import java.util.Scanner;
2
3 public class Edat1 {
4
5     // funció principal
6     public static void main(String args[])
7     {
8         int edat, any; // definim variables
9         Scanner scanner = new Scanner(System.in); // Obrim comunicació amb teclat
10
11         System.out.println("Entra edat : "); // impressió a pantalla
12         edat = scanner.nextInt(); // captura del teclat
13         any = 2024 - edat; // operació
14         System.out.println("Vas neixer l'any " + any ); // sortida final
15
16
17
18
19         scanner.close(); // tanquem la comunicació amb teclat
20     }
21 }
22
```
- TERMINAL:** Shows the execution of the program. The commands and output are:

```
admin@fedora:~/Documents$ ls
Edat1.java
admin@fedora:~/Documents$ java Edat1.java
Entra edat :
18
Vas neixer l'any 2006
admin@fedora:~/Documents$
```

2. Compilar i generar ByteCode i executar-lo a la MV

Construïm el bytecode de Java generem Edat.class : **C:> javac Edat1.java**

Executeu el programa: **C:> java Edat1**



The screenshot shows a terminal window titled 'admin@fedora:~/Documents' with the following content:

```
admin@fedora:~/Documents$ javac Edat1.java
admin@fedora:~/Documents$ java Edat1
Entra edat :
18
Vas neixer l'any 2006
admin@fedora:~/Documents$
```

```
import java.util.Scanner;

public class Edat1 {

    // funció principal
    public static void main(String args[])
    {
        int edat, any; // definim variables
        Scanner scanner = new Scanner(System.in); // Obrim comunicació amb teclat

        System.out.println("Entra edat : "); // impressió a pantalla
        edat = scanner.nextInt(); // captura del teclat
        any = 2024 - edat; // operació
        System.out.println("Vas neixer l'any " + any ); // sortida final

        scanner.close(); // tanquem la comunicació amb teclat
    }
}
```

Un programa en Go

Copieu el codi a Geany i guardeu-lo com a **Edat2.go**

Construïu l'executable de la següent forma: **C:> go build Edat2.go**

Executeu el programa: **C:> Edat2.exe**

```
package main

import "fmt"

// funció principal
func main() {
    // declaració de variables enteres
    var any int
    var edat int

    fmt.Print("Introdueix la teva edat: ") // Imprimir a pantalla
    fmt.Scanf("%d", &edat) // Obtenir dades numèriques pel teclat
    any = 2024 - edat // Operació resta
    fmt.Printf("Vas neixer l'any %d anys.\n", any) // sortida
}
```



The screenshot shows a terminal window titled "admin@fedora:~/Documents" with standard window controls. The terminal output is as follows:

```
admin@fedora:~/Documents$ go build Edat2.go
admin@fedora:~/Documents$ ./Edat2
Introdueix la teva edat: 18
Vas neixer l'any 2006 anys.
admin@fedora:~/Documents$
```

APARTAT D: Quin s'executa més ràpidament?, els programes compilats o interpretats.

Comproveu el temps d'execució dels 2 codis Java i Go. Els programes consten de bucles anidats.

JavaBucle.java

```
public class JavaBucle {  
  
    // funció principal  
    public static void main(String args[])  
    {  
        double maxim = 10000000000;  
        double j;  
  
        System.out.println(" Execució ... 0% ");  
        for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ } } }  
        System.out.println(" Execució ... 25% ");  
        for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ } } }  
        System.out.println(" Execució ... 50% ");  
        for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ } } }  
        System.out.println(" Execució ... 75% ");  
        for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ for(j=0;j<maxim;j++){ } } }  
        System.out.println(" Execució ... 100% ");  
    }  
}
```

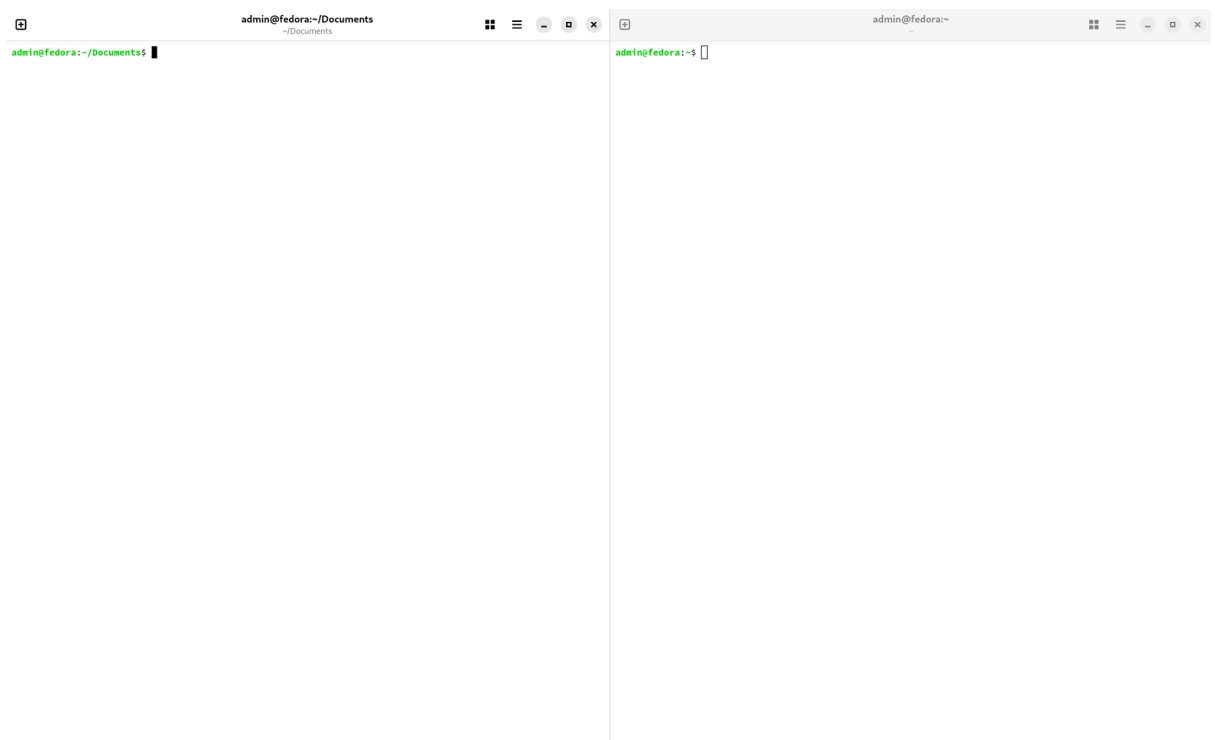
GoBucle.go

```
package main  
  
import "fmt"  
  
func main() {  
    var maxim float64 = 10000000000  
    var j float64  
  
    fmt.Println(" Execució ... 0% ")  
    for j=0 ;j<maxim ;j++){ for j=0 ;j<maxim ;j++ { for j=0 ;j<maxim ;j++{ } } }  
    fmt.Println(" Execució ... 25% ");  
    for j=0 ;j<maxim ;j++){ for j=0 ;j<maxim ;j++ { for j=0 ;j<maxim ;j++{ } } }
```

```
        fmt.Println(" Execució ... 50% ");  
        for j=0 ;j<maxim ;j++{ for j=0 ;j<maxim ;j++ { for j=0 ;j<maxim ;j++{ } }  
        fmt.Println(" Execució ... 75% ");  
        for j=0 ;j<maxim ;j++{ for j=0 ;j<maxim ;j++ { for j=0 ;j<maxim ;j++{ } }  
        fmt.Println(" Execució ... 100% ")  
    }  
}
```

Com dur a terme la prova:

1. Obriu dos terminals de Windows CMD o PowerShell



2. Genereu el bytecode en Java: `javac JavaBucle.java`
3. Compileu el programa en Go: `go build GoBucle.go`
4. Comproveu que teniu generats tant el bytecode de Java com el programa executable de Go
5. En els 2 terminals oberts:
 - Terminal 1: preparat per executar el bytecode de Java: `java JavaBucle`
 - Terminal 2: preparat per executar el programama compilat: `GoBucle.exe`

The image shows two terminal windows side-by-side, both running on a Fedora system (admin@fedora). The left window shows the compilation and execution of a Go program named 'GoBucle.go'. The right window shows the compilation and execution of a Java program named 'JavaBucle.java'.

Left Terminal (Go):

```
admin@fedora:~/Documents — java JavaBucle
~/Documents

admin@fedora:~/Documents$ go build GoBucle.go
admin@fedora:~/Documents$ javac JavaBucle.java
JavaBucle.java:1: error: repeated modifier
public public class JavaBucle {
^
JavaBucle.java:28: error: class, interface, enum, or record expected
{
^
2 errors
admin@fedora:~/Documents$ javac JavaBucle.java
admin@fedora:~/Documents$ java JavaBucle
Execució ... 0%
Execució ... 25%
Execució ... 50%
```

Right Terminal (Java):

```
admin@fedora:~/Documents
~/Documents

admin@fedora:~/Documents$ cd Documents
admin@fedora:~/Documents$ ./GoBucle
Execució ... 0%
Execució ... 25%
Execució ... 50%
Execució ... 75%
Execució ... 100%
admin@fedora:~/Documents$
```

Left Terminal (Go - continued):

```
admin@fedora:~/Documents
~/Documents

admin@fedora:~/Documents$ go build GoBucle.go
admin@fedora:~/Documents$ javac JavaBucle.java
JavaBucle.java:1: error: repeated modifier
public public class JavaBucle {
^
JavaBucle.java:28: error: class, interface, enum, or record expected
{
^
2 errors
admin@fedora:~/Documents$ javac JavaBucle.java
admin@fedora:~/Documents$ java JavaBucle
Execució ... 0%
Execució ... 25%
Execució ... 50%
Execució ... 75%
Execució ... 100%
admin@fedora:~/Documents$
```

Right Terminal (Java - continued):

```
admin@fedora:~/Documents
~/Documents

admin@fedora:~/Documents$ cd Documents
admin@fedora:~/Documents$ ./GoBucle
Execució ... 0%
Execució ... 25%
Execució ... 50%
Execució ... 75%
Execució ... 100%
admin@fedora:~/Documents$
```

6. Quin dels dos s'executa més reapidament en el computador? Perquè ?
S'executa més ràpid el programa generat en go, ja que es compila directament a codi màquina. Java és més lent per tenir una màquina virtual que tradueix el bytecode a codi màquina en temps real, segueix sent més ràpid que llenguatges interpretats, però no es compara amb llenguatges compilats

Generalitat de Catalunya Departament d'Educació Institut Caparrella	0487. Entorns de desenvolupament
	Activitat 1.2: Instal·lació de Java en Windows

7. Enumera els avantatges i inconvenients dels programes compilats vers interpretats.

Java		Go	
Avantatges	Inconvenients	Avantatges	Inconvenients
Fàcilment portable	Més lent que un llenguatge purament compilat	Compilat	S'ha de tornar a compilat per utilitzar-lo en altres plataformes
Molt utilitzat	Té competència amb llenguatges com Rust	Ràpid	És un llenguatge nou
És un llenguatge que porta temps ja utilitzant-se	No es pot fer una gestió manual de memòria	És un llenguatge senzill d'utilitzar, no fa falta descarregar mòduls necessaris per fer funcionar el llenguatge	No té tantes ofertes de treball com altres llenguatges
Més fàcil trobar treball per aquest llenguatge	No té tipus primitius del tipus Unsigned com c o c++	És més senzill d'utilitzar que C o C++	No té tants mòduls ni llibreries com altres llenguatges
Té molt support	Es necessita un hardware més potent que C o C++	És un llenguatge que s'actualitza seguidament	No és un llenguatge tant madur com C/C++ i Java
Fa gestió automàtica de memòria	Necessita més memòria ja que utilitza una màquina virtual i no es compila directament	És fàcil de programar en go	No està orientat a objectes (almenys no tant com C++ o Java)

OPCIONAL: Podeu fer les mateixes proves en Python.

En Python és encara més lent ja que s'interpreta el codi totalment sense convertir-lo en bytecode de la mateixa forma que en java