

Programació



UT1.1 Estructura bàsica d'un
programa informàtic.



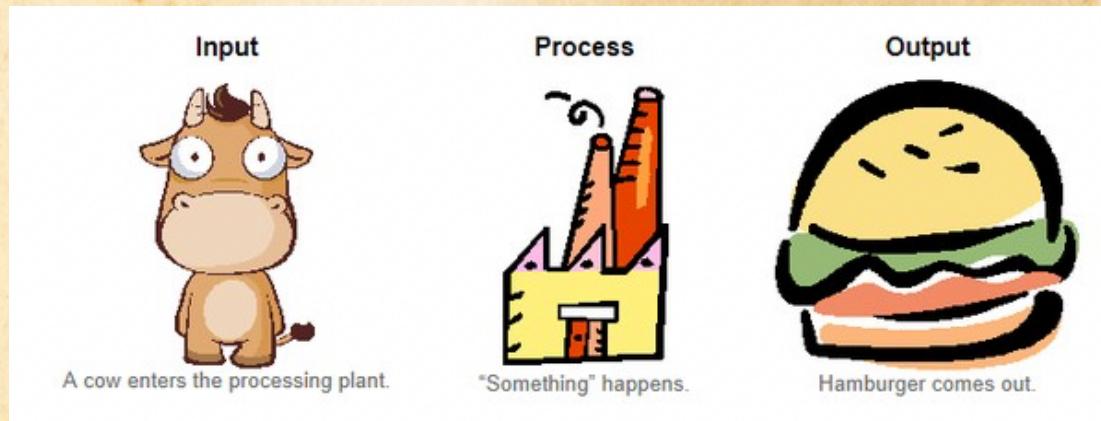
Iniciació a la programació

- Quantes accions de les que has realitzat hui, creus que estan relacionades amb la programació?



Iniciació a la programació

- un **ALGORISME** és la seqüència **ordenada** de passos, descrita **sense ambigüïtats**, que condueixen a la **solució** d'un problema donat.
- No té perquè estar relacionat amb els ordinadors, per exemple preparar una recepta.
- La resolució del problema parteix d'unes dades d'entrada, una seqüència de passos o instruccions i unes dades d'eixida.

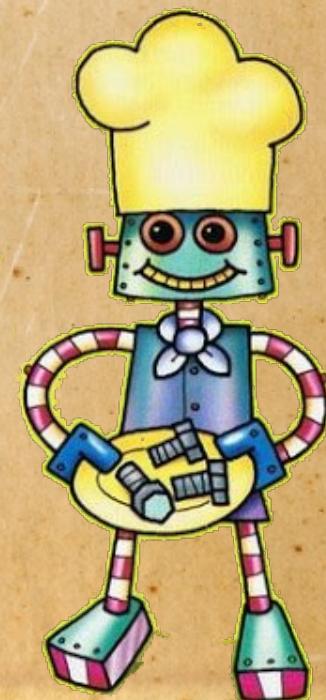


Iniciació a la programació

- Quan la realització d'esta sèrie de passos és efectuada per un ordinador (o altres dispositius electrònics) s'anomena algorisme computacional.
- En un **PROGRAMA** els passos que permeten resoldre el problema, s'han d'escriure en un determinat **llenguatge de programació** perquè puguen ser executats en l'ordinador i així obtindre la solució.
- En essència, **tot problema es pot descriure mitjançant un algorisme** i les característiques fonamentals que estos han de complir són:
 - Ha de ser precís i indicar l'ordre de realització pas a pas.
 - Ha d'estar definit. Si s'executa dos o més vegades, ha d'obtindre el mateix resultat cada vegada.
 - Ha de ser finit, ha de tenir un nombre finit de passos.

Exemple: programa d'un robot de cuina

- Recepta: **crema de panís**
 - Detectar que al robot hi haja farina de panís i mantega
 - Remoure durant 1 minut, incrementant la velocitat progressivament de l'1 al 5
 - Detectar que al robot hi haja llet i sal
 - Remoure durant 30 segons a velocitat 7
 - Remoure durant 10 minuts a velocitat 3 amb temperatura de 90 °C
 - Parar el robot. La crema ja està preparada.



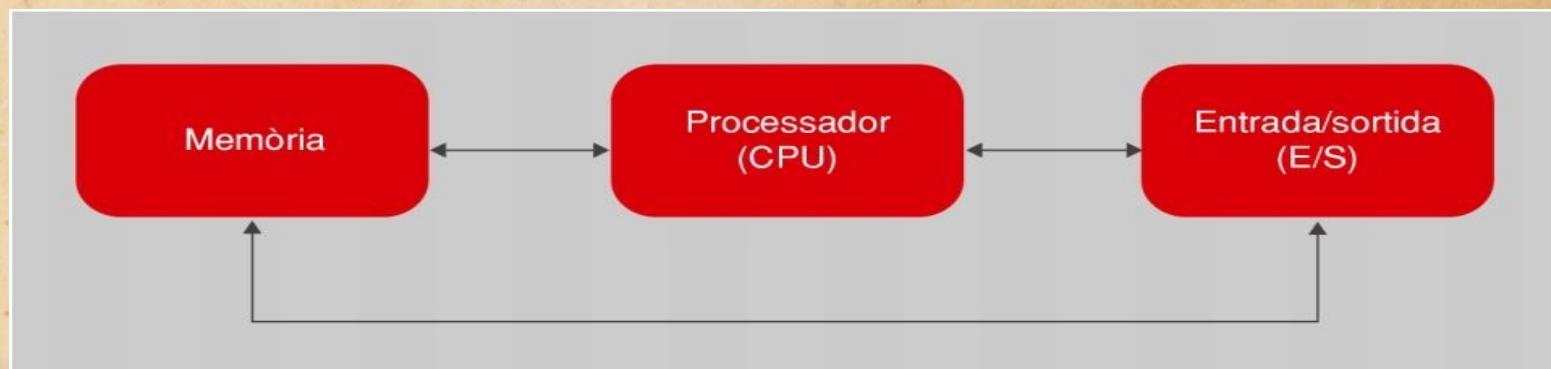
Faena del programador

- Per automatitzar, s'haurà triar
 - Quines ordres
 - En quin ordre
 - Sobre quines dades
- Complexitat, depèn de les tasques
- Eficàcia i eficiencia són conceptes distints

Conclusió: L'ordinador no ho fa tot

Tipus d'ordres que un ordinador accepta

- Limitades per les capacitats dels components. (Ex: Una rentadora no pot gratinar). Informació necessària per a saber què pot fer.
- Cada ordre està vinculada a estos tres parts de l' ordinador

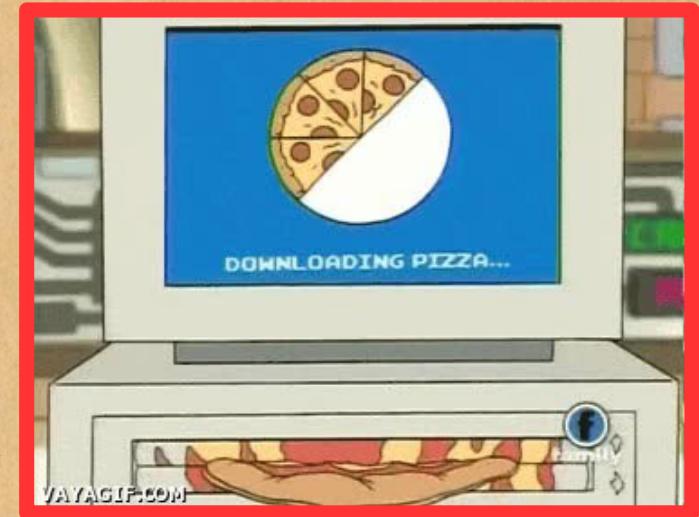


Què fan?

- **Processador:** gestionar ordres de manipulació i transformació de les dades.
- **Memòria:** emmagatzema les dades que el processador utilitzarà o ha processat. (RAM o ROM)
- **E / S:** intercanvi de dades amb l'exterior (teclat, monitor, etc ...)

Comparativa: Ordinador vs Pizzeria

- Cuiner manipula ingredients (processador)
- Frigorífic emmagatzema ingredients (Memòria)
- Motorista, cambrer, telèfon (E / S)
- Recepta (Programa)



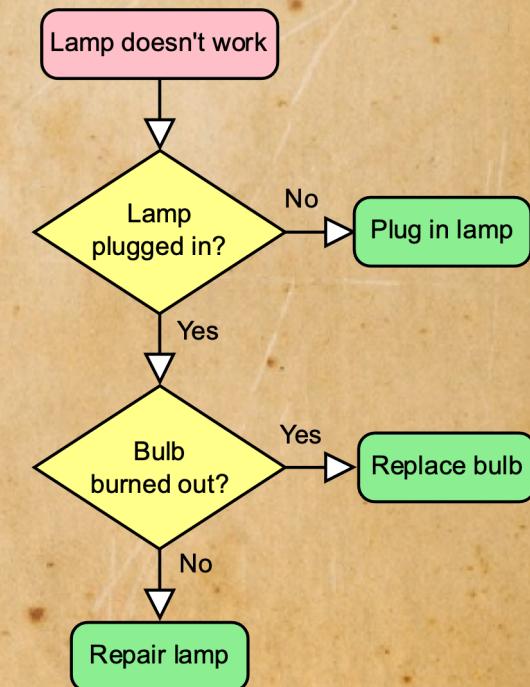
Multiplicar dos nombres (llenguatge natural)

ordre	element que ho realitza
Llegir un nombre per teclat	E / S (Teclat)
Guardar el nombre en memòria	memòria
Llegir un nombre per teclat	E / S (Teclat)
Guardar el nombre en memòria	memòria
Recuperar els dos nombres guardats i realitzar l'operació	processador
Guardar el resultat a la memòria	memòria
Mostrar el resultat	E / S (Pantalla)

Disseny de l'algorisme

- Podem dissenyar algoritmes de forma genèrica utilitzant pseudocodi o bé mitjançant un organograma o diagrama de flux.

```
inici
    escriure("Introdueix el primer enter: ")
    llegir(a)
    escriure("Introduix el segon enter: ")
    llegir(b)
    c <- a + b
    escriure("La suma és", c)
fi
```



Llenguatges de programació

- Llenguatge artificial dissenyat per a crear algoritmes que puguen ser utilitzats per un ordinador
- Multitud de llenguatges, cadascun amb la seua sintaxi (similar als idiomes)
- Les ordres seran **instruccions**
- Les instruccions s'emmagatzemen en fitxers

Llenguatges de programació



Crear un llenguatge:

- Dissenyar-lo (definir sintaxi, tipus de dades...)
- Implementar un compilador (o intèrpret)

Llenguatges de programació

- Cada llenguatge té les seues pròpies característiques.
- Podem classificar-los de moltes maneres:
 - Llenguatge **compilat o interpretat** (passos que segueix per obtindre un executable)
 - Llenguatge de **alt o baix nivell** (llunyania o proximitat amb el llenguatge màquina que un ordinador entén)

Llenguatges de programació

Llenguatges compilats

- En la compilació es tradueix tot el codi (codi font) a un llenguatge que la màquina entenga (llenguatge màquina) i per això solen ser més ràpids.
- Gràcies al **compilador** es tradueix el **codi font** i es **crea un arxiu executable**.
- L'arxiu executable **no serà multiplataforma**, i de voler executar en una altra plataforma s'haurà de tornar a compilar en esta plataforma.
- En cas d'existir un **error en el codi**, El compilador ho farà saber (i no es compilàrà el programa).

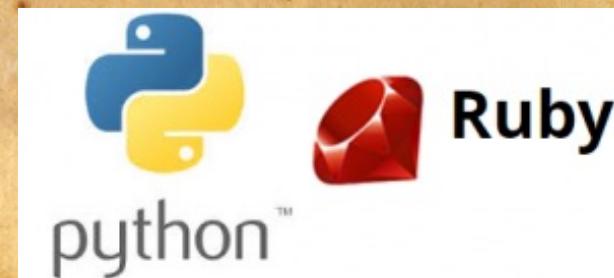


Llenguatges de programació



Llenguatges interpretats

- Els llenguatges interpretats són multiplataforma, per això són més portables però es requereix d'un intèrpret.
- Tot això succeeix a l'instant (al vol), ja que no es guarda la traducció de l'intèrpret, és a dir, es tradueix cada línia de codi de forma ordenada cada volta que s'executa (no es genera un fitxer executable),

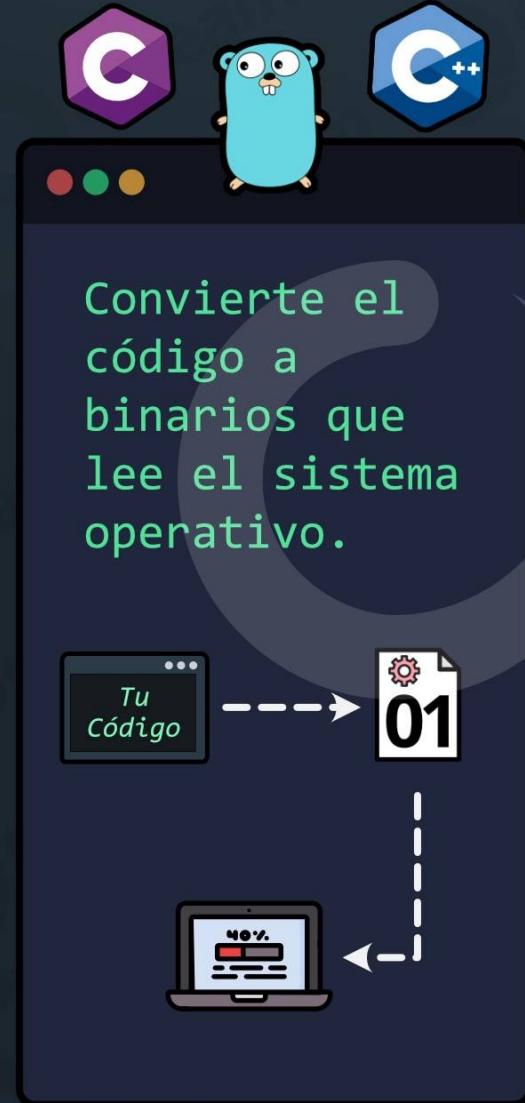


Llenguatges de programació

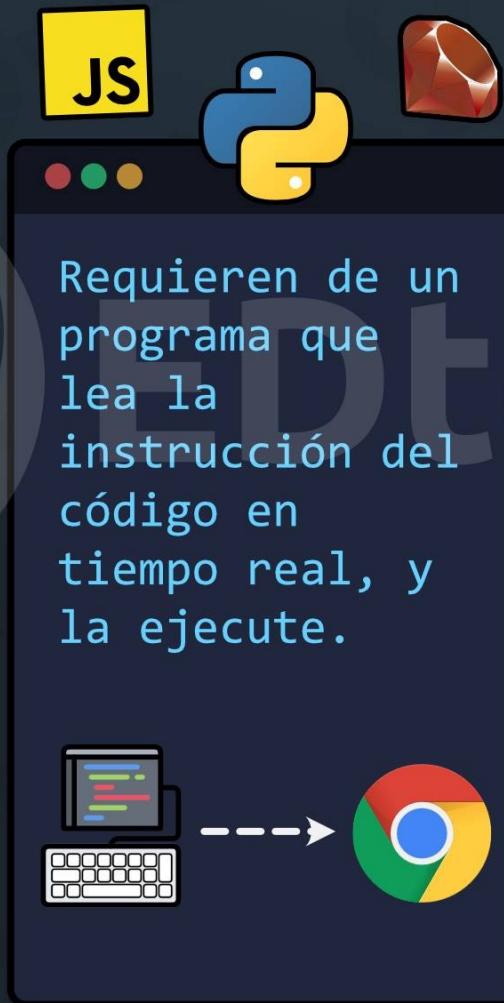
- **Java** és un llenguatge particular perquè és **compilat**, Però és compilat a un llenguatge intermedi anomenat **bytecode**, que després és interpretat. Els creadors de Java volien crear un llenguatge compilat, però que es poguera executar en qualsevol sistema operatiu i procesador **sense necessitat de crear diversos executables**.
- Per executar codi Java cal instal·lar el **JRE** (Java Runtime Environment), Que és el programa que s'encarrega d'interpretar el bytecode al que són compilats els programes de Java.
`(java -version)`
- Per a compilar codi Java no és suficient instal·lar el JRE, necessites el **JDK** (Java Development Kit) que inclou el compilador, entre d'altres eines de desenvolupament.
`(javac -version)`

TIPOS DE LENGUAJE

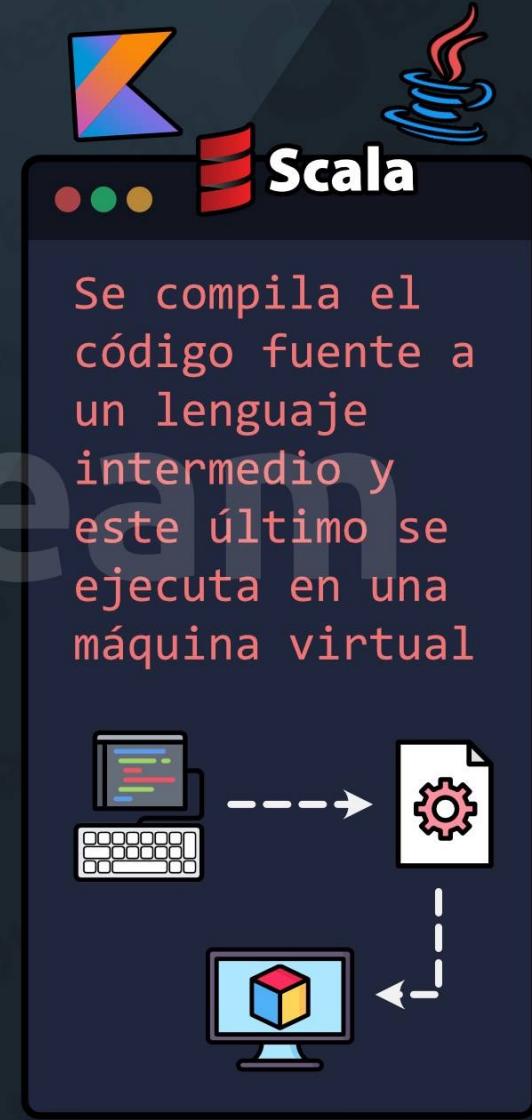
COMPILADO



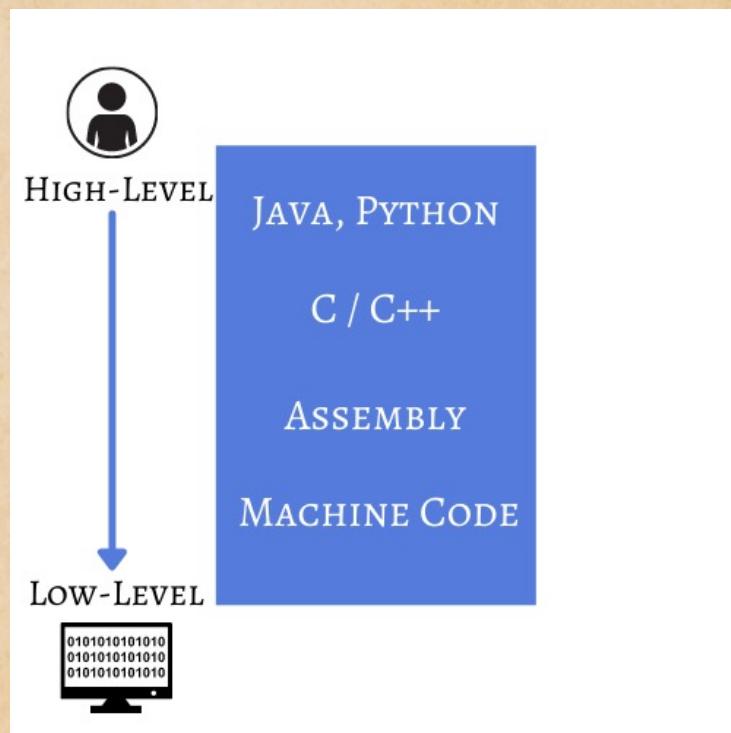
INTERPRETADO



INTERMÉDIO



Llenguatges de programació



Llenguatge de programació d'alt nivell: la seva característica principal consisteix en una estructura sintàctica i semàntica llegible, tenint en compte les capacitats cognitives dels éssers humans (*més abstracte*)

Llenguatge de programació de baix nivell: és aquell en el qual les seves instruccions exerceixen un control directe sobre el maquinari i estan condicionats per l'estructura física dels ordinadors que el suporten (*més específic*)

Llenguatges de programació



Llenguatges de baix nivell

LLENQUATGE
MÀQUINA



- Instruccions en binari, octal o hexadecimal.
- Cada microprocessador té el seu propi llenguatge màquina.

LLENQUATGE
ASSEMBLADOR

```
STA FFE7h
MVI KV,A
STA FFE8h
MVI KI,A
STA FFE9h
MVI KO,A
STA FFEAh
MVI KN,A
STA FFEBh
```

- Proper a l' llenguatge màquina. Substitueix el codi binari per operacions simples (ADD, INC, DEC, MOV ...)
- Ha de ser traduït (ensamblat) a cod. Màquina.

Llenguatges de programació



Llenguatges d'alt nivell

- Especificació d'una gramàtica **més similar a el llenguatge natural** el que facilita al programador crear programes complexos.
- L'ordinador haurà de **traduir** cada instrucció o estructura del llenguatge de programació en una seqüència molt més llarga d'instruccions en codi màquina.
- Existeix gran varietat: C, C ++, C #, Basic, Pascal, Ada, Clipper, COBOL, Delphi, Fortran, FoxPro, Java, JavaScript, Python, Ruby, Logo, PHP, Swift, Scratch...
- Cada llenguatge no és ni millor ni pitjor que un altre. Moltes voltes estan pensats per resoldre un tipus concret de problema o per cert àmbit.



Exemple de programa

NOTA: No et preocupes si no entens el codi. No cal que el comprehensiones ni que el proves. Es posen els següents exemples per a fer una comparativa visual del mateix programa en distints llenguatges de programació.



suma.py

```
1  enter1 = input("Escriu el primer enter: ")
2  enter2 = input("Escriu el segon enter: ")
3  resultat = int(enter1) + int(enter2)
4  print("La suma és: ", resultat)
```

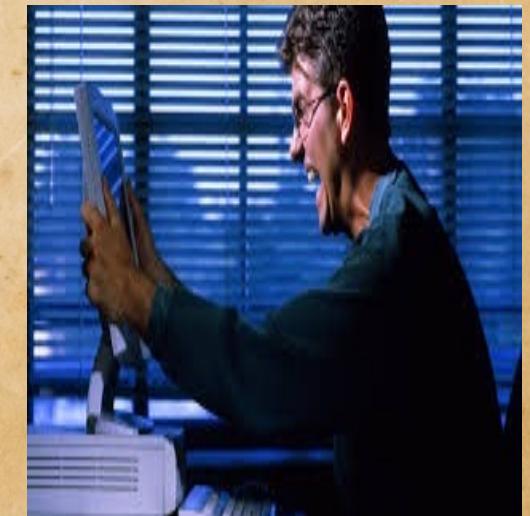
Exemple de programa

```
J Suma.java ×

1 import java.util.Scanner;
2
3 public class Suma {
4     Run | Debug
5     public static void main(String[] args) {
6         Scanner lector = new Scanner(System.in);
7         int enter1, enter2, resultat;
8
9         System.out.print("Introdueix el primer valor: ");
10        enter1 = lector.nextInt();
11        System.out.print("Introdueix el segon valor: ");
12        enter2 = lector.nextInt();
13        resultat = enter1 + enter2;
14        System.out.println("La suma es: " + resultat);
15
16        lector.close();
17    }
18}
```

Errors de compilació

- Errors de sintaxi en el codi font
- Impossible de tractar pel compilador
- Normalment, s'avisa el programador d'on està el error
- **Un programa sense errors de compilació NO ASSEGURA que realitze correctament la seua missió.**



Llenguatge natural vs Llenguatge de programació

- Frase "*vull ich helado un*" (Impossible d'entendre)
- Frase "*el gelat condueix un full de paper*" (gramaticalment correcta però no té cap sentit)
- Els llenguatges de programació reaccionen igual (tenen la seuia **sintaxi** i la seuia **semàntica**)

Entorns integrats de desenvolupament (IDE)

- Desenvolupament i execució de programes:
Editor i Compilador (inclou Linker) o Intèrpret.
- Un IDE facilita la tasca i inclou ambdues eines.



Apache
NetBeans IDE



Abans de començar el teu primer programa

- Decidir quin llenguatge utilitzem
 - ¿Java, C ++, JavaScript, Python?
- Aprendre a programar en un llenguatge concret facilita l'aprenentatge d'altres.
- El llenguatge en este curs serà **Java**



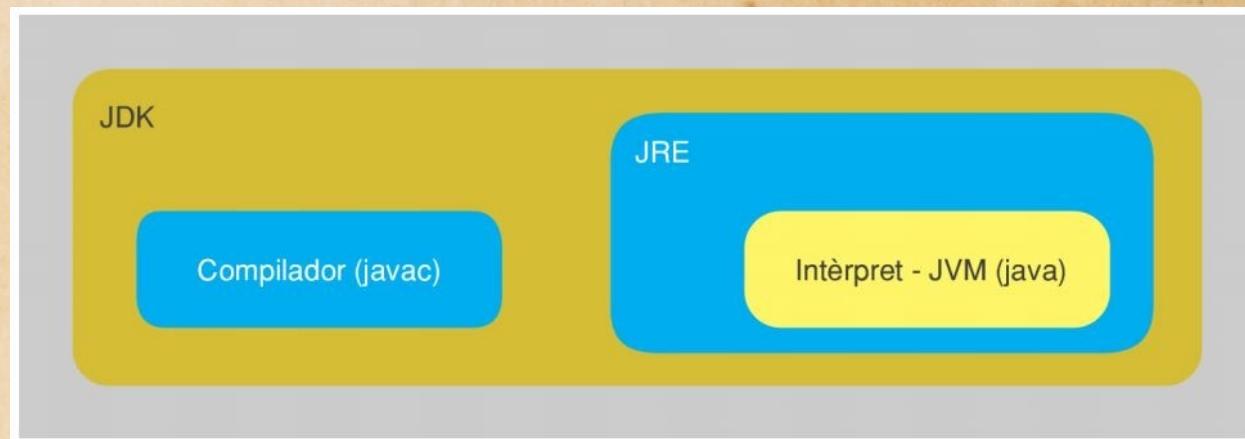
Creació i execució de programes

- Ferramentes necessàries:
 - Editor de text (Notepad, xed...)
 - Compilador de llenguatge
 - Intèrpret de Java
- El fitxer creat ha de tindre extensió `.java`
- El nom del fitxer segueix una notació "camell" en concret `UpperCamelCase` (ex: `Prova.java`, `HolaMon.java`)



Compilador i intèrpret

- Baixada gratuïta del JDK
(Baixa la última versió LTS)
- JDK conté tant el compilador com el intèrpret



Compilació i execució

- El programa que compila és "javac"
 - Exemple, en línia de comandaments:
 - javac HolaMon.java → genera un fitxer anomenat HolaMon.class (conté el *bytecode*)
- El programa que executa el bytecode generat és "java"
 - Exemple, en línia de comandaments:
 - java HolaMon

Fixa't que la instrucció anterior no inclou l'extensió ".class"

Vídeo amb exemple d'instal·lació del JDK i prova de compilació:
<https://www.youtube.com/watch?v=SZ0dfjCQ6dE>

Multiplataforma

- Si solament es necessites executar un programa Java, només caldrà tindre instal·lada la JVM (això es troba a la JRE)

```
[sh-3.2# ls Suma.class
Suma.class
[sh-3.2# java Suma
Introdueix el primer valor: 3
Introdueix el segon valor: 9
La suma es: 12
```

Això vol dir que si agafes un fitxer “.class” prèviament generat i el descarregues en un altre ordinador que només tinga el JRE (habitualment vindrà instal·lat) funcionarà correctament el programa.

Estructura HolaMon

delimitació del codi font Java

HolaMon.java

```
//El programa "Hola, mon!" en Java
public class HolaMon{
    public static void main(String[] args){
        System.out.print("Hola, món!");
    }
}
```

bloc de codi:
mètode principal

comentari del codi

primera instrucció

Pràctica 1

- *Copia el codi font del programa HolaMon.java i executa'l mitjançant línia d'ordres. El resultat per pantalla hauria ser la frase "Hola Món!"*

Respecta les lletres majúscules i minúscules. Java és sensible elles.

RECORDA: És important que realitzes aquestes pràctiques, faces proves i anotes qualsevol problema per a preguntar al professor.

Pràctica 2

- Modifica el codi font perquè el programa escriga ara "Adeu Món!"