1. Introducció i identificació dels elements d'un programa

2025-2026

Continguts

- ✓ Estructura i blocs fonamentals.
- ✓ Solucions i projectes.
- ✓ Utilització dels entorns integrats de desenvolupament.
- ✓ Variables. Tipus de dades. Literals. Constants.
- ✓ Operadors i expressions.
- ✓ Conversions de tipus.
- ✓ Comentaris.

Què és un programa?

“És la seqüència d’instruccions per un ordinador per obtenir una resposta determinada”

Tasca d’un programador:

“Escollir quines ordres constituiran un programa d’ordinador, en quin ordre s’han de dur a terme i sobre quines dades cal aplicar-les, perquè el programa porti a terme la tasca que ha de resoldre”

Què és un llenguatge de programació?

"Es tracta d'un llenguatge artificial dissenyat expressament per crear algorismes que puguin ser duts a terme per l'ordinador."

Tenen una sintaxi molt definida, que cal seguir perquè l'ordinador interpreti correctament cada ordre que se li dóna.

En un llenguatge de programació determinat, el seguit d'ordres concretes que es demana a l'ordinador que faci s'anomena conjunt d'instruccions.

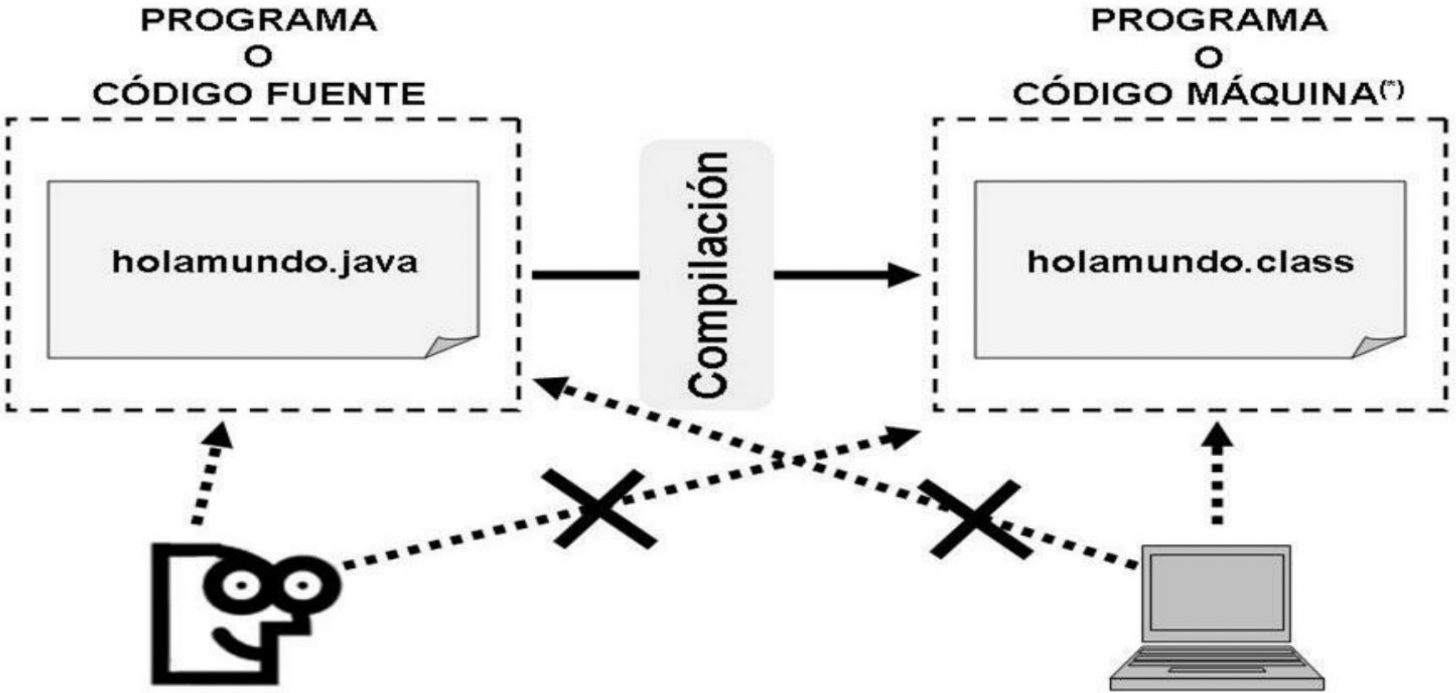
El conjunt de d'instruccions que s'utilitza a un programa es diu llenguatge de programació

Classificació dels llenguatges de programació

Hi ha dues formes de classificació:

- Segons si es tracta d'un llenguatge compilat o interpretat.
  - Aquesta propietat afecta les passes que cal seguir per arribar a obtenir un fitxer executable. O sigui, un fitxer amb el mateix format que el de les aplicacions que podeu tenir instal·lades al vostre ordinador.
- Segons si es tracta d'un llenguatge de nivell alt o baix.
  - Aquesta propietat indica el grau d'abstracció del programa i si les seves instruccions estan més o menys estretament vinculades al funcionament del maquinari d'un ordinador. Primera, segona, tercera o quarta generació. Quant més alta és la generació més s'acosta al llenguatge humà.

Interpretació del llenguatge



Llenguatge Màquina

“És el llenguatge que triaríem si volguéssim fer un programa que treballés directament sobre el processador.”

Cadascuna de les instruccions es representa amb una seqüència binària, en zeros (0) i uns (1).

Un programa escrit en llenguatge de màquina és específic per a un tipus de processador concret. No es pot executar sobre cap altre processador, tret que siguin compatibles. Un processador concret només entén directament el llenguatge de màquina especificat pel seu fabricant.

Llenguatge Màquina

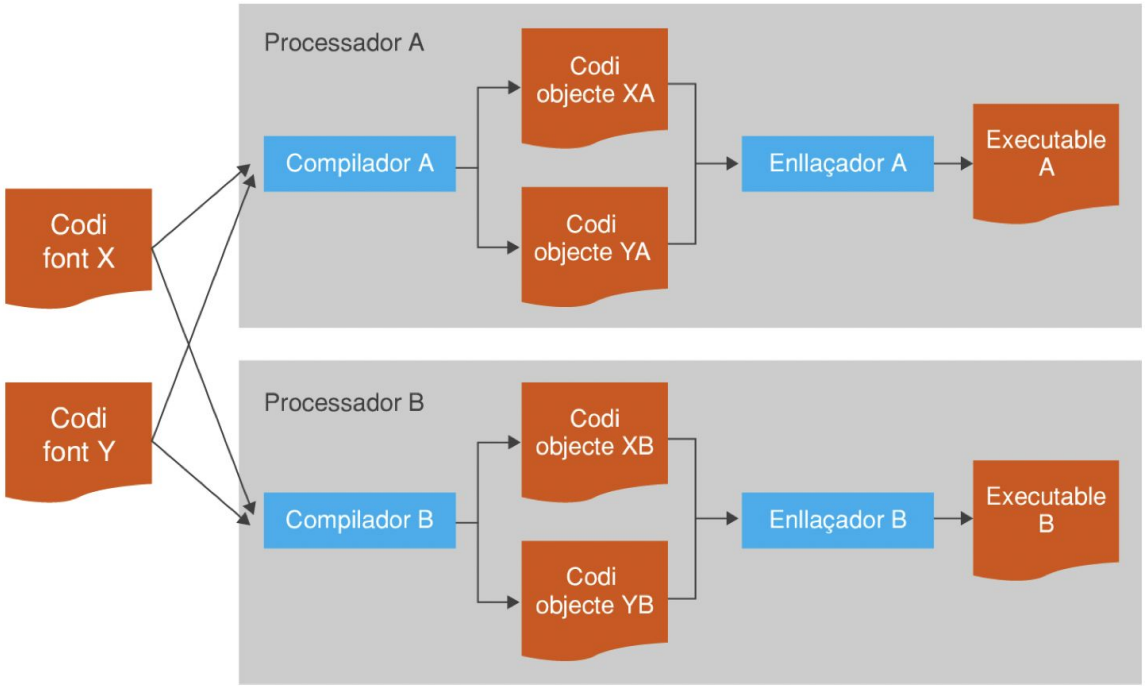
El que s'usa és un sistema auxiliar de mnemotècnics en el qual s'assigna a cada instrucció en binari un identificador en format de text llegible, més fàcilment comprensible per als humans.

Aquest recull de mnemotècnics és el que es coneix com el **llenguatge assemblador**.

Compilació

- És la traducció del codi font dels fitxers del programa en fitxers en format binari que contenen les instruccions en un format que el processador pot entendre.
- El contingut d'aquests fitxers s'anomena codi objecte.
- El programa que fa aquest procés s'anomenacompilador.
- Després del procés de compilació hi ha un pas addicional anomenat enllaçament, en el qual aquests dos codis es combinen per generar un únic fitxer executable.
- Quan el compilador detecta que una part del codi font no segueix les normes del llenguatge, el procés de compilació s'interromp i anuncia que hi ha un error de compilació.

Compilació



Procés de generació del fitxer executable usant un llenguatge compilat.

Components d'un llenguatge de programació

Tres components bàsics:

- Semàntica

  Contempla el **significat** de les construccions

- Sintaxi

  Permet la construcció de **notacions vàlides** emprant les notacions disponibles del llenguatge

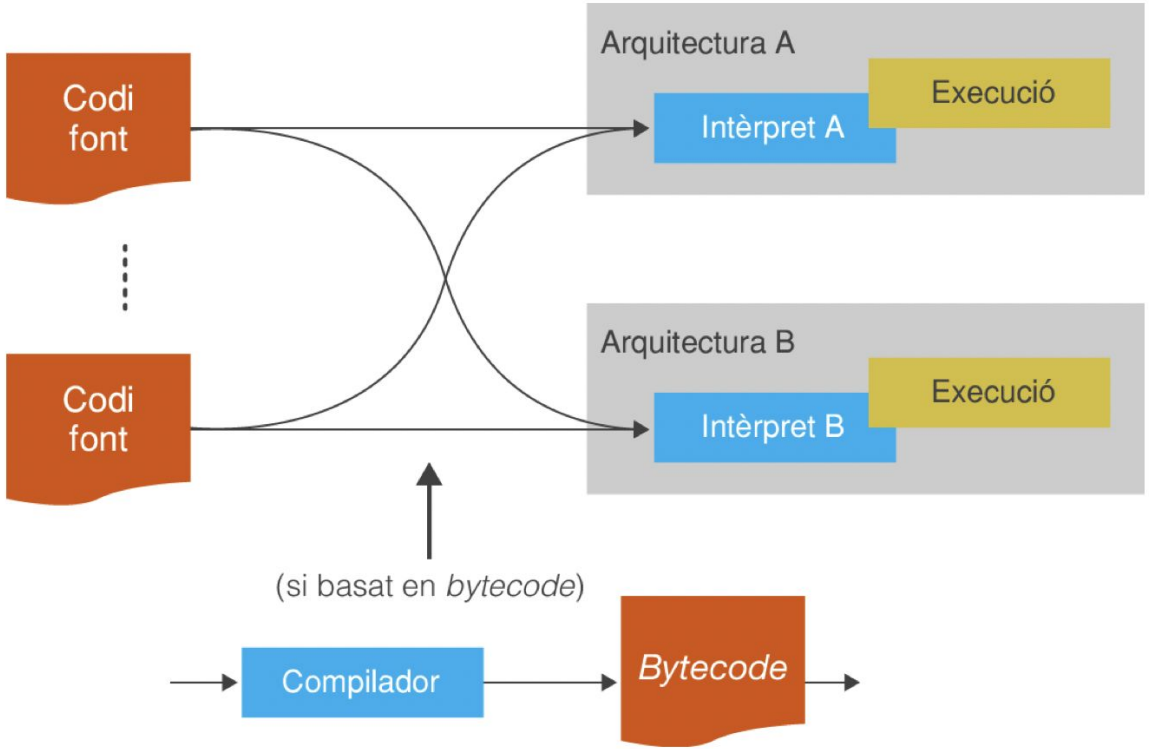
- Pragmàtica

  Tracta **l'ús** del llenguatge de programació a la pràctica (eficiència, portabilitat...)

Llenguatge interpretat

- La immensa majoria de llenguatges interpretats són de **nivell alt**.
- Ni es genera cap codi objecte ni cap fitxer executable.
- **Es treballa directament amb el fitxer de codi font**.
- Un llenguatge interpretat s'executa indirectament, mitjançant l'ajut d'un programa auxiliar anomenat **intèrpret**, que processa el codi font i en gestiona l'execució.
- Alguns llenguatges interpretats usen una aproximació híbrida. El codi font es compila i com a resultat es genera un fitxer de dades binàries anomenades bytecode.
- Aquest bytecode, però, no és formalment codi objecte, ja que no és capaç d'entendre'l el maquinari de cap processador. Només un intèrpret el pot processar i executar. Simplement és una manera d'emmagatzemar més eficient i en menys espai, en format binari i no en text, les instruccions incloses al codi font.

Llenguatge interpretat



Entorns integrats de programació

Per programar necessitem:

- Compilador o integrador + Editor (Haureu d'anar alternant execucions entre els dos)

o

- IDE: **Integrated Development Environment** o entorn integrat de desenvolupament que és una eina que integra tot el que cal per generar programes d'ordinador, de manera que la feina sigui molt més còmoda.

Característiques d'un IDE

- Possibilitat de fer ressaltar amb **codis de colors** els diferents tipus d'instruccions o aspectes rellevants de la sintaxi del llenguatge suportat, per facilitar la comprensió del codi font.
- Accés a documentació i ajuda contextual sobre les instruccions i sintaxi dels llenguatges suportats.
- **Detecció**, i en alguns casos fins i tot correcció, **automàtica d'errors de sintaxi** en el codi, de manera similar a un processador de text. Així, no cal compilar per saber que el programa està malament.
- Suport simultani del desenvolupament de **llenguatges de programació diferents**.
- Un **depurador**, una eina molt útil que permet pausar l'execució del programa en qualsevol moment o fer-la instrucció per instrucció, de manera que permet analitzar com funciona el programa i detectar errades.
- En els més avançats, sistemes d'ajut per a la **creació d'interfícies gràfiques**.

Llenguatge orientat a objectes

"Qualsevol llenguatge que permeti la definició de tipus de dades, d'operacions noves sobre aquests tipus de dades i instanciar el tipus de dades podria ser considerat orientat a objectes"

L'objecte pot contenir dades o algoritmes

Les dades defineixen l'estat de l'objecte i l'algoritme el seu comportament.

Les classes és un conjunt de variables i mètodes d'instància que contenen la definició d'un objecte.

Exemples de llenguatges orientats a objectes: C++, **Java**, C#...

Tipatge d'un Llenguatge de Programació

Tipatge Estàtic

- **Definició**: Els tipus de dades es determinen en temps de compilació.
- **Característiques**: Permet detectar errors de tipus abans de l'execució del programa, la qual cosa facilita la depuració i millora la seguretat.
- **Exemples**: Llenguatges com Java i C++.

Tipatge Dinàmic

- **Definició**: Els tipus de dades es determinen en temps d'execució.
- **Característiques**: Ofereix més flexibilitat en la manipulació de dades, però pot resultar en errors que només apareixen durant l'execució.
- **Exemples**: Llenguatges com Python i JavaScript.

Tipatge Fort

- **Definició**: No permet operacions entre tipus incompatibles sense conversió explícita.
- **Característiques**: Redueix la probabilitat d'errors relacionats amb tipus de dades, ja que exigeix que els tipus siguin compatibles.
- **Exemples**: Llenguatges com Java i Ruby.

Tipatge Feble

- **Definició**: Permet conversió automàtica entre diferents tipus de dades.
- **Característiques**: Facilita operacions ràpides i pot ser més permissiu, però pot portar a comportaments inesperats i errors difícils de detectar.
- **Exemples**: Llenguatges com JavaScript i PHP.

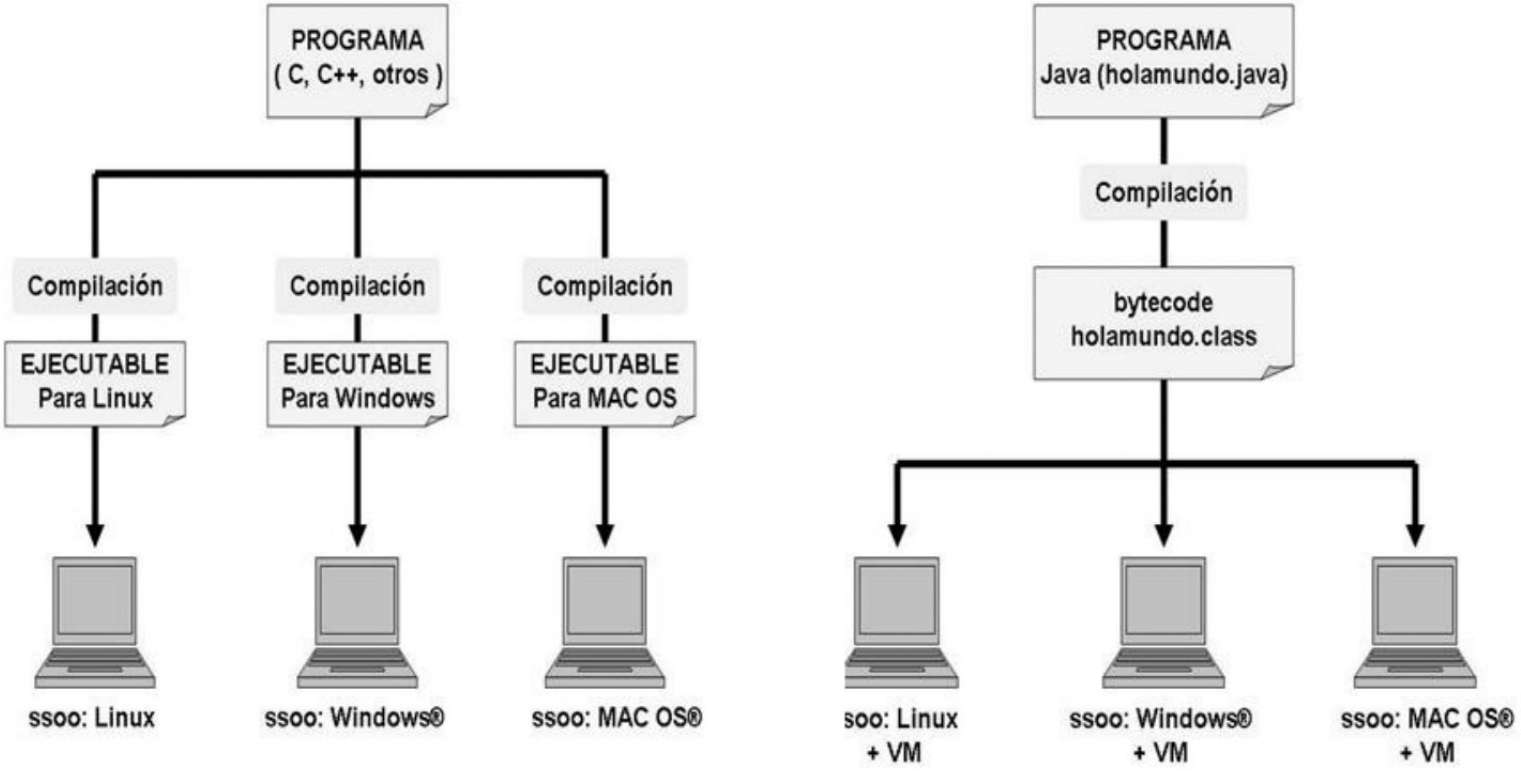
Tipatge dels Llenguatges de Programació

Llenguatge	Facilitat d'aprenentatge	Àmbit d'aplicació principal	Tipatge	Compilat/Interpretat	Usos típics	Nivell de programació
1. Python	Fàcil	Desenvolupament general	Dinàmic	Interpretat	Ciència de dades, IA, web	Alt nivell
2. C	Difícil	Sistemes i maquinari	Fortament tipat	Compilat	Sistemes operatius, embeguts	Baix nivell
3. C++	Moderat	Desenvolupament de jocs, sistemes	Fortament tipat	Compilat	Jocs, motors gràfics, sistemes	Baix-mitjà nivell
4. Java	Moderat	Aplicacions empresarials, Android	Fortament tipat	Compilat	Empresarial, Android	Alt nivell
5. C#	Moderat	Desenvolupament d'aplicacions, jocs	Fortament tipat	Compilat	Aplicacions .NET, Unity	Alt nivell
6. JavaScript	Fàcil	Desenvolupament web	Dinàmic	Interpretat	Web (front-end, back-end)	Alt nivell
7. Visual Basic	Moderat	Aplicacions empresarials, automatització	Dinàmic	Compilat	Aplicacions empresarials	Alt nivell
8. PHP	Fàcil	Desenvolupament web	Dinàmic	Interpretat	Web, gestió de contingut	Alt nivell
9. SQL	Fàcil	Bases de dades relacionals	Declaratiu	Interpretat	Gestió de bases de dades	Alt nivell
10. Go	Moderat	Sistemes distribuïts, núvol	Fortament tipat	Compilat	Serveis web, aplicacions en núvol	Alt nivell
11. Swift	Moderat	Desenvolupament per a iOS	Fortament tipat	Compilat	Aplicacions mòbils	Alt nivell
12. Ruby	Fàcil	Desenvolupament web	Dinàmic	Interpretat	Web (Ruby on Rails)	Alt nivell
13. Assembly	Difícil	Maquinari, microcontroladors	No tipat	Compilat	Desenvolupament de maquinari	Baix nivell
14. MATLAB	Fàcil	Càlcul numèric i simulacions	No tipat	Interpretat	Simulació, càlcul científic	Alt nivell
15. Fortran	Moderat	Càlcul científic, simulacions	Fortament tipat	Compilat	Càlcul numèric, ciències	Alt nivell
16. Rust	Moderadament difícil	Desenvolupament de sistemes	Fortament tipat	Compilat	Sistemes, aplicacions segures	Baix-mitjà nivell
17. Kotlin	Moderat	Aplicacions Android	Fortament tipat	Compilat	Android, back-end	Alt nivell
18. R	Moderat	Anàlisi de dades, estadística	Dinàmic	Interpretat	Ciència de dades, estadística	Alt nivell
19. Dart	Fàcil	Aplicacions mòbils, UI	Fortament tipat	Compilat	Web, mòbil (Flutter)	Alt nivell
20. Perl	Moderat	Scripts, automatització	Dinàmic	Interpretat	Administració de sistemes, web	Alt nivell
21. Ada	Difícil	Sistemes crítics, temps real	Fortament tipat	Compilat	Sistemes aerospacials, militar	Alt nivell

Introducció a Java

- Java va ser creat l'any 1995 per la firma Sun Microsystems, que el 2009 va ser comprada per l'empresa de bases de dades Oracle.
- De nivell alt amb **compilador estricte**
- **Multiplataforma** (els seus programes es poden executar en qualsevol plataforma sense que calgui tornar a compilar)
- **Orientat a objectes**
- **Interpretat amb bytecode** (En tractar-se d'un llenguatge interpretat, cal disposar de l'intèrpret correctament instal·lat i configurat a cada màquina on vulgueu executar el vostre programa)
- És un llenguatge de **propòsit general**

Comparació entre Java i altres programes



A tenir en compte

L’extensió d’un programa JAVA és **.java**

La nomenclatura de fitxers es farà en notació de **camell** (UpperCamelCase): HolaMon.java

No emprar a la nomenclatura per evitar errors de compilació:

- Accents
- Espais en blanc
- Caràcters especials

Instal·lacions necessàries:

- Compilador
- Editor de text
- Intèrpret de Java

Tipus d'errors a Java

Existeixen 3 tipus:

- Errors en temps de compilació o de sintaxi: És detectat pel compilador
- Errors en temps d'execució: Es produeixen quan s'executa el programa pel que el compilador no els reconeix. Per evitar-los podem fer ús de les excepcions.
- Errors lògics: Provocats per la introducció incorrecta del programador dels requeriments funcionals.

JDK

- Java Development Kit: conté el JRE (Java Runtime Enviroment) que és l'entorn d'execució i la JVM (Java Virtual Machine) que és l'intèrpret de Java.
- Proporciona un seguit d'executables via línia d'ordres que serveixen per fer diferents tasques amb codi font Java.
- El programa que posa en marxa el compilador és l'executable anomenat javac (en un sistema Windows, javac.exe)
- No conté camp eina gràfica
- Conté aplicacions de consola, eines de compilació, documentació i depuració.
- El JRE proporciona les biblioteques de classes i altres recursos per executar Java.
- Per al fitxer amb codi font Java anomenat HolaMon.java caldria obrir una línia d'instruccions i executar:

```
javac HolaMon.java
```

El fitxer amb *bytecode* resultant del procés de compilació s'anomena igual que el fitxer de codi font, però amb l'extensió .class.

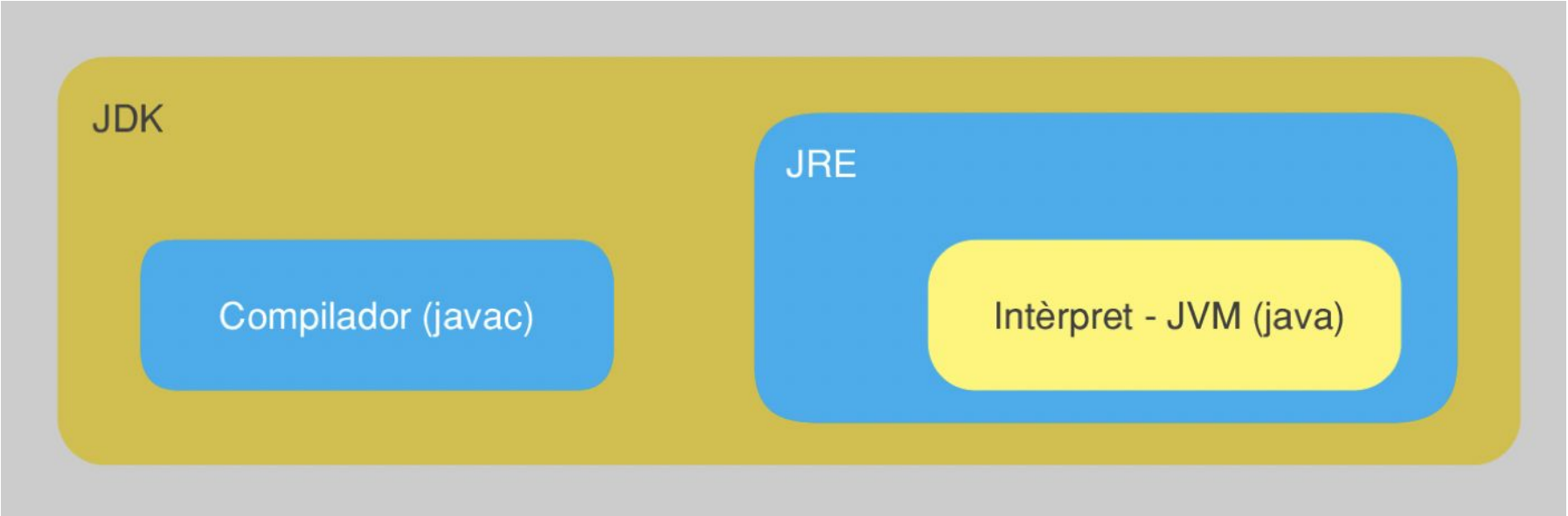
JDK

- L’executable que posa en marxa l’intèrpret de Java és l’anomenat java (en un sistema Windows, java.exe).
- Un cop es disposa del fitxer de *bytecode* HolaMon.class es pot executar des de la línia d’ordres fent:

```
java HolaMon
```

- Noteu que no s’especifica cap extensió.
- Ell sol dedueix que l’extensió ha de ser .class.
- Immediatament, el programa es posarà en marxa.

Estructura del JDK



IDE

IDE: Entorn bàsic de desenvolupament de JAVA a un únic entorn.

Alguns IDE’s:

- Netbeans
- Eclipse
- JCreator
- IntelliJ IDEA

Booleà

El tipus de dada **booleà** representa un valor de tipus lògic per tal d'establir la certesa o falsedat d'un estat o afirmació.

A Java s'identifica amb "boolean"

Valors que pot tenir:

- True
- False

Exemples:
Estar casat, encés o apagat, contrasenya correcta...

Enter i real

El tipus de dada **enter** representa un valor numèric, positiu o negatiu, sense cap decimal.

A java s'identifica amb "**int**"

Exemples:
Edat, dia del mes, nombre de fills…

El tipus de dada **real** representa un valor numèric, positiu o negatiu, amb decimals.

A Java s'identifica amb "**double**"

Exemples:
Distància, preu...

Caràcter

El tipus de dada **caràcter** representa una unitat fonamental de text usada en qualsevol alfabet, un nombre o un signe de puntuació o exclamació.

A Java s'identifica amb "**char**"

Exemples:
A, T, >, ?

Ús de sistemes de representació de caràcters:

- UNICODE: Representa 65.563 caràcters
- ASCII: no permet segons quins caràcters per idiomes de signes o ciríl·lic ja que només permet 128 ó 256 depèn la versió.

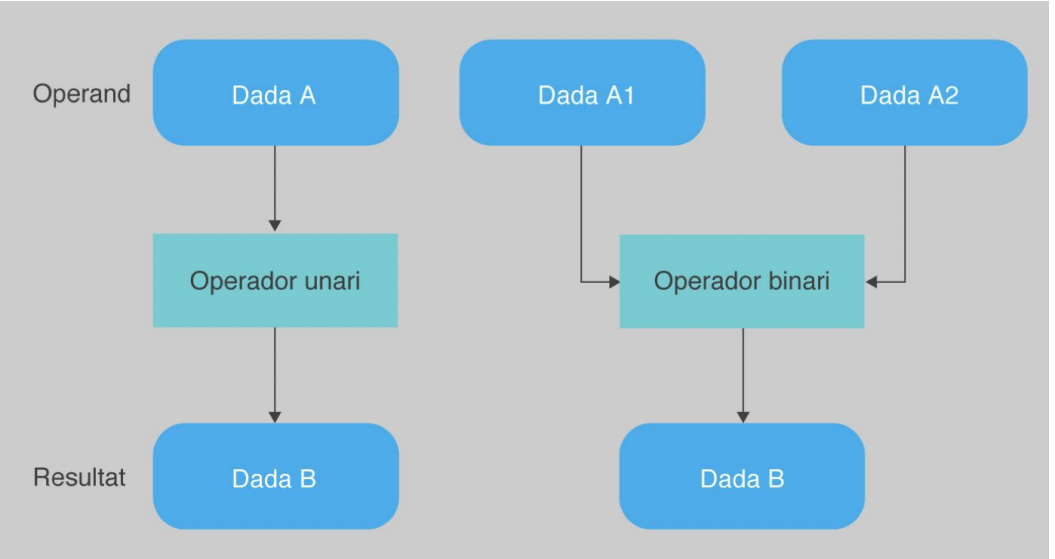
Transformació de dades

Les dades són transformades mitjançant el que anomenam **operacions**.

Les operacions es divideixen en dos tipus: unàries o binàries, segons el nombre d'operands que usen (un o dos).

S'han de respectar les operacions amb els tipus de dades

El símbol com s'identifica cadascuna de les operacions dins d'un tipus de dada és el seu operador. Les dades sobre les quals s'aplica una operació són els seus operands.



Operacions entre booleans

&& Conjunció
|| Disjunció
! Negació

Operands		Resultats de l'operació		
A	B	A && B	A || B	! A
false	false	false	false	true
true	false	false	true	false
false	true	false	true	true
true	true	true	true	false

Operacions entre booleans

== Igual
!= Diferent

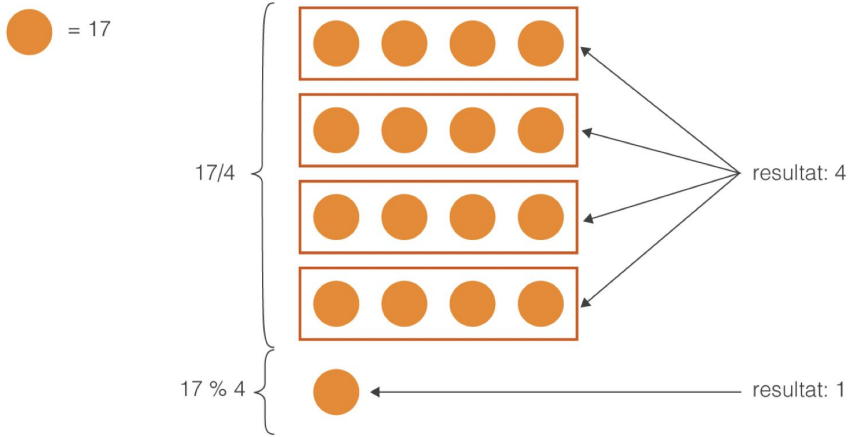
Operands		Resultats de l'operació	
A	B	A == B	A != B
false	false	true	false
true	false	false	true
false	true	false	true
true	true	true	false

Operacions entre enters

Les operacions bàsiques que es poden fer sobre dades d'aquest tipus són de dos tipus: aritmètiques i relacionals.

Els operadors aritmètics generalment suportats són el canvi de signe (-, operador unari), la suma (+), resta (-, operador binari), multiplicació (*) i divisió (/).

En alguns llenguatges de programació també hi ha una cinquena operació anomenada mòdul (%). El resultat d'aplicar-la sobre dos enters retorna la resta de l'operació de divisió



Operadors relacionals

Operands		Resultat de les operacions				
A	B	A == B	A > B	A < B	A >= B	A ≤ B
4	3	false	true	false	true	false
14	-2	false	true	false	true	false
-78	34	false	false	true	false	true
12	12	true	false	false	true	true

Operadors Unitaris

Operador	Uso	Operación
~	~A	Complemento a 1 de A
-	-A	Cambio de signo del operando
--	A--	Decremento de A
++	A++	Incremento de A
!	! A	Not A (ya visto)

Operadors d'assignació

Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación. La operación A+=B equivale a A=A+B.
-=	A -= B	Resta y asignación. La operación A-=B equivale a A=A-B.

Construcció d'expressions

Una **expressió** és una combinació qualsevol d'operadors i operands.

Ha de ser correcta en dos nivells, sintàcticament i semànticament.

Les expressions sempre s'escriuen en una sola línia.

Ordenant els elements d'esquerra a dreta i de dalt a baix.

Normes bàsiques d'expressions

- Sintàcticament
  - Es considera que un literal sol és en si mateix una expressió.
    - 6, per a la primera regla.
  - Donada una expressió correcta E, també ho és escriure-la entre parèntesis: (E).
    - (6 + 5), per a la segona regla, en què E és 6 + 5.
  - Donada una expressió correcta E i un operador unari qualsevol *op*, *op*E és una expressió correcta.
    - -4,per a la tercera regla, en què *op* és -i E és 4.
  - Donades dues expressions correctes E1 i E2 i un operador binari qualsevol *op*, E1 *op* E2 és una expressió correcta.
    - (6 + 5) * -4, per a la quarta regla, en què E1 és (6 + 5), E2 és -4 i *op* és *.
- Semànticament
  - Qualsevol operació sempre ha de ser entre dades del mateix tipus.
  - L'operació usada ha d'existir per al tipus de dada.

Avaluació d'expressions

- Entenem com a *avaluar* una expressió anar aplicant tots els seus operadors sobre les diferents dades que la conformen fins arribar a un resultat final, que és la dada resultant.
- L'avaluació sempre es comença calculant les expressions que es troben entre parèntesis, en ordre de més intern a més extern.
- Un cop han desaparegut tots els parèntesis, llavors s'apliquen les operacions segons el seu ordre de precedència.
- L'**ordre de precedència** d'un conjunt d'operadors és la regla usada per establir de manera no ambigua l'ordre com s'han de resoldre les operacions dins d'una expressió.

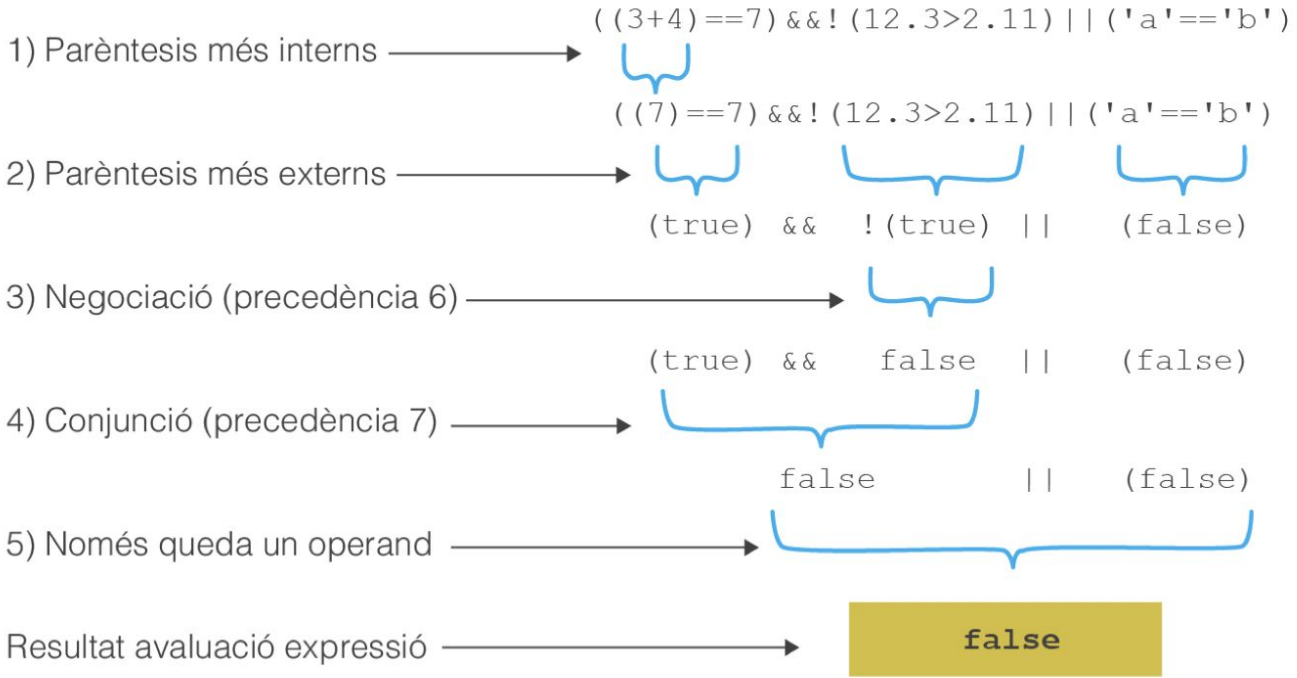
Ordre de preferència

Ordre	Operació	Operador
1	Canvi de signe	-(unari)
2	Producte, divisió i mòdul	* / %
3	Suma i resta	+ -
4	Relacionals de comparació	> < <= >=
5	Relacionals d'igualtat	== !=
6	Negació	! (unari)
7	Conjunció	&&
8	Disjunció	||

Exercici ordre preferència

```
1  //Comprova una expressió complexa
2  public class ProvaExpressio {
3    public static void main(String[] args) {
4        System.out.println(((3 + 4) == 7 )&&!(12.3 > 2.11)||('a' == 'b'));
5    }
6  }
```

Resultat exercici ordre preferència



Desbordaments i errors de precisió

Els llenguatges de programació només poden representar un rang concret. Si es supera el rang es diu que té “overflow” o desbordament.

Tampoc no és possible representar qualsevol seqüència de decimals amb un nombre real.

La conseqüència directa d’aquest fet és que en fer càlculs amb dades reals poden aparèixer errors de precisió.

Tipus de dades

Tipo de datos	Información representada	Rango
byte	Datos enteros	-128 ←→ +127
short	Datos enteros	-32768 ←→ +32767
int	Datos enteros	-2147483648 ←→ +2147483647
long	Datos enteros	-9223372036854775808 ←→ +9223372036854775807
char	Datos enteros y caracteres	0 ←→ 65535
float	Datos en coma flotante de 32 bits	Precisión aproximada de 7 dígitos
double	Datos en coma flotante de 64 bits	Precisión aproximada de 16 dígitos
boolean	Valores booleanos	true/false

Com es fan emprar?

Tipo de dato	código
byte	byte a;
short	short b, c=3;
int	int d = -30; int e = 0xC125;
long	long b=434123 ; long b=5L ; /* la L en este caso indica Long*/
char	char car1='c'; char car2=99; /*car1 y car2 son lo mismo porque el 99 en decimal es la 'c' */
float	float pi=3.1416; float pi=3.1416F; /* la F en este caso indica Float*/ float medio=1/2F; /*0.5*/
double	double millón=1e6; /* $1x10^6$ */ double medio1/2D; /*0.5 la D en este caso indica Double*/
boolean	boolean adivinanza=true;

Com es fan emprar?

Existeixen 4 sistemes alfanumèrics: Decimal, Hexadecimal, Octal i binari

	Decimal	Hexadecimal	Octal	Binari
Caracters	0,1,2,3,4,5,6,7,8,9	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	0,1,2,3,4,5,6,7	0,1
Identificació a Java sempre a l'inici		0X	0	0B

A més es pot emprar . per indicar decimals i _ per remarcar dígits i ha d'anar amb número abans i després.

Ús de la memòria

Una **variable** és una dada emmagatzemada a la memòria que pot veure modificat el seu valor en qualsevol moment durant l'execució del programa.

- **La memòria és un espai compartit** entre tots els programes que es troben en execució, com el sistema operatiu.
- **Les dades emmagatzemades no són persistents**, només existeixen mentre el programa està resolent la tasca en curs.
- Recordeu que **les dades es representen en format binari** quan es troben dins de l'ordinador.
- **Les dades s'organitzen en cel·les** dins de la memòria, d'una manera més aviat semblant a un full de càlcul. Cadascuna pot contenir 8 bits d'informació. Ara bé, una dada pot ocupar més d'una cel·la i el nombre exacte que n'ocupa depèn del seu tipus i del llenguatge usat.

Declaració de variables

Tota variable dins del codi font d'un programa ha d'haver estat **declarada** prèviament pel programador abans de poder fer-la servir.

Per declarar una variable dins el vostre programa, només cal especificar tres coses:

- el tipus
- un identificador o etiqueta únic
- un valor inicial.

No tots els llenguatges requereixen assignar un valor inicial a una variable. Per aquest cas és convenient inicialitzar la variable sempre.

Sempre a JAVA s'han de declarar a una classe

Exemple declaració variable

```java
//El programa que només declara una variable de tipus enter
public class DeclararEnter {
  public static void main(String[] args) {
    int elMeuEnter = 2;
    //Ara hi ha una variable en memòria que conté el valor enter 2.
  }
}
```

Identificadors de variables

És un text arbitrari per identificar una variable

```java
//Programa A. Un codi poc entenedor.
public class DivideixISuma {
  public static void main(String[] args) {
    double x = 20.0;
    double y = 6.0;
    double z = 3.0;
    //Ara vindria la resta del codi
    ...
  }
}
```

```java
//Programa B. Un codi més entenedor.
public class DivideixISuma {
  public static void main(String[] args) {
    double dividend = 20.0;
    double divisor = 6.0;
    double sumarAlFinal = 3.0;
    //Ara vindria la resta del codi
    ...
  }
}
```

Característiques identificadors

- No pot contenir espais.
- No pot començar amb un nombre.
- Es desaconsella usar accents.
- No pot ser igual que alguna de les paraules clau del llenguatge.

Paraules Clau o reservades

El conjunt de paraules reservades en Java s'enumera tot seguit: abstract, long, class....

Hi ha una convenció de codi per anomenar-los en Java. En aquest cas cal usar lowerCamelCase (notació de camell minúscula)

Recordeu que el Java és sensible a majúscules i minúscules, per la qual cosa elMeuEnter i elmeuEnter es consideren identificadors totalment diferents.

Una **paraula clau (o reservada)** d'un llenguatge de programació és aquella que té un significat especial dins la seva sintaxi i s'usa per compondre certes parts o instruccions en el codi font d'un programa.

Àmbit de les variables

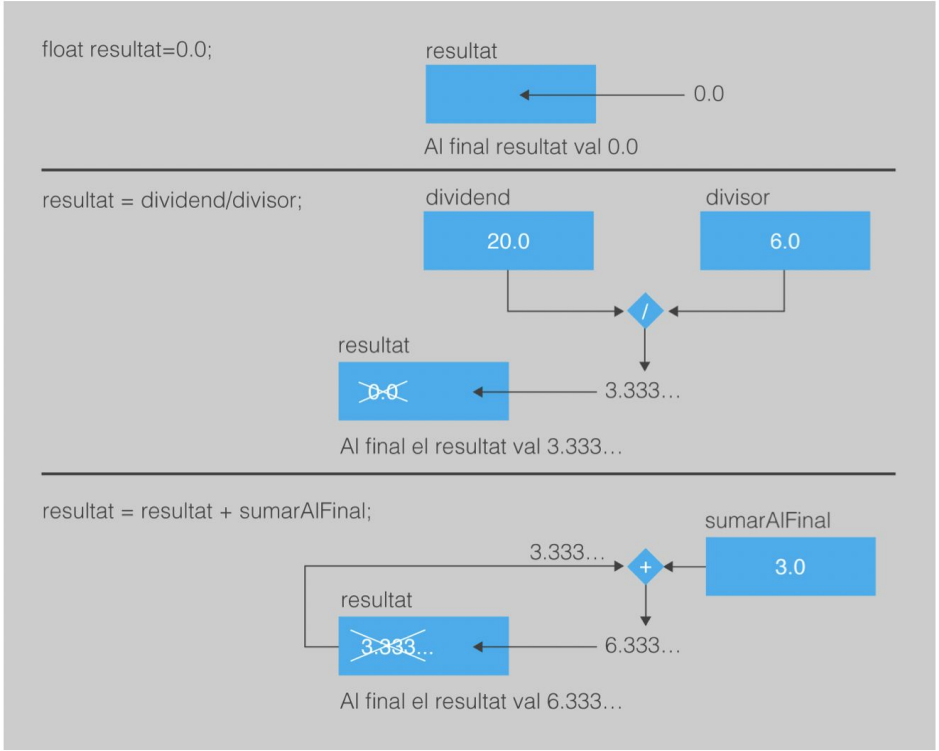
- Quan una variable ha estat declarada, el seu identificador es pot usar exactament igual que un literal dins de qualsevol expressió. El valor que representarà serà el del valor emmagatzemat en memòria en aquell instant per a aquella variable.
- la declaració d'una variable només té validesa des de la línia de codi on es declara fins a trobar el delimitador que marca el final del bloc de codi on s'ha declarat
- Fora d'aquest rang d'instruccions del codi font, el seu àmbit, és com si no estigués declarada.
- i s'intenta fer ús d'un identificador que no correspon a cap variable declarada (ja sigui perquè mai no s'ha declarat o perquè se'n fa ús fora del seu àmbit), en compilar el codi font es produirà un error.

L'àmbit d'una variable és el context sota el qual es considera declarada.

Exercici comportament d'una variable

```java
//Un programa que calcula una divisió i una suma, però a poc a poc
public class DivideixISumaDetallat {
  public static void main(String[] args) {
    double dividend = 20.0;
    double divisor = 6.0;
    double sumarAlFinal = 3.0;
    double resultat = 0.0;
    //El resultat serà una variable que s'anirà modificant.
    resultat = dividend/divisor;
    //Una expressió que usa la mateixa variable que es modifica!
    resultat = resultat + sumarAlFinal;
    System.out.println(resultat);
  }
}
```

Resultat exercici comportament d'una variable



Constants

Una **constant** és un tipus especial de variable que té la particularitat que dins del codi del programa el seu valor només pot ser llegit, però mai modificat.

La sintaxi és idèntica a la definició de la variable, però abans de la paraula clau per al tipus de dada cal afegir les paraules reservades **private static final** , en aquest ordre.

Aquesta expressió no pot contenir variables encara que el seu valor inicial por ser un literal o una expressió.

Les constants les posarem en majúscules mentre que les variables poden anar en minúscules.

Exemple Constant

```java
//Calcula el preu de comprar diverses coses
public class CalculaPreu {
  public static void main(String[] args) {
    double preuFinal = 0.0;
    //Va sumant preus de productes
    ...
    preuFinal = preuFinal + (preuFinal * 0.18);
    System.out.println(preuFinal);
  }
}
```

```java
//Calcula el preu de comprar diverses coses, i queda més clar com ho fa
public class CalculaPreu {
  private static final double IVA = 0.18;
  public static void main(String[] args) {
    double preuFinal = 0.0;
    //Va sumant preus de productes
    ...
    preuFinal = preuFinal + (preuFinal * IVA);
    System.out.println(preuFinal);
  }
}
```

Conversions de tipus

Una **conversió de tipus** és la transformació d'un tipus de dada en un altre de diferent

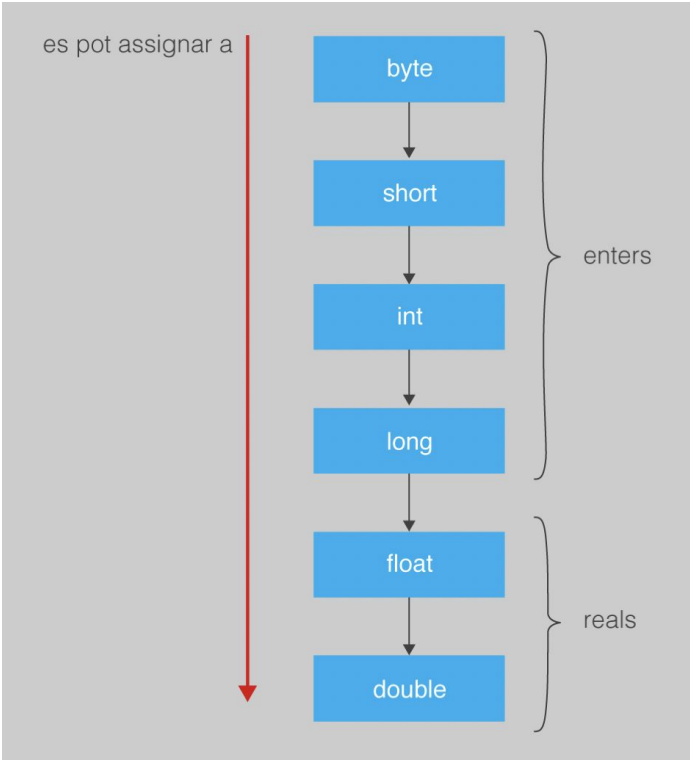
Hi ha dos tipus de conversions:

- Implícita: aquelles conversions fàcils de resoldre i que el llenguatge de programació és capaç de gestionar automàticament sense problemes
- Explícita: més complexes i el força el programador, si és possible amb l'operació "cast"

Conversió implícita

```
1  public class ConversioTipus {
2    public static void main (String[] args) {
3      //El literal "100" és un enter i "real" és un float. Són tipus diferents,
           no?
4      float real = 100;
5      //Però, "doble" i "real" són dos variables de tipus diferent, no?
6      double doble = real;
7      System.out.println(doble);
8    }
9  }
```

Relacions de compatibilitat entre tipus numèrics



Conversió explícita

Sense la conversió de sencer a int no compilaria

```
1  public class ConversioExplicita {
2    public static void main (String[] args) {
3      double realLlarg = 300.0;
4      //Assignació incorrecta. Un real té decimals, no?
5      long enterLlarg = (long)realLlarg;
6      //Assignació incorrecta. Un enter llarg té un rang major que un enter, no?
7      int enter = (int)enterLlarg;
8      System.out.println(enter);
9    }
10 }
```

Comentaris

Tipus:

- D'una línea: //
- Vàries línies: /* aquí el texte que volguem */
- Comentaris de la documentació: /* el texte indicant cada línia amb "*" i finalitzam */

```
/**Comienza el comentario
*
*las etiquetas se utilizan para especificar un parámetro
*o método o título
*Las etiquetas HTML también se pueden usar
*como <h1>
*
*el comentario termina*/
```

https://www.baeldung.com/javadoc

Etiquetes disponibles per comentaris

ETIQUETA	DESCRIPCIÓN	SINTAXIS
@author	Agrega el autor de una clase.	@author nombre
{@code}	Muestra texto en la fuente del código sin interpretar el texto como marcado HTML o etiquetas javadoc anidadas.	{@code texto}
{@docRoot}	Representa la ruta relativa al directorio raíz del documento generado desde cualquier página generada.	{@docRoot}
@deprecated	Agrega un comentario que indica que esta API ya no se debe usar.	@deprecated texto-obsoleto
@exception	Agrega un subtítulo “Throws” a la documentación generada, con el nombre de clase y el texto de descripción.	@exception descripción del nombre-clase
{@inheritDoc}	Hereda un comentario de la clase heredable más cercana o la interfaz implementable.	Hereda un comentario de la superclase inmediata.
{@link}	Inserta un enlace con la etiqueta de texto visible que apunta a la documentación del paquete, clase o nombre de miembro especificado de una clase referenciada.	{@linkplain package.class#etiqueta-miembro}
{@linkplain}	Idéntico a {@link}, excepto que la etiqueta del enlace se muestra en texto plano que en la fuente del código.	{@linkplain package.class#etiqueta-miembro}
@param	Agrega un parámetro con el nombre de parámetro especificado seguido de la descripción especificada en la sección “Parámetros”.	@param descripción del nombre-parámetro
@return	Agrega una sección “Returns” con el texto de descripción.	@return descripción
@see	Agrega un encabezado “See Also” (“Ver también”) con un enlace o entrada de texto que apunta a la referencia.	@see referencia
@serial	Usado en el comentario del documento para un campo serializable predeterminado.	@serial campo-Descripción | incluir | excluir

ETIQUETA	DESCRIPCIÓN	SINTAXIS
@serialData	Documenta los datos escritos por los métodos writeObject() o writeExternal().	@serialData descripción-datos
@serialField	Documenta un componente ObjectStreamField.	@serialField campo-nombre campo-tipo campo-descripción
@since	Indica a partir de que versión de la API fue incluida la clase o método.	@since release
@throws	Las etiquetas @throws y @exception son sinónimos.	@throws descripción del nombre de la clase
{@value}	Cuando {@value} se usa en el comentario del doc de un campo estático, muestra el valor de esa constante.	{@value package.class#campo}
@version	Agrega un subtítulo de “Versión” con el texto de versión especificado a los documentos generados cuando se usa la opción -version.	@version texto-version