Sintaxi bàsica

La sintaxi d'un llenguatge defineix els elements del llenguatge i com es combinen entre si per formar un programa. Cal tenir cura amb l'ús de la sintaxi i seguir un conjunt de convencions de codi per tal de fer-ho mes llegible. El codi es divideix en diferents parts, els anirem veient en les pròximes sessions i son els següents:

- **Importació de biblioteques:** Mecanisme per afegir noves instruccions al repertori bàsic del llenguatge.
- **Identificadors:** Un identificador es un nom que identifica a una variable, un mètode o funció o a una classe, les regles per escriure un identificador son les següents:

    - **Tots els identificadors han de començar amb una lletra, caràcter de subratllat(_) o el símbol de dolar ($)**
    - **Pot incloure però no pot començar per un número.**
    - **No pot tenir espais en blanc**
    - **Es sensible a majúscules i minúscules**
    - **No es poden utilitzar les paraules reservades com a identificadors.**

    A part d'aquestes regles sintàctiques hi ha certes convencions que fan més llegible un programa però no afecten a la execució del programa. La primera es escollir un nom que de per si sigui significatiu, això facilita la lectura del programa en cas que revisem el codi al cap d'un cert temps.

TIPUS IDENTIFICADOR	CONVENCIÓ	EXEMPLE
Nom de classe	Comença amb lletra majúscula	Rectangle, Cotxe,...
Nom de funció	Comença amb lletra minúscula	calcularDistancia, setSalari,...
Nom de variable	Comença amb lletra minúscula	area, color, anys,dataNaixement
Nom de constant	En lletres majúscules	MAX_VIDES, PI,...

- **Comentaris:** Un comentari es un text addicional que afegim al codi per explicar la seva funcionalitat, be per al propi programador com per a altres que puguin llegir el codi. El compilador ignora els comentaris, per tant no augmenten la mida de l'arxiu executable. Altre ús i potser el més habitual es el de comentar una línia de codi que ens marca un error, normalment a la fase de depuració del programa.

TIPUS COMENTARI	EXEMPLE
Una sola línia	`// Comentari de una sola línia`

Comentari diverses línies	`/* Comentari de moltes línies`
	`bla, bla`
	`*/`

Comentari de documentació	`/** Comentari de documentació que pot`
	`*ocupar diferents línies`
	`*/`

- **Sentencies:** Una sentència es una ordre especifica que realitza una tasca concreta, totes les sentències acaben amb un punt i coma (;) Les sentències poden declarar una variable, cridar a una funció, etc.

Exemples:

```
int i=1;
System.out.println("El primer programa");
rect.mover(10, 20);
System.out.println("Hola Mon!!");
```

- **Blocs de codi:** Un bloc de codi es un grup de sentencies que es comporten com a una unitat. Un bloc de codi està sempre limitat per unes claus de apertura { i tancament }. Com exemples tenim la definició d'una classe, una funció, estructures de control i sentencies agrupades per claus.

EXEMPLE:

```
import java.awt.*; //IMPORTACIÓ DE API

public class HolaMon { //inici del bloc corresponent a la classe
void mover(int x,int y)
{
	//soc una funció sense sentencies
}
	public static void main(String[] args) { //INICI FUNCIO MAIN
		// Comentari de una sola línia
		/* Comentari de moltes línies
		   bla, bla
		   ble, ble
		 */
		/** Comentari de documentació que també
		 * pot ocupar
		 * diferents línies
		 */
		{ //Inici bloc de de sentencies
		 int i=1;
		 System.out.println("El primer programa");
		 rect.mover(10, 20);
		 System.out.println("Hola Mon!!");
		} //Fi bloc de sentencies
		if (i==0)
		{// Inici bloc estructura de control condicional simple
			i=i+10;
		}// Fi bloc condicional

	}// FI BLOC MAIN
} //fi de bloc de la classe
```

Hem de tenir present que les variables que declarem dins d'un bloc de codi son visibles només en aquells blocs de codi. Estudiarem aquest comportament amb pràctiques.

- **Expressions:** Una expressió es tot allò es pot posar a la dreta del operador d'assignació (=).

EXEMPLES

```
Z=148;
```

```
C=(z-20)/5;
Sou=calcula_nomina(2000);
```

- **Variables:** Una variable es un nom que te associat una adreça de memòria del ordinador, en la que es guarda el valor assignat a la variable. Hi ha diferents tipus de variables i cada tipus requereix diferents espais de memòria. Totes les variables han de ser declarades abans de fer-les servir, la declaració inclou el tipus de variable i el nom de la variable. Un cop declarada li podem assignar valors.
- **Literals:** Un literal és un text usat per representar un valor fix dins del codi font d'un programa. Hem vist alguns als exemples d'aquest nucli formatiu.
  EXEMPLES

```
public class Literals {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println(true);	//literals booleans
		System.out.println(false);
		System.out.println(0);		//literals enters
		System.out.println(-345);
		System.out.println(-9653.3333);// literals reals
		System.out.println(100.0003);
		System.out.println('a');	//literals character
		System.out.println('A');
	}

}
```

Les variables son un dels elements bàsics de qualsevol programa i s'han de:

- **DECLARAR**
- **INICIALITZAR**
- **UTILITZAR-LES**

EXEMPLES

```
int x=0;
String nom="Anna";
double a=5.5, b=0.9, c=-8.8;
boolean condueix=true;
int[] dades;
```

DEVANT DE CADA NOM DE VARIABLE HEM D'ESPECIFICAR UN TIPUS DE VARIABLE.
Java disposa de les següents variables:

- **Variables de instancia (NO ESTÀTIQUES):** Les estudiarem en profunditat a partir de la UF4, son variables que guarden l'estat d'un objecte. Tècnicament son camps no estàtic, el que vol dir que cada objecte pot tenir un valor diferent en aquesta variable.

- **Variables de classe (ESTÀTIQUES):** Son similars a les variables de instancia, amb la diferència de que els valors que es guarden son els mateixos per a tots els objectes d'una classe determinada.
- **Variables locals:** S'utilitzen dins de les funcions membre o mètodes. Una variable local existeix des de el moment de la seva declaració fins el final del bloc on es troba. Durant les primeres 3 UFS treballarem principalment amb variables locals.

EXEMPLES

```
public class UsVariables {


static final float iva=0.21f; //variable estàtica o de instància
float preu;                   //variable no estàtica o de classe

      float calcular_preu()
      {
            float preu_final;         //variable local
            preu_final=preu*iva;
            return preu_final;
      }
}
```

- **Tipus de dades primitius:**

TIPUS DE DADA	DESCRIPCIÓ	VALORS	VALOR PER DEFECTE
boolean	Pot contenir els valors true o false. Ocupa un bit.	True o False	false
char	Caràcter unicode de 16 bits. Son els mateixos que els ASCII amb el bit de major pes amb valor 0.	0 - 65535	'\u0000'
byte	Mida de 8 bits.	-2^7 fins 2^7-1 (-128 a 127)	0
short	Mida de 16 bits.	-2^{15} fins $2^{15}-1$ (-32768 a 32767)	0
int	Mida de 32 bits	2^{31} fins $2^{31}-1$ (-2147483648 a 2147483647)	0
long	Mida de 64 bits	-2^{63} fins $2^{63}-1$ (-9223372036854775808 a 9223372036854775807)	0L
float	Mida de 32 bits, contenen números en coma flotant de simple precisió Estándar IEEE 754-	(de 1.40239846e–45f a 3.40282347e+38f)	0.0f

	1985		
double	Mida 64 bits. Números en coma flotant de doble precisió Estándar IEEE 754-1985	(de 4.94065645841246544e–324d a 1.7976931348623157e+308d.)	0.0d

- **Caràcters:** En Java els caràcters no estan restringits al codi ASCII, son Unicode. Sempre va envoltat de cometes simples 'A','9'.
  Un tipus especial de caràcters son les anomenades **seqüencies d'escapament**, que s'utilitzen per representar caràcters de control o que no s'imprimeixen.
  Una seqüencia d'escapament esta formada per una barra invertida (\) i un caràcter.
  Mostrem els més utilitzats.

CARÀCTER	SEQÜENCIA D'ESCAPAMENT
Retorn de carro	\r
Tabulador horitzontal	\t
Nova línia	\n
Barra invertida	\\

- **Valors constants:** Quan declarem una variable de tipus final estem obligats a inicialitzar-la i no es podrà canviar el seu valor en cap punt del programa. La convenció diu que ha de escriure's en majúscules. En Java ho farem fora del bloc principal.
  EXEMPLES

```
final double PI=3.141592653589793;
final int MAX_CAPACITY=50;
```

- **Cadenes de caràcters o strings:** A part dels tipus anteriors, molt sovint tindrem la necessitat de emmagatzemar paraules o frases. Els estudiarem en breu.
  EXEMPLE

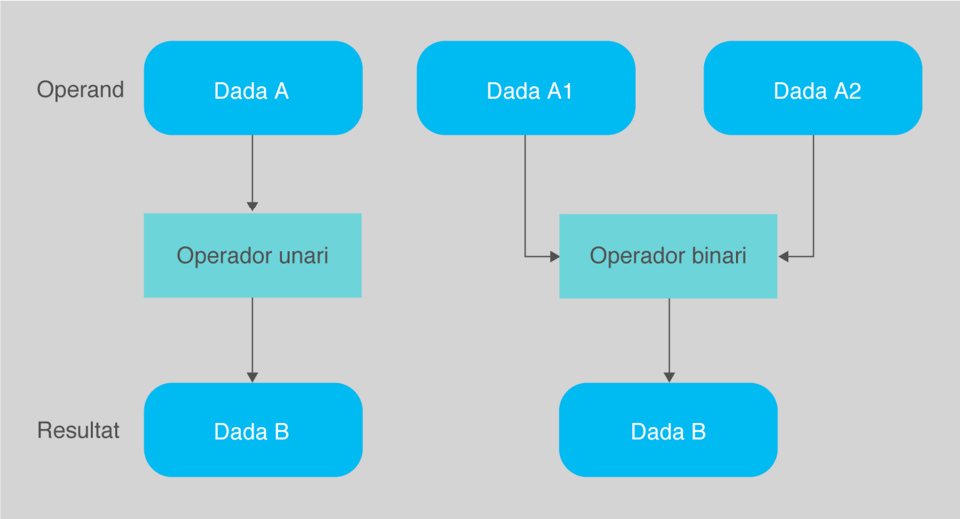
```
public class SampleString {

 public static void main(String[] args) {
       String cadena="Sóc una cadena";
       System.out.println(cadena);

 }

}
```

- **Paraules reservades:** Aquest llistat conté paraules que fa servir el propi llenguatge, i per tant el programador/ora no pot utilitzar com a identificador. Les paraules amb asterisc no s'utilitzen. Les paraules reservades es poden classificar per categories:
  - Tipus de dades: boolean, float, double, int, char
  - Sentencies condicionals: if, else, switch
  - Sentencies iteratives: for, do, while, continue
  - Tractament de las excepcions: try, catch, finally, throw

- Estructura de dades: class, interface, implements, extends
- Modificadors y control de accés: public, private, protected, transient
- Altres: super, null, this.

abstract	boolean	break	byte	byvalue*
case	cast*	catch	char	Class
const*	continue	default	do	Doublé
else	extends	false	final	Finally
float	for	future*	generic*	goto*
if	implements	import	inner*	instanceof
int	interface	long	native	New
null	operator*	outer*	package	Private
protected	public	rest*	return	Short
satatic	super	switch	synchronized	This
throw	transient	true	try	var*
void	volatile	while		

- **Operadors:** Tal i com hem vist a les expressions podem fer us d'operadors, anem a conèixer els diferents tipus i el seu comportament.



Tal i com es veu a la figura anterior podem dividir els operadors en dos tipus, unaris (amb un únic operand) o binaris (dos operands)

- **ARITMETICS:**

OPERADOR	NOM	EXEMPLE
+	SUMA	5+3
-	Diferencia	6-3
*	Producte	3*4
/	Divisió	60/3
%	Mòdul	10%3, 7.5%3

- **ASSIGNACIÓ: (=)** Es el operador mes freqüent, el farem servir per desar un valor a una variable.

  EXEMPLES

```
numero=20;
a=1234;
a=numero;
double area=calculaArea(radi);
superficie=amplada*al;
c=a=b;
```

L'operador d'assignació es pot combinar amb els operadors aritmètics.

EXPRESSIÓ	SIGNIFICAT
x+=y	x=x+y
x-=y	x=x-y
X*=y	x=x*y
x/=y	x=x/y

- **Concatenació de cadenes:** Si volem unir dos cadenes de caràcters farem servir l'operador (+)

  EXEMPLES

```
String nom="Pep ";
nom+="García";
System.out.println(nom);
// La sortida per pantalla seria Pep Garcia
```

- **Unaris:** Actuen sobre un únic operand, hem de tenir en conte si l'operador esta a la dreta o a l'esquerra per interpretar com funciona.
  - Increment ++

    i=i+1;

    i=i++;
  - Decrement --

    i=i-1;

    i=i--;

    Segons la posició del operador podem tenir comportaments diferents, estudiem els següents exemples.

Exemple 1

j=i++;

En aquest cas primer assignem el valor de i a j i posteriorment incrementem i, per tant es equivalent a escriure les següents sentencies.

```
j=i;
```

```
i++;
```

Exemple 2
j=++i;
En aquest exemple primer s'incrementa el valor de i, i posteriorment assignem el valor a j.

- **Operadors relacionals:** Es poden fer servir sobre qualsevol tipus bàsic de dades. No podrem fer servir amb Strings o cadenes.

OPERADOR	NOM	EXEMPLE
==	Igual	a==b
!=	Diferent	a!=b
>	Major que	a>b
<	Menor que	a<b
>=	Major o igual que	a>=b
<=	Menor o igual que	a<=b

- **Operadors lògics o booleans:** El seu nom es deu a l'àlgebra de Boole. S'operen sobre operands boolean i com a resultat obtenim un valor boolean. L'àlgebra de Boole és el conjunt de normes que ens diu com es poden combinar els true o false. Les taules de la veritat son una representació que ens permet esbrinar els resultats d'una operació a partir de totes les combinacions dels valors possibles dels operands. Les farem servir en expressions a les estructures de control que estudiarem al següent nucli formatiu.

OPERADOR	NOM
&	AND
|	OR
^	XOR
&&	AND lògica
||	OR lògica
!	NOT

Les taules que es mostren a continuació son la mateixa però expressades amb valors true o false o 1 i 0. El true es equivalent al 1 i el false equivalent a 0.

OPERANDS		RESULTATS OPERACIONS		
A	B	A&&B	A||B	!A
false	false	false	False	True
false	true	false	True	True
true	false	false	True	False
true	true	true	True	False

OPERANDS		RESULTATS OPERACIONS		
A	B	A&&B	A||B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

- **Precedència dels operadors:** La precedència dels operadors s'aplica en funció de la prioritat del operador, quin seria l'ordre lògic per avaluar la següent expressió?

  x= z+y*3;

  En qualsevol compilador primer faríem la multiplicació i després la suma. Si volem canviar, hem de trencar la prioritat amb els parèntesis.

  x= (z+y)*3;

  La següent taula ens indica la prioritat dels operadors:

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

- **Conversió de tipus:** Serà força habitual que dins una expressió barregem dades de diferents tipus. En aquests casos es necessari fer conversions de tipus per tal que les operacions siguin correctes.
  - **Conversió automàtica:** Es la desitjable, ja que mai es perd informació i el programador/ora no ha de especificar res. Es pot fer sempre i quan

    - **Els dos tipus de dades siguin compatibles.**
      - Tots els tipus numèrics son compatibles entre si, sense importar que siguin enters o reals.
      - El tipus char es compatible amb el tipus int.
      - El tipus boolean no es compatible amb cap altre tipus.
    - **El tipus de destí sigui més gran que el tipus origen (espai en memòria)**

- **Conversió explícita:** En els casos en que no es pugui fer de forma automàtica però necessitem fer-ho, hem de forçar-ho. Ho farem posant a la dada que volem canviar uns parèntesis davant seu, amb el tipus de dada al que volem transformar. Les regles que seguirem seran les següents:
  - **Entre números enters si el destí es major que l'origen, el valor resultant serà la resta(mòdul) de la divisió entera amb el rang del tipus de destí.**
  - **Si l'origen es un número real i el destí un enter, la part decimal es truncarà (pèrdua d'informació) i si la part entera no pot guardar-se al destí, s'aplicarà el criteri del mòdul.**
  - **Entre valors reals es guarda el màxim valor possible.**

```
public class Conversions {

	public static void main(String[] args) {
		//declaració i inicialitzacio de variables
		char c='a', c2;
		int i=23,i2;
		short s;
		double d;

		//assignacions amb conversió automàtica
		i2=c; //assignació correcta
		s=c; //incorrecte
		d=c; //correcte
		s=678;

		//assignacións amb conversions de casting o explícita
		double dou=134.54;
		int destí=(int) dou; //pèrdua d'informació destí contindrà 134
		dou=3.40282347E+50;
		float fl=(float)dou; //la informació supera la
                                                //capacitat d'un tipus float
		int in=257;
		byte b;
		b=(byte)in; //b contindrà 1 aplica la regla del modul
                            //257 mod 256=1

	}
}
```

Entrada de dades i visualització de resultats en Java

Hi ha moltes maneres d'entrar o mostrar les dades a l'usuari, només cal veure les interfícies gràfiques dels sistemes moderns, però aquí veureu la més simple: cadenes de text en pantalla o per teclat.

- Instruccions de sortida de dades per pantalla.

```
//Mostra a la pantalla el valor de x.
System.out.print(x);
//Fa el mateix, afegint un salt de línia.
System.out.println(x);
//Mostrem una cadena
String holaMon = "Hola, món!";
System.out.println(holaMon);
//Exemple de concatenació de variables amb cadenes de text
double dividend = 18954.74;
double divisor = 549.12;
double resultatDivisio = dividend/divisor;
String missatge = "El resultat obtingut és " + resultatDivisio + ".";
System.out.println(missatge);
//Mostra text usant caràcters de control.
System.out.println("Linia 1\n\tLínia2\nLínia 3");
```

- Instruccions d'entrada de dades: Explicar com introduir dades pel teclat amb Java sense haver explicat una sèrie de conceptes es una mica complicat. Per tant ho farem des de el punt de vista pràctic, agafarem els següents exemples com a patró i els aplicarem sempre que ho necessitem. Estudiem el següent exemple.

```
import java.util.Scanner;
public class EntradaSortida {

	public static void main (String[] args) {
		//S'inicialitza la biblioteca.
		Scanner lector = new Scanner(System.in);
		//Es posa un missatge de benvinguda.
		System.out.println("Anem a sumar dos nombres enters");
		//Es llegeix un valor enter per teclat.
		//S'espera que es pitgi la tecla de retorn.
		System.out.print("Escriu un nombre i pitja la tecla de
retorn:");
		int primerEnter = lector.nextInt();
		lector.nextLine();
		//Es torna a fer...
		System.out.print("Torna a fer-ho: ");
		int segonEnter = lector.nextInt();
		lector.nextLine();
		//Fem l'operació.
		int resultat = primerEnter + segonEnter;
		//Imprimeix el resultat per pantalla!
		//S'usa l'operador suma entre una cadena de text i un enter
		System.out.println("La suma dels dos valors és "+ resultat
+".");
		}

	}
```

De moment, n'hi ha prou que aprengueu com podeu llegir dades d'una en una, usant diferents línies de text per entrar cada dada individual. Per fer-ho, cal alternar les instruccions de lectura enumerades a la taula que mostrem a continuació, segons el tipus de dada que es vol llegir des del teclat, i la instrucciólector.nextLine();, tal com es veu a l'exemple.

INSTRUCCIONS DE LECTURA PER TECLAT

Instrucció	Tipus de dada llegida
lector.nextByte()	byte
lector.nextShort()	short
lector.nextInt()	int
lector.nextLong()	long
lector.nextFloat()	float
lector.nextDouble()	double
lector.nextBoolean()	boolean
lector.next()	String

Aquest mecanisme no permet llegir caràcters individuals. Recordeu, però, que des del punt de vista del tipus de dada, una cadena de text que conté un únic caràcter **no** és el mateix que un caràcter individual. Un altre aspecte important és que la manera com reconeix els valors reals, si interpreta els decimals separats amb un punt o una coma (per exemple, 9.5 o 9,5), depèn de la configuració local del sistema. En les configuracions catalana i espanyola se separen amb una coma.

Si feu proves us adonareu que en aquest programa, si us equivoqueu i introduïu un valor diferent d'un enter, el programa s'atura immediatament i mostra un missatge d'error per pantalla. Això és normal, ja que la dada escrita per teclat ha d'encaixar amb el tipus de dada que llegeix la instrucció usada.