Programació

UT1.3 Variables

¿Només usem literals?

- La resposta és no. Necessitem que la informació siga canviant (variable) durant l'execució del programa.
- Variable: és un **dada emmagatzemada en la memòria** que pot veure modificat el seu valor en qualsevol moment durant l'execució del programa.

  RAM → Volàtil i limitada → **No utilitzar a la babalà**

Funcionament de la memòria. Símil

- Resoldre una multiplicació de dos xifres.
- Només hi ha un espai limitat per a realitzar operacions definitives i auxiliars

Funcionament de la memòria. Símil

- És un espai compartit
- Dades no persistents
- Dades en format binari
- Dades en cel·les de memòria

Declaració de variables

- Tota variable ha de ser **declarada** per poder ser usada
- La sintaxi de Java és: TIPUS + IDENTIFICADOR + (VALOR INICIAL ASSIGNAT)

int valor; **valor = 5;**	**int valor = 5;**

RECORDA: Utilitzem "==" per a COMPARAR si dos valors son iguals.
Utilitzem "=" per a ASSIGNAR un valor a una variable.

Declaració de variables

```
int unEnter = 2;
```

- Com hem vist anteriorment el tipus de dada **enter** en Java utilitza **4 bytes** de memòria.

- Quan declarem una variable de tipus enter es reserven 4 bytes, com s'aprecia a la imatge, on es podrà guardar un valor enter que **podrà canviar durant l'execució del programa.**

Identificador (nom de variable)

- No pot contindre espais
- No pot començar amb un número
- És **aconsellable** utilitzar SEMPRE caràcters de l'alfabet anglès
- Recomanable la notació "camell" (lowerCamelCase)
- Recomanable utilitzar noms identificatius
- No es poden utilitzar **paraules reservades**.

Paraules reservades

abstract, continue, for, new, switch, assert, default, goto, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp, volatile, const, float, native, super, while

Ús de variables

- Tota variable declarada es referencia pel seu identificador, el qual representarà el valor emmagatzemat en cada instant de l'execució del programa.

```
int variable1, variable2;

variable1 = 23;
variable1 = 32;
variable2 = variable1 + 5;
variable1 = 40;
System.out.println(variable2); // Imprimirà 37
```

- Cada variable té el seu context (àmbit o scope)

Ús de variables. Exemple

```
//Un programa que calcula una divisió i una suma, completat
public class DivideixISuma {
  public static void main(String[] args) {
    double dividend = 20.0;
    double divisor = 6.0;
    double sumarAlFinal = 3.0;
    //Les variables es poden usar com a literals dins una expressió.
    //Aquí és equivalent a fer (20.0/6.0) + 3.0
    System.out.println((dividend/divisor) + sumarAlFinal);
  }
}
```

RECORDA: És important que executes aquestos exemples, on es pot comprovar el funcionament de les variables, i com les imprimeix per consola (println) el llenguatge de programació Java.

Pràctica 7

- Modifica el programa anterior perquè faça ús d'una variable addicional que emmagatzeme, en primer lloc el resultat de la divisió i posteriorment el de la suma total, mostrant després per pantalla el resultat.

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

Pràctica 8

- Realitza un programa amb dos variables que, sense utilitzar literals excepte en el cas de les seues inicialitzacions, vaja calculant i imprimint successivament els 5 primers valors de la taula de multiplicar del 4.

  Podeu utilitzar operadors aritmètics i d'assignació, si voleu.

Operacions amb variables

- Les mateixes que aplicàvem als literals segons el seu tipus.
- A més existeixen 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.

En el siguiente ejemplo se puede observar la utilización de operadores de asignación:

```
int  num=5;
num += 5; //  num = 10, equivale a num = num + 5
```

Operacions amb variables

Els operadors d’increment (++) i decrement (--) s’utilizen per a sumar 1 o restar 1 respectivament a una variable.

Els operadors d'increment i decrement es poden col·locar abans (preincrement/predecrement) o després (postincrement/postdecrement) de la variable a la qual s'apliquen.

Si col·loqueu un operador d'increment o de decrement abans de la seua variable, l'operador s'aplica abans que s'avalue la resta de l'expressió. Si col·loqueu l'operador després de la variable, l'operador s'aplica després d'avaluar l'expressió.

Operacions amb variables

Exemple:

```
int a = 5;
int b = 3;
int c = a * b++;    // Després d'executar la línia a=5, b=4 i c=15
```

Equival a: *c=a*b; b=b+1;*

```
int a = 5;
int b = 3;
int c = a * ++b;    // Després d'executar la línia a=5, b=4 i c=20
```

Equival a: *b=b+1; c=a*b;*

```
int a = 5;
int b = 3;
int c = --a * b++;    // Després d'executar la línia a=4, b=4 i c=12
```

Equival a: *a=a-1; c=a*b; b=b+1;*

```
int a = 5;
int b = 3;
int c = a-- * ++b;    // Després d'executar la línia a=4, b=4 i c=20
```

Equival a: *b=b+1; c=a*b; a=a-1;*

Constants

- Són dades que tenen la particularitat que el seu valor, dins del programa, només pot ser llegit (no modificat). Utilitzem la paraula reservada **“final” per a declarar-les**.

```
final double NUMERO_PI = 3.1416;
```

- Es recomana usar sempre lletres majúscules per a l’identificador, separant les paraules pel símbol '_' (UPPER_SNAKE_CASE)

Constants. Exemple

```
// Calcula el preu de comprar diverses coses.
public class CalculadoraDePreu{
    public static void main(String[] args) {
        final double IVA = 0.21;
        double preuFinal = 0.0;
        // Va sumant els preus dels productes
        // ...
        preuFinal = preuFinal + (preuFinal * IVA);
        System.out.println(preuFinal);
    }
}
```

IMPORTANT: Només podem assignar el valor a “IVA” una ÚNICA volta. Si intentem assignar-li després un altre valor ens apareixerà un error de compilació.

Conversions de tipus implícites

- Implícita (**type coercion**): compatibilitat entre tipus

Enters

Reals

Java fa la conversió **de forma automàtica**

```
float real = 100;
double doble = real;
```

El literal 100 és un enter i ***real*** és de tipus float. [Guarda 100f]
La variable ***doble*** és de tipus diferent a la variable ***real.*** [Guarda 100d]

IMPORTANT: Si guardem una dada entera en una variable real no hi ha cap pèrdua d'informació. Al contrari (guardar una real en una entera) no es pot fer de forma implícita ja que comporta pèrdua d'informació (es perdria la part fraccionaria).

Conversions de tipus explícites

- Explícita (**type casting**): el programador la força

(tipus) expressió

```
double valor = 5.7;
int valor2 = (int) valor;
// valor val 5.7 i valor2 val 5

int lletra = 97;
char lletra2 = (char) lletra;
// lletra val 97 i lletra2 val 'a'
```

Eixida de dades per pantalla

- System.out.print(); → mostra per consola la informació
- System.out.println(); → igual que "print" però afegint un **salt de línia al final**
- System.out.printf(); → mostra la informació amb format.

Més informació sobre printf: https://www.baeldung.com/java-printstream-printf

Cadenes de text (String)

Fixa't que quan el nom del tipus de dada comença per majúscula ens indica que no és un tipus de dada primitiu

- Important: NO ÉS UN TIPUS PRIMITIU
- Representa una cadena de caràcters

```
String text = "Prova de text";
System.out.println(text);
```

```
//Mostra el resultat d'una divisió simple entre reals.
public class Dividir{
  public static void main(String[] args) {
    double dividend = 18954.74;
    double divisor = 549.12;
    double resultatDivisio = dividend/divisor;
    String missatge = "El resultat obtingut és " + resultatDivisio + ".";
    System.out.println(missatge);
  }
}
```

Caràcters d'escapament (especials)

- \t → tabulador
- \n → salt de línia
- \' → cometa simple
- \" → cometa doble
- \\ → contrabarra

```
//Mostra text usant caràcters de control.
public class CaractersControl{
  public static void main(String[] args) {
    System.out.println("Linia 1\n\tLínia2\nLínia 3");
  }
}
```

Pràctica 9

Còpia este programa i observa l'eixida per pantalla. Fes modificacions perquè s'utilitzen també la resta de caràcters especials

Cadenes de text (String)

Important: Com que no es un tipus primitiu mai s'ha de comparar el contingut amb operadors primitius. Més endavant ho vorem en detall.

```
String s1 = "HOLA";

System.out.println(s1 == "Hola");

System.out.println(s1.equals("Hola"));
```

Pràctica 10

- Fes un programa que mostre per pantalla de manera tabulada la taula de veritat d'una expressió de disjunció entre dos variables booleanes.

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

Entrada simple de dades per teclat. Classe Scanner

```
import java.util.Scanner //Importa la classe Scanner
public class Salari{
   public static void main(String [] args){
      String nom;
      int hores;
      double pagamentPerHora, pagamentTotal;

      Scanner teclat = new Scanner(System.in); // Crea lector
      System.out.println("Com et dius? ");
      nom = teclat.nextLine(); // Fa la lectura d'una línia
      System.out.println("Quantes hores treballes esta setmana? ");
      hores = teclat.nextInt();
      System.out.println("Quant et paguen per hora?");
      pagamentPerHora = teclat.nextDouble(); // Fa la lectura d'un real
      teclat.close(); // Tanca l'objecte lector del teclat
      pagamentTotal = hores * pagamentPerHora;
      System.out.println("Hola " + nom);
      System.out.println("El teu sou és de " + pagamentTotal + " €");
   }
}
```

Mètodes d'Scanner

`lector.nextByte()`	byte
`lector.nextShort()`	short
`lector.nextInt()`	int
`lector.nextLong()`	long
`lector.nextFloat()`	float
`lector.nextDouble()`	double
`lector.nextBoolean()`	boolean
`lector.next()`	String

Pràctica 11

- Fes un programa que mostre per pantalla la multiplicació de 3 reals introduïts per teclat

Pràctica 12

1) Crea el següent programa.
2) Executa'l i analitza el que passa. Raona el comportament del programa i troba una solució.

```java
public class ScannerFalla {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("Insert a number: ");
        input.nextInt();
        System.out.print("Text1: ");
        input.nextLine();
        System.out.print("Text2: ");
        input.nextLine();
    }

}
```

Comentaris

Un comentari és un text que s'escriu dins d'un programa per tal de facilitar-ne la comprensió. S'utilitzen per explicar el codi font.

En Java es poden utilitzar tres tipus de comentaris:

- Comentari tradicional
- Comentari d’una sola línea
- Comentari de documentació

Comentari tradicional

Són els mateixos que es fan servir en C/C++

Les característiques dels comentaris tradicionals Java són les següents:

- Comença amb els caràcters /* i acaba amb */.
- Poden ocupar més d' una línia i poden aparèixer en qualsevol lloc on puga aparèixer un espai en blanc.
- No poden aniuar-se.

```
public class ExempleCommentari {
    public static void main(String[] args) {
        /* Aquest és un comentari tradicional de diverses línies.
           Pot ser utilitzat per explicar el propòsit del codi
           o per desactivar temporalment una secció de codi.
        */
        System.out.println("Hola, món!");
    }
}
```

Comentari d’una sola línia

Les característiques dels comentaris d' una línia són les següents:

- Comencen amb una doble barra (//)
- Es poden escriure al principi de la línia o a continuació d' una instrucció.
- No tenen caràcter d' acabament.

```
public class ExempleCommentari {
    public static void main(String[] args) {
        System.out.println("Hola, món!"); // Aquest és un comentari d'una sola línia
    }
}
```

Comentari de documentació

Són comentaris especials per generar documentació del programa.

També es coneixen com a comentaris Javadoc.

Comencen amb /** i acaben amb */

```
public class ExempleCommentariDocumentacio {
    /**
     * Mètode principal que s'executa quan s'inicia l'aplicació.
     *
     * @param args Array de cadenes de caràcters que es passen com a arguments de línia d'ordres
     */
    public static void main(String[] args) {
        System.out.println("Hola, món!"); // Imprimeix un missatge a la consola
    }
}
```