BLOC 2:

1. Estructura d’un programa informàtic

a. Variables. Tipus i utilitat.

b. Conversions de tipus de dades.

c. Constants. Tipus i utilitat.

d. Operadors del llenguatge de programació.

- Variables. Tipus i utilitat.

- A Java, una variable és un espai de memòria que reservem per guardar un valor/dada en concret.
- Exemples de variables:
  - **int** numero **=** 2**;**
  - **String** cadena **=** "Hola"**;**
  - **long** decimal **=** 2.4**;**
  - **boolean** flag **= true;**
- Les variables s'identifiquen amb un nom (identificador), per a que el program pugui accedir a la dada concreta que té emmagatzemada.
- Contenen un tipus de dada en concret, per exemple, nombres decimals, nombres enters, text, objectes, etc…
- Cada variable conté un rang de valors que pot admetre. Si el valor que s’emmagatzema en aquella variable no pertany al rang assignat, el programa ens donarà un error.
- Les variables segueixen unes regles en concret:
  - L’identificador d’una variable no sol començar en $.
  - No es pot utilitzar un valor booleà (cert/fals) ni un valor nul (null)per identificar una variable.
- Segons la zona on col·loquem una variable, aquesta actuarà d’una manera o d’una altra.
- Tipus de variables:
-
  - Variables locals:
    - Son variables que es declaren dins d’un bloc/mètode.
    - Aquestes variables es creen al entrar al bloc/mètode i es destrueixen al sortir.

- Sol podem accedir a aquestes variables entrant primer al bloc/mètode on es troben.
- Exemple de variable local:

```
public class StudentDetails
{
    public void StudentAge()
    {   //variable local age
        int age = 0;
        age = age + 5;
        System.out.println("La edad del estudiante es : " + age);
    }

    public static void main(String args[])
    {
        StudentDetails obj = new StudentDetails();
        obj.StudentAge();
    }
}
```

- Variables d'instància:
  - Son variables no estàtiques i es declaren a una classe fora de qualsevol bloc/mètode.
  - Aquestes variables es creen quan es crea un objecte dins d'una classe i es destrueix quan aquest objecte finalitza.
  - A diferència de les variables locals, podem utilitzar especificadors d'accés per a aquestes variables.
  - Exemple de variable d'instància:

```
import java.io.*;
class Points
{
    //Estas variables son variables de instancia.
    //Estas variables están en una clase y no están dentro de ninguna función/método
    int engPoints;
    int mathsPoints;
    int phyPoints;
}

class PointsDemo
{
    public static void main(String args[])
    {   //primer objeto
        Points obj1 = new Points();
        obj1.engPoints = 50;
        obj1.mathsPoints = 80;
        obj1.phyPoints = 90;

        //segundo objeto
        Points obj2 = new Points();
        obj2.engPoints = 80;
        obj2.mathsPoints = 60;
        obj2.phyPoints = 85;
```

- Variables estàtiques:
    - També es coneixen com a ‘variables de classe’.
    - Sol podem tenir una còpia d’una variable estàtica dins de cada classe, independentment dels objectes que creem.
    - Es creen a l’inici de l’execució del programa i es destrueixen automàticament quan finalitza l’execució.
    - Per accedir a una variable estàtica hem d’introduïr “*nom_classe.nom_variable;*”
    - Exemple de variable estàtica:

```
import java.io.*;
class Emp {

   // salario como variable estatica
   public static double salary;
   public static String name = "Alex";
}

public class EmpDemo
{
     public static void main(String args[]) {

      //acceder a la variable estatica sin objeto
      Emp.salary = 1000;
       System.out.println(Emp.name + " tiene un salario promedio de: " + Emp.salary);
   }

}
```

- Conversions de tipus de dades:

- Una conversió es pot definir com la transformació d’una variable/objecte d’un tipus a un altre.
- La sintaxis d’una conversió és: “*tipus_variable_nova = (nou_tipus) variable_antiga;*”
- No es pot convertir qualsevol tipus de variable en qualsevol tipus de variable. Les conversions que son segures són les següents:
    - *byte → double, float, long, int, char, short*
    - *short → double, float, long, int*
    - *char → double, float, long, int*
    - *int → double, float, long*
    - *long → double, float*
    - *float → double*

- Constants. Tipus i utilitat.

- Una constant és una variable del sistema que manté el seu valor al llarg de tota la vida del programa. Per exemple, si volem indicar els dies d'una setmana (sempre tindrà un valor de 7 parlem del que parlem) construirem la constant així:

  final dies_setmana = 7;

- Les constant les utilitzarem quan volguem fixar el valor d'una variable que en principi no ha de canviar durant l'execució del programa, per exemple l'identificador d'un usuari, una data de naixement, etc…

- Operadors del llenguatge de programació.

- Java proporciona la possibilitat d'utilitzar diferents tipus d'operadors. Es classifiquen segons la funció que fan de la següent manera:
  - Operadors aritmètics
    - Serveixen per realitzar les operacions aritmètiques següents:
      - Multiplicació = *****
      - Divisió = **/**
      - Mòdul = **%**
      - Addició/Suma = **+**
      - Resta = **-**
    - Exemple:

```
// Programa Java per il·lustrar
// operadors aritmètics
public class operators
{
  public static void main (String [] args)
  {
    int a = 20, b = 10, c = 0, d = 20, i = 40, f = 30;
    String x = "Thank", i = "You";

    // Operador + i -
    System.out.println ( "a + b =" + (a + b));
    System.out.println ( "a - b =" + (a - b));

    // L'operador + si es fa servir amb strings
    // concatena les cadenes donades.
    System.out.println ( "x + i =" + x + i);
```

- Operadors unaris
    - S'usen per incrementar, disminuir o negar un valor.
        - Unari (-) = Negar valors
        - Unari (+) = Donar valors positius (només per passar de nombre negatiu a positiu)
        - Unari (++) = Incrementar un valor (+1)

```
// Programa Java per il·lustrar
// operadors unaris
public class operators
{
  public static void main (String [] args)
  {
    int a = 20, b = 10, c = 0, d = 20, i = 40, f = 30;
    boolean condition = true;

    // operador de pre-increment
    // a = a + 1 i llavors c = a;
    c = ++ a;
    System.out.println ( "Valor de c (++ a) =" + c);

    // operador de post-increment
    // c = b llavors b = b + 1 (b passa a ser 11)
    c = b ++;
    System.out.println ( "Valor de c (b ++) =" + c);
```

- Operador d'assignació
    - Es fa servir per assignar un valor a qualsevol variable.

```
// Programa Java per il·lustrar
// Operadors d'assignació
public class operators
{
    public static void main (String [] args)
    {
        int a = 20, b = 10, c, d, e = 10, f = 4, g = 9;

        // operador d'assignació simple
        c = b;
        System.out.println ( "Valor de c =" + c);
```

- Operadors relacionals
    - S'utilitzen per verificar relacions. Operen amb *booleans*.
        - **= =** (igual a) = *true* si el valor esquerre és igual al dret.
        - **! =** (no igual a) = *true* si el valor esquerre no és igual al dret.
        - **<** (menor que) = *true* si el valor esquerre és menor que el dret.
        - **< =** (menor o igual que) = *true* si el valor esquerre és menor o igual que el dret.
        - **>** (major que) = *true* si el valor esquerre és major que el dret.
        - **> =** (major o igual que) = *true* si el valor esquerre és major o igual que el dret.

- Operadors lògics
    - S'utilitzen per a realitzar operacions "lògiques AND" i "lògiques OR". La segona condició no s'avalua si la primera és falsa.
        - **AND lògic (&&) =** retorna quan les condicions són correctes.
        - **O lògic (||) =** retorna si una condició és correcta.

- Operador ternari
    - Si la condició s'avalua coma correcta, llavors executarà les instruccions després del "?", i si no, executa les instruccions després del ":".

- Operadors bit a bit
    - S'utilitzen per manipular bits d'un número individualment. Es poden fer servir amb qualsevol nombre enter.
        - **AND (&) =** 1 & 1 = 1 // 0 & 0 = 0
        - **OR (|) =** 1 | 0 = 1 // 0 | 0 = 0
        - **XOR (^) =** 1 ^ 0 = 1 // 1 ^ 1 = 0 // 0 ^ 0 = 0
        - **NOT (~) =** 01 = 10 // 10 = 01 (inverteix els bits)

- Operadors shift
    - Es fan servir per desplaçar bits cap a l'esquerra o cap a la dreta. Es poden usar quan hem de multiplicar/dividir un nombre per dos.
        - **<<** = Desplaça bits cap a l'esquerra i completa els bits desplaçats amb "0".
        - **>>** = Desplaça bits cap a la dreta i completa els bits desplaçats amb "0".
        - **>>>** = Desplaça bits cap a la dreta i completa els bits desplaçats amb "0". El bit de l'extrem esquerre es converteix en "0".

```java
// Programa Java per il·lustrar
// operadors shift
public class operators
{
    public static void main (String [] args)
    {

        int a = 0x0005;
        int b = -10;

        // operador de desplaçament a l'esquerra
        // 0000 0101 << 2 = 0001 0100 (20)
        // similar a 5 * (2 ^ 2)
        System.out.println ( "a << 2 =" + (a << 2));
```

- Operador d'instància
    - S'utilitza per a verificar el tipus. Es pot usar per a provar si un objecte és una instància de una classe, una subclasse o una interfície.

- Precedència i Associativitat d'Operadors
    - S'utilitzen quan es fan equacions híbrides que involucren més d'un tipus d'operador. Aquestes regles determinen quina part de l'equació considerar primera, ja que poden haver moltes valoracions diferents per a la mateixa equació.

- Tipus de dades (simples):

- ***byte:*** com el seu nom indica, emmagatzema un byte (8 bits) d'informació.
- ***short:*** utilitza el doble d'informació que la variable *byte*.
- ***int:*** emmagatzema 4 bytes d'informació i és la variable numèrica més utilitzada.
- ***long:*** conté 8 bytes (64 bits) d'informació.
- ***float:*** emplea 32 bits.
- ***double:*** és similar al *float* però emmagatzema 64 bits en lloc de 32.
- ***boolean:*** serveix per definir tipus de dades booleanes, és a dir, dades que tenen un valor de true/false. Ocupa 1 bit. (el seu valor per defecte és *false*)
- ***char:*** representa un caràcter senzill de 16 bits.