





1ª Tema Java

 Proyecto

 Paquetes

 Clases

 Código

-
- Workspace - Carpeta donde se guardan todos los programas que hagamos. **eclipse-workspace**
 - .metadata **NO TOCAR**

Un programa en JAVA se llama Proyecto

1. **Editar el programa** (Escribir el código)
2. **Compilarlo**
3. **Ejecutarlo**
4. **Depurarlo**

- **Llevar la carpeta HolaMundo dentro de workspace para hacer cosas en casa copiando y pegando en USB o mandar por mail comprimiendo en 7zip**
- **En casa file → importar → carpeta general → existing projects into workspace en eclipse**

```
package primerhola;  
  
public class Principal {
```

```

public static void main(String[] args) {
    // TODO Auto-generated method stub

    int edadUsuario = 34;
    short anyo = 2022;
    byte mes = 9;
    System.out.println("Hello World!");
    System.out.println("Hola me llamo ALex \n"
        + "Tengo una casa"
        + "guapa \n"
        + "Mi madre es alemana y mi padre español \n"
        + "Mi hermano");
}

```

```

}

```

TIPOS DE DATOS

Tipos primitivos (NÚMEROS)

NOMBRE	TIPO	OCUPA	RANGO APROXIMADO
byte	Entero	1 byte	-128 A 127
short	Entero	2 bytes	-32768 A 32767
int	Entero	4 bytes	2*10 ⁹
long	Entero	8 bytes	Muy grande
float (8 decimales)	Decimal simple	4 bytes	Muy grande
double (16 decimales)	Decimal doble	8 bytes	Muy grande
char (comilla simple su valor)	Carácter simple	2 bytes	
boolean	Valor true or false	1 byte	

Syntax: tipo nombreVariable = valor; ej: int edadUsuario = 27;

- En la variable primera letra en minúscula y después podremos utilizar mayúscula para distinguir. Ej: edadUsuario

- **= es una asignación**

Este año por defecto utilizaremos prácticamente siempre pero en un futuro no se debería ya que ocupa mucho

Tarea hacer una guía de nuestro primer hola mundo completo (ayer) Desde que abrimos eclipse (workspace)

- Eclipse entiende con **float** que el valor es **double** por defecto por ello tendremos que poner una **F** detrás de los decimales para que entienda que es float. Ej: 1.26f
- Hasta que no se diga lo contrario todas las variables del programa se declaran en las primeras líneas

```
1º
int edad =23;
float precio=1.26f;
double sueldoBase=2500.98;
char letraPiso='D';
boolean mayorEdad=false;
2º
System.out.println("Bienvenidos");
```

- **String:** Es una variable para una cadena de texto. Primera mayúscula y su valor tiene que ir entre “comillas”

int: resérvame un espacio en la memoria que voy a identificar con x y guarda dentro y

Deberes: Como mostrar resultados de la suma

Para sumar en java ponemos primero las variables de los numeros y la de su suma y después lo reflejamos en la consola del tal manera:

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int num1=3;
    int num2=2;
    int suma=0;

    suma=num1+num2;
    // Para comentario corto
    /*Cuando el comentario es más de una línea*/

    System.out.println("Bienvenidos, la suma total de "+num1+" y de "+num2+" es: ");
}
```

```

        + ""+suma+"");

//¿Se puede sumar dentro del println?
    System.out.println("La suma es "+(num1+num2));
//Así se puede pero si queremos guardarnos el valor hay que hacerlo en la variable +suma

```

- El + en Java sirve para sumar y también concatenar
- Si ponemos solo el nombre de la variable sin texto, también se va a declarar
- El resultado de las operaciones con variables siempre se debe guardar en una variable
- Las operaciones con variables se hacen en la línea en la que hace falta. No es necesario realizar todas las operaciones al principio y poner todos los (\n) al final

Nota: Es preferible guardar más variables de la cuenta

```

//Para dividir
    double num1=10;
    double num2=4;
    double suma=0;
//División de 2 números
    double div=0.0;
    div=num1/num2;
System.out.println("La división es :"+div);

```

Nota: Cuando declaramos variables del mismo tipo, solo cuando son del mismo tipo, se pueden declarar varias variables en la misma línea

```

int edad=34, largo=50, ancho=60;

```

```

//redondear a 2 decimales

System.out.printf("El precio es %.2f", num1)

//2f porque son 2 decimales

System.out.printf("El precio es %.2f y %.3f \n", num1, num2)

```

▼ Package: utilidades / Class: Leer

```

package utilidades;

import java.io.*;

public class Leer{

    public static String dato()
    {
        String sdato=" ";
        try{
            InputStreamReader isr=new InputStreamReader(System.in);
            BufferedReader flujoE = new BufferedReader(isr);
            sdato=flujoE.readLine();
        }
        catch (IOException e)
        {
            System.out.println("Error "+e.getMessage());
        }
        return sdato;
    }

    public static int datoInt()
    {
        return Integer.parseInt(dato());
    }

    public static float datoFloat()
    {
        return Float.parseFloat(dato());
    }
    //Leer un char por teclado

    public static char datoChar( ){
        char c = ' ';
        try {
            java.io.BufferedInputStream b = new BufferedInputStream(System.in);
            c = (char) b.read();
        } catch (IOException e) {
            System.out.println("Error al leer");
            e.printStackTrace();
        }
        return c;
    }

    public static long datoLong()
    {
        return Long.parseLong(dato());
    }

    public static short datoShort()

```

```

{
    return Short.parseShort(dato());
}

public static double datoDouble()
{
    return Double.parseDouble(dato());
}
}

```

- Para poner en printf un String **%s** y si quiero imprimir una variable entera **%d** (int), para char **%c**
- El operador **%** se llama operador **módulo**, y nos da el resto de una división

\t ---> Tabulador (espacios)

Operadores	Precedencia
postfix	<i>expr++ expr--</i>
unarios	<i>++expr --expr</i>
multiplicativos	<i>* / %</i>
aditivos	<i>+ -</i>
de movimiento (shift)	<i><< >> >>></i>
relacionales	<i>< > <= >= instanceof</i>
de igualdad	<i>== !=</i>
AND a nivel de bit (bitwise & AND)	
OR exclusivo a nivel de bit ^ (bitwise exclusive OR)	
OR inclusivo a nivel de bit (bitwise inclusive OR)	
AND lógico	<i>&&</i>
OR lógico	<i> </i>
ternarios	<i>? :</i>
de asignación	<i>= += -= *= /= %= &= ^= = <<= >>= >>>=</i>

Operador	Uso	Equivalente a
+=	<i>op1 += op2</i>	<i>op1 = op1 + op2</i>
-=	<i>op1 -= op2</i>	<i>op1 = op1 - op2</i>
*=	<i>op1 *= op2</i>	<i>op1 = op1 * op2</i>
/=	<i>op1 /= op2</i>	<i>op1 = op1 / op2</i>
%=	<i>op1 %= op2</i>	<i>op1 = op1 % op2</i>
&=	<i>op1 &= op2</i>	<i>op1 = op1 & op2</i>

$op1 += op2 \rightarrow op1 = op1 + op2$

$suma++ \rightarrow suma = suma + 1$

- Para escribir pi no hace falta variable; Cuando la mencionemos en una operación se verá así

```
longitud=radio1*multi*Math.PI;
```

- `final double num1=23.43`: Le dice a JAVA que es una constante



OJO: VISIBILIDAD Y VIDA DE LAS VARIABLES: LA VISIBILIDAD DE UNA VARIABLE (O SCOPE) ES LA PARTE DEL CÓDIGO DE UNA APLICACIÓN DONDE LA VARIABLE ES ACCESIBLE Y PUEDE SER UTILIZADA. EN JAVA, LAS VARIABLES NO PUEDEN DECLARARSE FUERA DE UNA CLASE Y POR LO GENERAL, TODAS LAS VARIABLES QUE ESTÁN DENTRO DE UN BLOQUE, ES DECIR, ENTRE DOS LLAVES {}, SON VISIBLES Y EXISTEN DENTRO DE DICHO BLOQUE. AL ACABAR EL TROZO DE CÓDIGO CON LA LLAVE DE CIERRE, MUEREN COMO BELLACAS.

`Math.pow(3,4) / Math.pow(base, exp)` → 3 elevado a 4

`Math.PI` Para poner PI