

TRABAJANDO CON ECLIPSE	2
Crear un proyecto Java con Eclipse	2
Crear paquetes dentro del proyecto	3
Crear clases java (.java)	4
Crear directorio de paquetes en Java	5
Nomenclatura camelCase	7
TIPOS DE DATOS	8
Datos de tipo numérico (números)	8
Datos de tipo alfanuméricos (letras, palabras, etc)	9
Datos tipo booleano	10
OPERADORES.....	11
Operadores de asignación.....	11
Operadores lógicos	11
==	11
!=	12
>	13
<	13
>=	13
<=	13
Operador AND	14
Operador OR	15
COMPARANDO STRINGS	16
COMENTARIOS EN JAVA.....	17
Comentarios en 1 línea	17
Comentarios multilínea	18

TRABAJANDO CON ECLIPSE

Crear un proyecto Java con Eclipse

1º Click en "File"

2º Click en "New"

3º Click en "Java Project"

4º Nombráis vuestro proyecto Java como queráis (evitad los espacios en blanco)

5º Desmarcáis la casilla "Create module-info.java file"

6º Click en "Finish"

New Java Project

Create a Java project in the workspace or in an external location.

Project name: insertadElNombreQueQuerais

☒ Use default location

Location: C:\Users\diego\Desktop\Workspaces\workspace_eclipse\insertadElNombreQueQuerais [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-16

☐ Use a project specific JRE: jre

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☐ Create separate folders for sources and class files [Configure default...](#)

Working sets

☒ Add project to working sets [New...](#)

Working sets: [Select...](#)

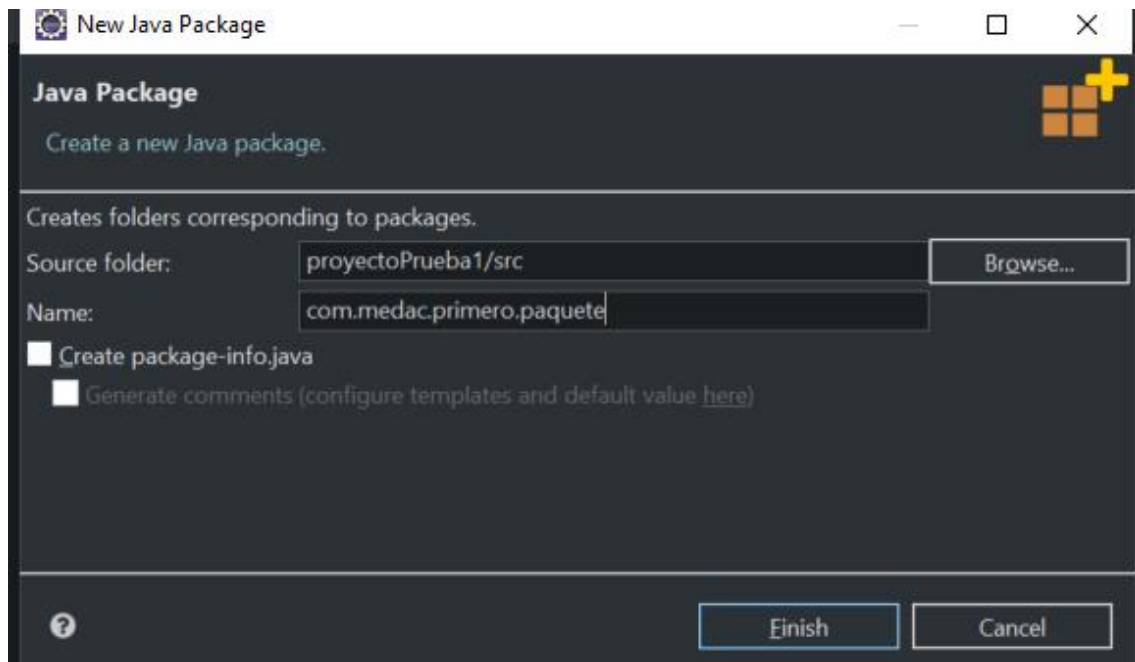
Module

☐ Create module-info.java file

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

Crear paquetes dentro del proyecto

- 1º Desplegáis vuestro proyecto Java
- 2º Click derecho en “src”
- 3º Click en “New” -> “Package”
- 4º Creáis vuestro directorio de paquetes como se explica [aquí](#)
- 5º Desmarcáis la casilla “Create package-info.java”



Crear clases java (.java)

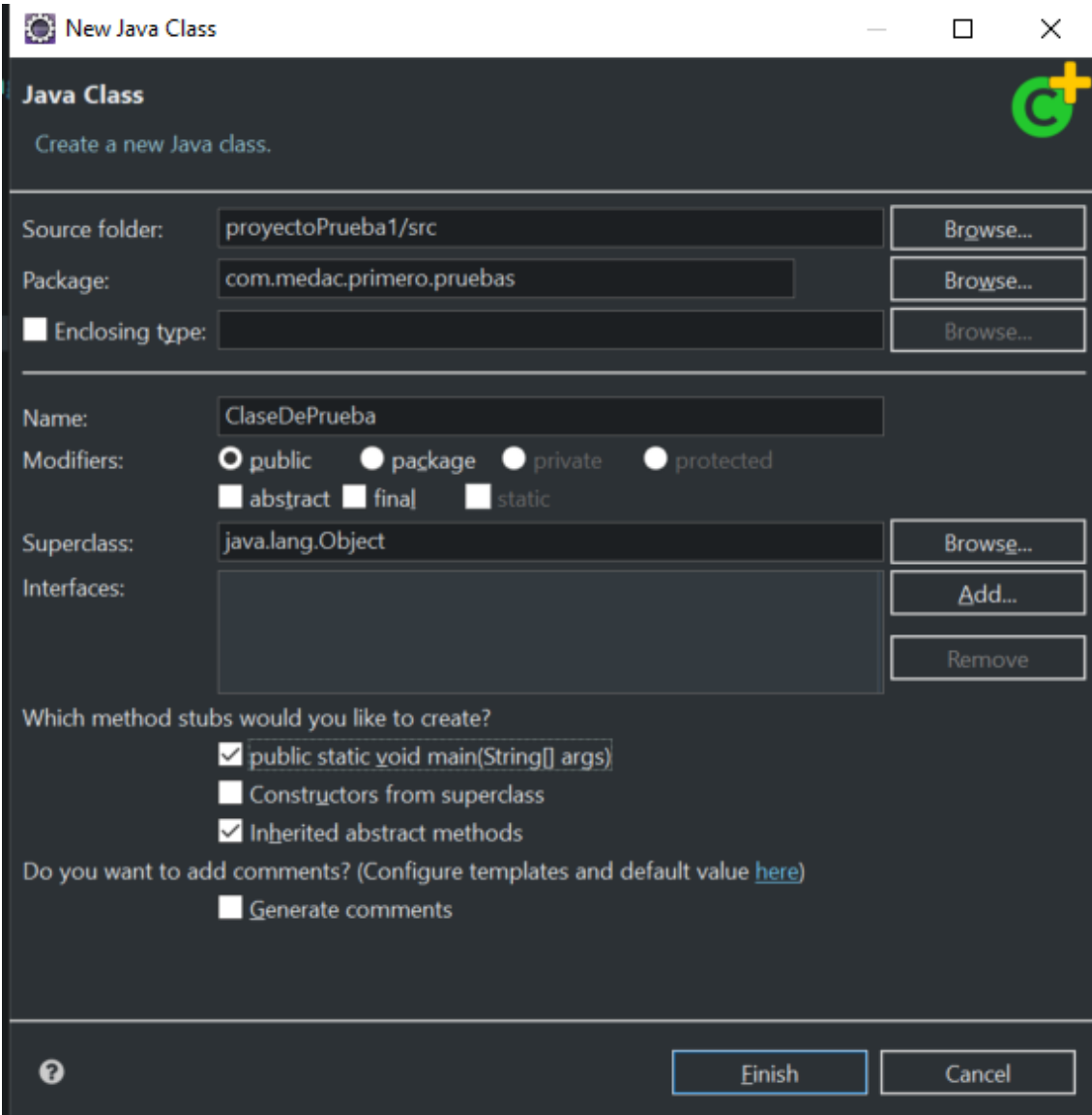
Las clases SIEMPRE deben empezar por una letra mayúscula, y las siguientes palabras deben seguir la nomenclatura camelCase

1º Click derecho en el paquete (package) donde queramos crear la clase

2º Click en “New” -> “Class”

3º Nombramos la clase como queramos. EN LAS CLASES SIEMPRE SE EMPIEZA POR MAYÚSCULA, y después, se sigue la nomenclatura camelCase.

4º (opcional) Marcamos la casilla “public static void main(String[] args)” -> Esto es para crear un método “main” o principal de forma automática.



New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ `public static void main(String[] args)`

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Crear directorio de paquetes en Java

Los paquetes en Java (packages) son la forma en la que Java nos permite agrupar de alguna manera lógica los componentes de nuestra aplicación que estén relacionados entre sí. Dicho con otras palabras, funcionan como las carpetas en tu ordenador.

Cuando creamos el directorio de paquetes, cada paquete (o carpeta) se dividirá por un punto. Es decir, si queremos crear un directorio de paquetes para nuestros diferentes módulos dentro del ciclo formativo del MEDAC, a la hora de crear un paquete, deberíamos nombrarlo así:

medac.programacion.actividad1

medac.programacion.actividad2

medac.programacion.actividad3

medac.programacion.pruebas

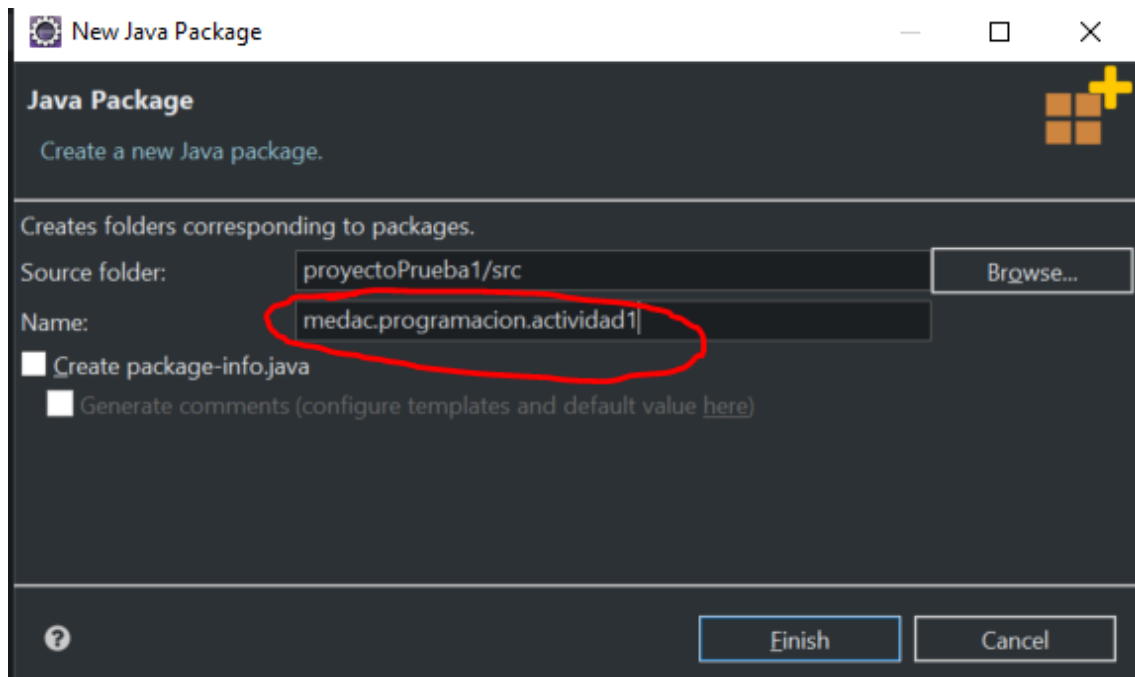
medac.entornos.practicaEntornos

medac.entornos.pruebas

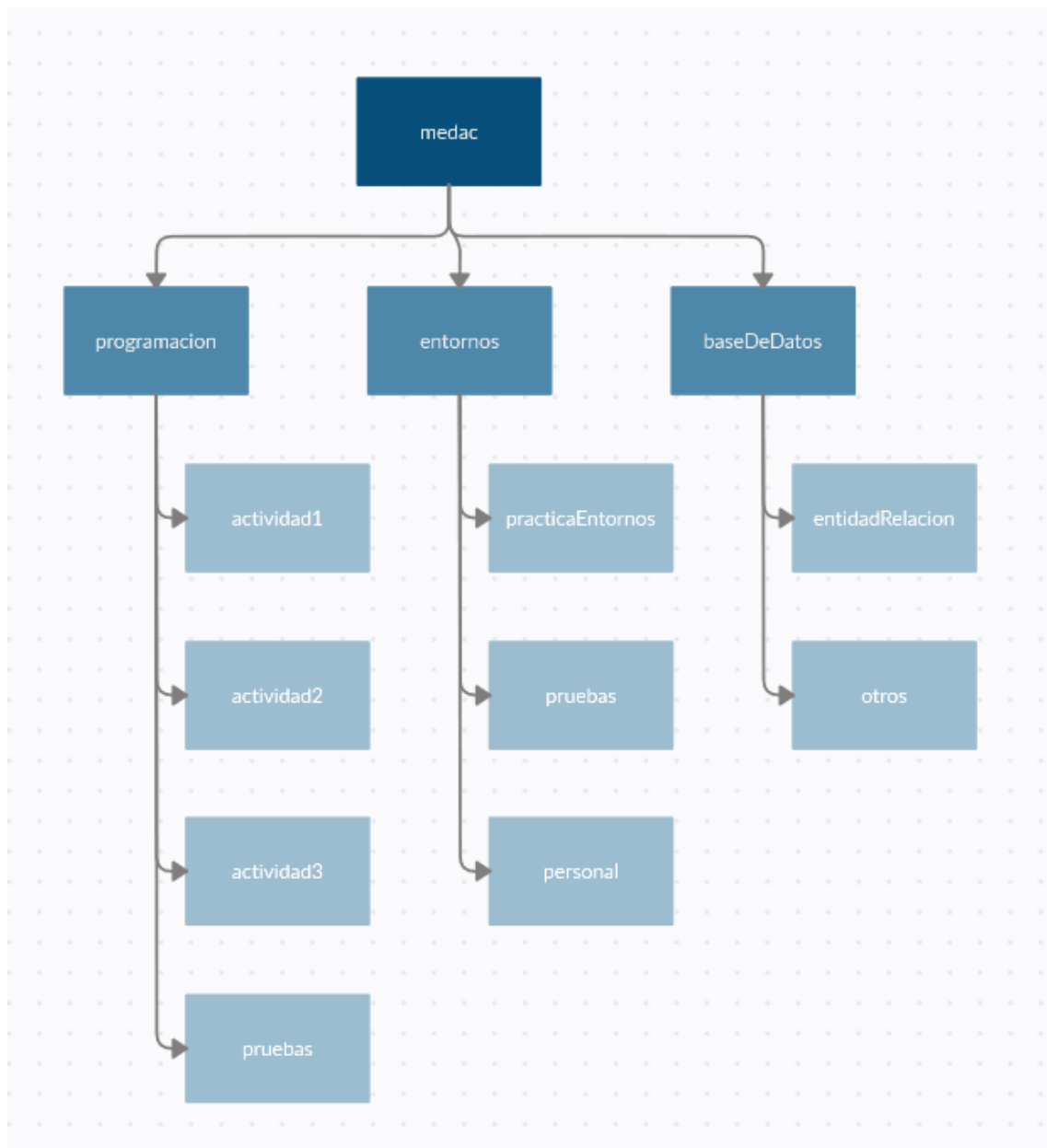
medac.entornos.personal

medac.baseDeDatos.entidadRelacion

medac.baseDeDatos.otros



Esto nos crearía una carpeta medac. Dentro de la carpeta medac habría 3 carpetas (programación, entornos y baseDeDatos). Dentro de cada una de las carpetas, habría otras carpetas que identificarían los contextos de trabajo que nosotros quisiéramos identificar o delimitar.



Nomenclatura camelCase

Es una práctica de escritura que consiste en la unión de dos o más palabras sin espacios entre ellas, pero las diferencian una letra mayúscula inicial a partir de la segunda palabra

Ejemplo variable: var**C**adena

Ejemplo variable: var**E**jemplo**C**amel

Ejemplo variable: cadena**L**arga

Ejemplo clase: Hola**M**undo**R**edondo

Ejemplo clase: Clase**D**e**P**ueba

Ejemplo genérico: Se**D**ebe**E**scribir**L**a**P**rimera**L**etra**E**n**M**ayuscula**E**mpezando**P**or**L**a**S**egunda

TIPOS DE DATOS

Datos de tipo numérico (números)

int -> son los números enteros (sin decimales)

double -> son los números reales (con decimales)

Ejemplos:

int i = 1;

int num = 12355;

int loQueSea = -12;

double d = 1.123;

double estoSoloIndicaElNombreDeLaVariable = -5.07;

```
int num = 1;
double d = -7.87;
```

otros datos de tipo numérico

short -> igual que los int pero ocupan menos espacio en memoria

byte -> igual que los int pero ocupan menos espacio en memoria

long -> igual que los int pero ocupan más espacio en memoria

float -> igual que los double pero ocupan menos espacio en memoria

```
short s = 123;
long misMuelas = 3854839248741;
```

Tipo	Bytes ocupados en memoria	Rango de valores
byte	1	$[-128, 127]$
short	2	$[-32768, 32767]$
int	4	$[-2^{31}, 2^{31}-1]$
long	8	$[-2^{63}, 2^{63}-1]$

Tipo	Bytes ocupados en memoria	Rango de valores	
		En los negativos	En los positivos
float	4	$[-3.4E^{38}, -1.4E^{45}]$	$[1.4E^{-45}, 3.4E^{48}]$
double	8	$[-1.8E^{308}, -4.9E^{324}]$	$[4.9E^{-324}, 1.48E^{308}]$

Datos de tipo alfanuméricos (letras, palabras, etc)

String -> representa varias letras o caracteres juntos, es decir, representa una cadena de caracteres. Las cadenas de caracteres se representan entre comillas dobles -> “ ”

Ejemplos:

String cadena = “Soy una cadena de caracteres”;

String cad = “YoTambién”;

String pepito = “IncluyoYo XD barra baja _ hola”;

String elNombreQueQuieras = “123”;

char -> representa UN ÚNICO CARÁCTER. Se representa con comillas SIMPLES -> ‘ ’

Ejemplos:

char caracter = ‘i’;

char c = ‘a’;

char aquiLePonesElNombreQueQuieras = ‘A’;

```
char caracter = 'i';
char conmigoQuienquieraContraMiQuienPueda = 'A';
String cadena = "HolaMundo";
String meInventoElNombreYSigueFuncionando = "hola";
```

Datos tipo booleano

boolean -> Indican verdadero o falso. ÚNICAMENTE PUEDEN SER VERDADERO (TRUE) O FALSO (FALSE).

```
boolean soyRico = false;
```

```
boolean soyPobre = true;
```

```
boolean nombreDeLaVariable = false;
```

```
boolean algo = true;  
boolean otraCosa = false;
```

OPERADORES

Operadores de asignación

Operador	Ejemplo	Equivalencia
=	Int a = b;	
+=	Int a += b;	Int a = a + b;
-=	Int a -= b;	Int a = a - b;
*=	Int a *= b;	Int a = a * b;
/=	Int a /= b;	Int a = a / b;
%=	Int a %= b;	Int a = a % b;

= -> Para asignar un valor a una variable.

+ -> Para sumar dos variables (una suma aritmética de toda la vida)

- -> Para restar dos variables

* -> Para multiplicar dos variables

/ -> Para dividir dos variables

% -> Para **obtener el resto de una división entre dos variables**

Ejemplo de %

Si queremos saber si un número es par o impar. *variable % 2* -> Si nos devuelve 0 (es par). Si nos devuelve diferente a 0 (es impar).

Operadores lógicos

Operador igual que, se representa como ==.

Operador distinto, se representa como !=.

Operador mayor que, se representa como >.

Operador menor que, se representa como <.

Operador mayor o igual que, se representa como >=.

Operador menor o igual que, se representa como <=.

== -> Sirve para comparar si 2 variables **son iguales**. CUIDADO CON LOS STRINGS, se comparan con un método (ver apartado "comparando Strings"). Si las dos variables son **iguales**, entonces nos devuelve **TRUE**. Si no son iguales, nos devuelve **FALSE**.

Sería una pregunta tal que así: "¿Es i igual a j? Si sí, hace una cosa. Si no (else), hace otra." O así: "¿Es adoptar igual a true? ¿Es adoptar true? ¿Es adoptar verdadero? Si sí, hace una cosa. Si no (else) hace otra".

Ejemplos:

```
int i = 5;
int j = 5;
if ( i == j ) {
    System.out.println("Los dos int son iguales");
} else {
    System.out.println("Los dos int son diferentes");
}
```

```
boolean adoptar = true;
if ( adoptar == true ) {
    System.out.println("Bien hecho");
} else {
    System.out.println("Adopta, no compres");
}
```

!= -> Sirve para comparar si dos variables **son diferentes**. CUIDADO CON LOS STRINGS, se comparan con un método (ver apartado "comparando Strings"). Si las dos variables son **diferentes**, entonces nos devuelve TRUE. Si son iguales, nos devuelve **FALSE**.

Sería una pregunta tal que así: *"¿Es k **diferente** a z? Si sí, hace una cosa. Si no (else), hace otra."*

Ejemplos:

```
int k = 2;
int z = 7;
if ( k != z ) {
    System.out.println("Los dos int son diferentes");
} else {
    System.out.println("Los dos int son iguales");
}

boolean adoptar = true;
if ( adoptar != true ) {
    System.out.println("Adopta, no compres");
} else {
    System.out.println("Bien hecho");
}
```

```
}
```

> -> Sirve para comprobar si una variable es mayor a otra. **ESTRICTAMENTE MAYOR**. Si la variable de la izquierda es mayor a la variable de la derecha **nos devuelve TRUE**. Si no, nos devuelve **FALSE**.

< -> Sirve para comprobar si una variable es menor a otra. **ESTRICTAMENTE MENOR**. Si la variable de la izquierda es menor a la variable de la derecha **nos devuelve TRUE**. Si no, nos devuelve **FALSE**.

Ejemplos:

```
int i = 1;
```

```
int j = 2;
```

```
int k = 3;
```

```
if ( i < j ) {
```

```
    System.out.println("i es mejor que j");
```

```
}
```

```
if ( k > j ){
```

```
    System.out.println("k es mayor que j");
```

```
}
```

>= -> Sirve para comprobar si una variable es mayor o igual a otra. **MAYOR O IGUAL**. Si la variable de la izquierda es mayor o igual a la variable de la derecha **nos devuelve TRUE**. Si no, nos devuelve **FALSE**.

<= -> Sirve para comprobar si una variable es menor o igual a otra. **MENOR O IGUAL**. Si la variable de la izquierda es menor o igual a la variable de la derecha **nos devuelve TRUE**. Si no, nos devuelve **FALSE**.

Ejemplos:

```
int i = 1;
```

```
int j = 3;
```

```
int k = 3;
```

```
if ( i <= j ) {
```

```
    System.out.println("i es mejor que j");
```

```
}
```

```
if ( k >= j ){
```

```
    System.out.println("k es igual que j");
```

}

Operador AND, representado como &.

Operador OR, representado como |.

Operador NOT, representado como !.

Operador AND

Se representa con un & . Significa que ambas condiciones se tienen que cumplir para que el resultado sea TRUE. Si cualquiera de las dos condiciones no se cumple, entonces el resultado sería FALSE. Por ejemplo, para aprobar la asignatura hay que asistir a clase y aprobar todo.

Dicho con otras palabras:

Si aprobarTodo es true & asistirAClase es true entonces “se aprueba la asignatura”.

Dicho con otras palabras:

boolean aprobarTodo = true;

boolean asistirAClase = true;

```
if ( aprobarTodo == true & asistirAClase == true ) {  
    System.out.println("Puedo pasar de curso");  
} else {  
    System.out.println("A repetir se ha dicho");  
}
```

Otro ejemplo **evaluando las condiciones a false (es decir, preguntando si las condiciones son false)**. Para tener buena salud hay que no comer bollería industrial y no consumir drogas duras.

Dicho con otras palabras:

Si **no** bolleriaIndustrial & **no** drogasDuras, entonces “vivirás una vida saludable”

Dicho con otras palabras:

```
if ( bolleriaIndustrial == false & drogasDuras == false ) {  
    System.out.println ( "Buena salud");  
} else {  
    System.out.println("Únicamente Keith Richards puede permitírselo");  
}
```

Operador OR

Se representa con un `|` . Significa que una u otra condición se tiene que cumplir para que el resultado sea TRUE. Es decir, **BASTA CON QUE SE CUMPLA UNA CONDICIÓN PARA QUE EL RESULTADO GLOBAL SEA TRUE**. Por ejemplo, para acceder a la universidad puedes ir por bachillerato o por FP.

Dicho con otras palabras:

Si bachillerato es true | fp es true entonces “puedes acceder a la universidad”.

Dicho con otras palabras:

```
boolean bachillerato = false;
```

```
boolean fp = true;
```

```
if ( bachillerato == true | fp == true ) {  
    System.out.println("Puedo ir a la uni");  
} else {  
    System.out.println("No puedo ir a la uni");  
}
```

Para que el ejemplo anterior se evaluara globalmente a false, ambas deberían ser **false**

Dicho con otras palabras:

```
boolean bachillerato = false;
```

```
boolean fp = false;
```

```
if ( bachillerato == true | fp == true ) {  
    System.out.println("Puedo ir a la uni");  
} else {  
    System.out.println("No puedo ir a la uni");  
}
```

COMPARANDO STRINGS

Comparar 2 Strings se hace de una manera diferente, puesto que los Strings no son primitivos, si no, clases.

Si quiero comparar dos Strings, lo haría con el método `equals()`; o con el método `equalsIgnoreCase()`;

`equals()` -> Compara dos cadenas y comprueba si son EXACTAMENTE IGUALES. Discriminando incluso las mayúsculas y minúsculas.

`equalsIgnoreCase()` -> Compara dos cadenas y comprueba si son iguales pero no discrimina las mayúsculas ni las minúsculas.

Ejemplo

```
String cadena1 = "Perro";
```

```
String cadena2 = "Perro";
```

```
String cadena3 = "perro";
```

```
if ( cadena1.equals(cadena2) ){  
    System.out.println("Son iguales");  
} else {  
    System.out.println("Son diferentes");  
}
```

```
if ( cadena1.equals(cadena3) ) {  
    System.out.println("Son iguales");  
}
```


COMENTARIOS EN JAVA

Comentarios en 1 línea

Para poner un comentario en una línea, se utiliza `//` . Todo lo que haya a la derecha de estas dos barras, estará comentado, es decir, no se ejecutará.

Ejemplos.

```
int num = 15; //Declaro una variable que se llama num y es de tipo entero. Le asigno un valor de 1.
```

```
int edad = 10;
```

```
double k; //Declaro una variable que se llama k y es de tipo double. No le asigno ningún valor.
```

```
boolean fp = true; //Declaro una variable fp de tipo booleano y la inicializo a true.
```

```
//Pequeño fragmento de código para intercambiar el valor de dos variables
```

```
int aux; //declaro una variable auxiliar de tipo int.
```

```
aux = edad; //le asigno a esta variable aux el valor de edad
```

```
edad = num; //Ahora edad vale lo que vale num
```

```
num = aux; //Ahora num vale lo que vale aux
```

```
boolean bach = false; //Creo una variable bach de tipo booleano. La inicializo a false  
boolean fp = true; //Creo una variable fp de tipo booleano. La inicializo a true  
if(bach | fp){ //Si bach o fp es true, entonces imprimo por pantalla la cadena de caracteres (String) "es true". Si no, imprimo "es false"  
    System.out.println("es true");  
}else {  
    System.out.println("es false");  
}
```

Comentarios multilínea

Para poner comentarios en varias líneas seguidas, utilizamos `/*` para indicar el comienzo de nuestro bloque de comentarios. Utilizamos `*/` para concluir nuestro bloque de comentarios.

Ejemplo

`/*` Voy a crear un método que lea por teclado una cadena de caracteres.

Después, mi método evaluará con un `if` si esa cadena de caracteres es “Diego” o no.

Si no lo es, imprimirá un mensaje que dirá “Fuera de aquí, intruso”.

Si sí, lo es, imprimirá un mensaje que dirá “Hola Diego”.

`*/`

```
Scanner scan = new Scanner(System.in);
```

```
System.out.println("Introduce la cadena");
```

```
String cad= scan.nextLine();
```

```
if(cad.equalsIgnoreCase("Diego")) {
```

```
    System.out.println("Hola Diego");
```

```
}else {
```

```
    System.out.println("Fuera de aquí, intruso");
```

```
}
```

```
scan.close();
```