

UNIDAD 2: USO DE ESTRUCTURAS DE CONTROL

Módulo Profesional: Programación

Índice

RESUMEN INTRODUCTORIO.....	3
INTRODUCCIÓN.....	3
CASO INTRODUCTORIO	4
1. INTRODUCCIÓN A LA PROGRAMACIÓN	5
1.1 Instalación software. Eclipse	5
1.2 Estructuras selectivas o alternativas.....	14
2. ESTRUCTURAS REPETITIVAS	22
RESUMEN FINAL	30

RESUMEN INTRODUCTORIO

A lo largo de esta unidad veremos el uso de las estructuras de control en el cuerpo de un programa, desde las estructuras selectivas o alternativas que se pueden usar hasta las estructuras repetitivas y las semejanzas y diferencias entre ellas. Para cada uno de los apartados veremos ejemplos en el lenguaje de programación Java.

INTRODUCCIÓN

El cuerpo de un programa se compone de un conjunto de sentencias que especifican las acciones que se realizan durante su ejecución. Dentro de cualquier programa se escriben sentencias que definen la secuencia de acciones a ejecutar. Estas sentencias incluyen acciones de cálculo, entrada y salida de datos, almacenamiento de datos, etc. Las sentencias se ejecutan una a una en el orden en el que han sido escritas.

El flujo de un programa es el orden en el que se ejecutan las sentencias que forman parte del cuerpo de un programa. Las estructuras de control son una característica básica de los lenguajes que se utilizan para modificar el flujo de un programa. Estas estructuras permiten realizar diferentes tareas con la información y son las que aportan diversas posibilidades en la programación.

Aunque nos parezca increíble las estructuras de control están presentes en nuestro día a día. ¿Cuántas veces no nos habrán dicho: si está lloviendo coge el paraguas? Y nunca nos hemos parado a pensar que se trata de una estructura de control en concreto una estructura selectiva simple (SI).

CASO INTRODUCTORIO

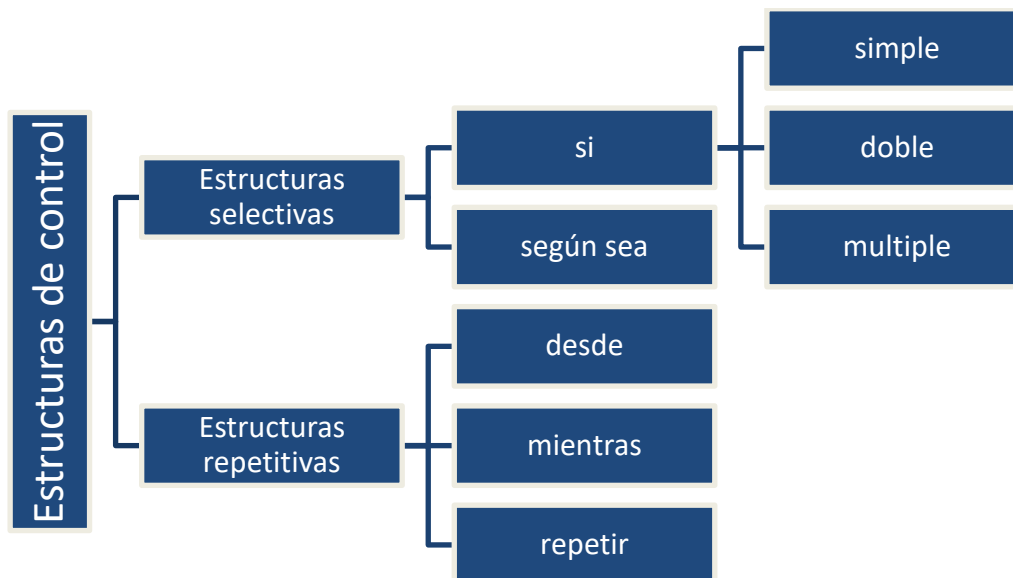
En la empresa en la que trabajas eres el responsable de realizar un proyecto de programación para un cliente. Para ello debes analizar cuáles son los equipos con los que trabajan e ir analizando uno a uno qué características y qué sistema operativo tiene. Esta tarea tenemos que realizarla de forma repetitiva y finalmente decidir si realizamos el programa en Java o C#.

Al finalizar la unidad el alumnado:

- ✓ Conocerá las distintas estructuras de control y para qué se utiliza cada una.
- ✓ Será capaz de aplicar las estructuras de control para mejorar el código de sus programas.

1. INTRODUCCIÓN A LA PROGRAMACIÓN

Se pueden encontrar dos tipos principales de estructuras de control:



Las **estructuras selectivas** son las que permiten ejecutar una serie de instrucciones u otra dependiendo del valor de una expresión booleana o del valor de una variable numérica, de carácter o de texto.

Las **estructuras repetitivas** son las que permiten ejecutar una serie de instrucciones un número de veces indeterminado que va a depender de una expresión booleana, o bien un número predeterminado de veces.

1.1 Instalación software. Eclipse

A la hora de comenzar a programar, se dispone de diversas herramientas que permiten realizar un desarrollo del software. Son conocidos como **IDE** o Entorno de Desarrollo Integrado.

El IDE o "Integrated Development Environment", en inglés, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.



ARTÍCULO DE INTERÉS

Para obtener más información acerca de los IDE's, se puede consultar la siguiente referencia:

https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

Actualmente existen multitud de entornos de desarrollo de software. Se va a detallar cómo realizar la instalación del entorno **Eclipse**.



ENLACE DE INTERÉS

Web oficial de Eclipse: <http://www.eclipse.org>



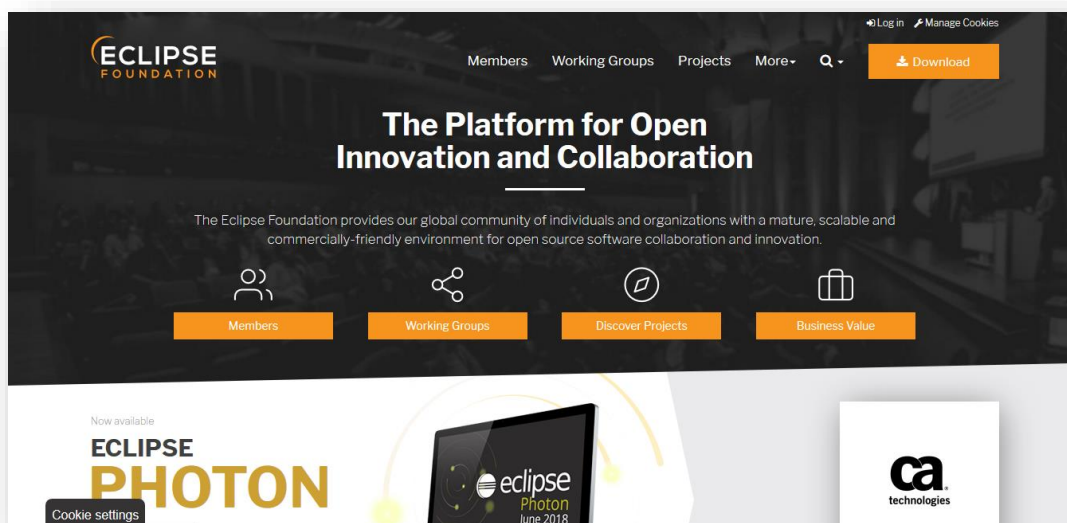
ARTÍCULO DE INTERÉS

A continuación, se indican diversas referencias en las que se muestran los mejores IDE's en el mercado para el desarrollo de software:

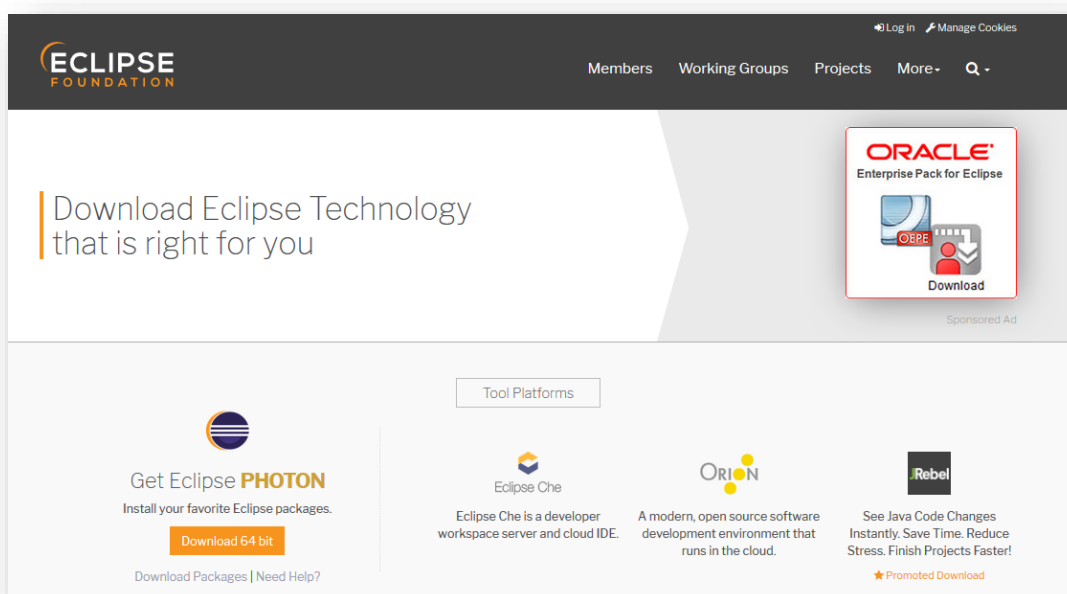
<https://www.diarlu.com/mejores-ide-programar-java/>

<https://www.redeszone.net/2017/03/24/conoce-estos-6-entornos-programacion-ide-programar-varios-lenguajes/>

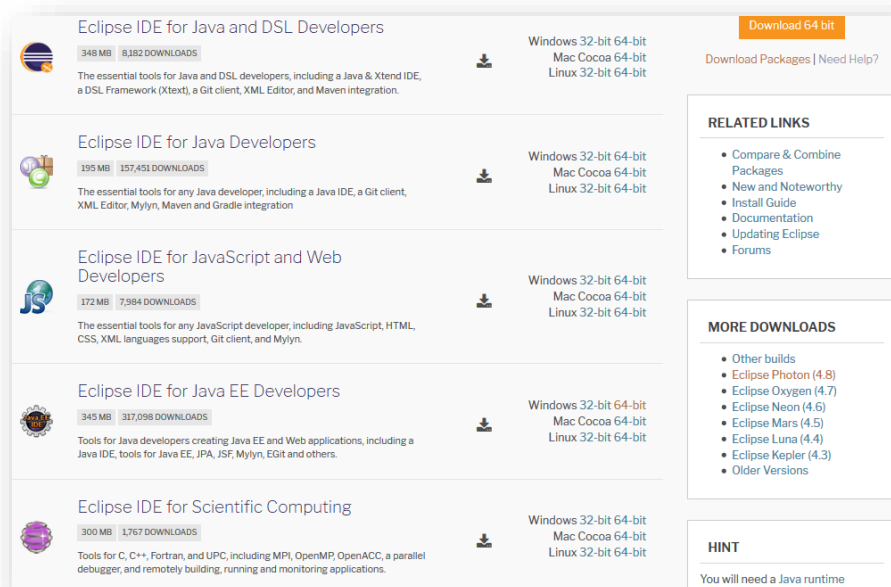
Para comenzar la instalación del IDE Eclipse, se procede a su descarga a través de la página oficial.



Se pulsa sobre el botón Download que aparece en la esquina superior derecha de la pantalla.

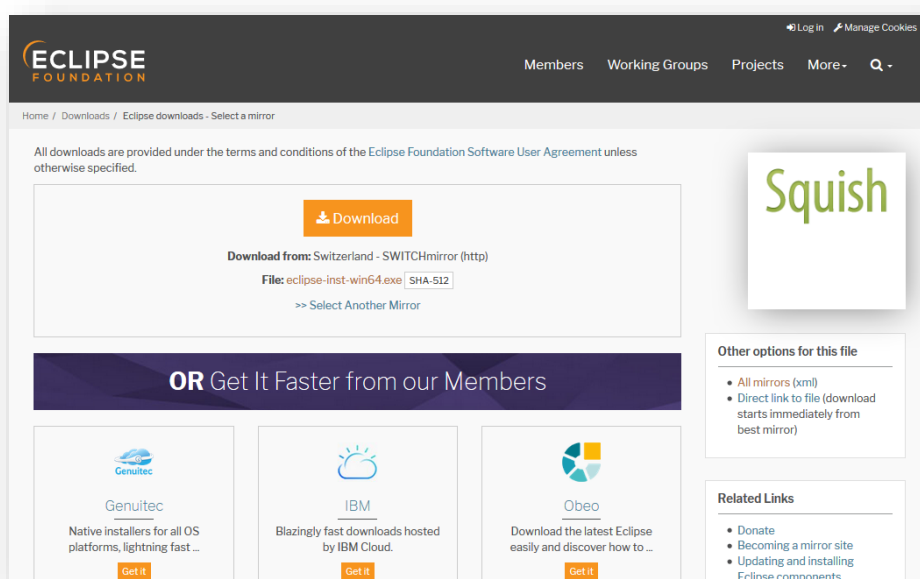


A continuación, se pulsa sobre el botón Download 64 bit que aparece en la esquina inferior izquierda. Si se realiza la descarga para un ordenador de 32 bits, o se está utilizando otro sistema operativo (Linux, MacOS), se podrá descargar la versión exacta pulsando en el enlace Download Packages. Se seleccionará la versión oportuna de Eclipse IDE for Java Developers, o bien Eclipse IDE for Java EE Developers.



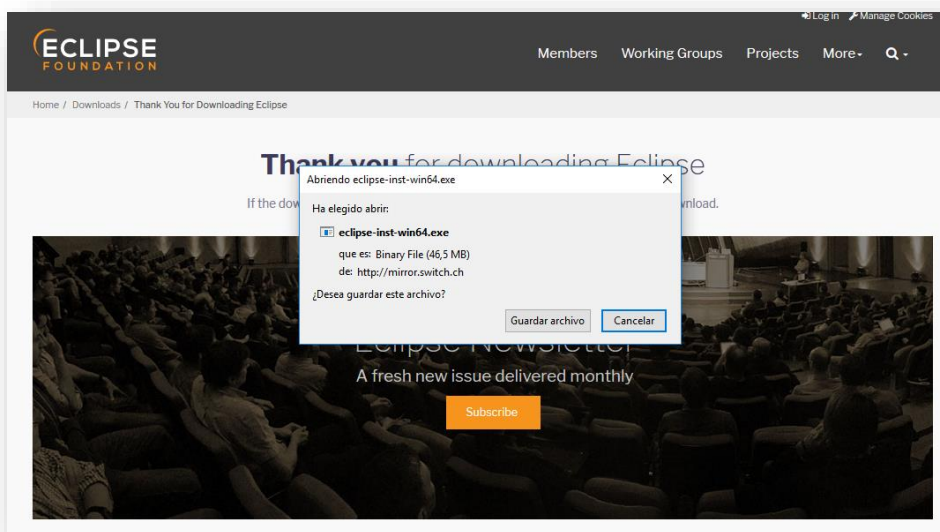
The screenshot shows the Eclipse IDE download page. It lists five different IDEs for download, each with a description, size, download count, and supported operating systems. The first IDE, 'Eclipse IDE for Java and DSL Developers', is highlighted with an orange 'Download 64 bit' button. To the right, there are sections for 'RELATED LINKS' (including Compare & Combine Packages, New and Noteworthy, Install Guide, Documentation, Updating Eclipse, and Forums) and 'MORE DOWNLOADS' (including Other builds, Eclipse Photon (4.8), Eclipse Oxygen (4.7), Eclipse Neon (4.6), Eclipse Mars (4.5), Eclipse Luna (4.4), Eclipse Kepler (4.3), and Older Versions). A 'HINT' section at the bottom states: 'You will need a Java runtime (JRE 8 or later)'.

En este caso, se va a realizar la descarga del archivo de instalación para Windows de 64 bits.

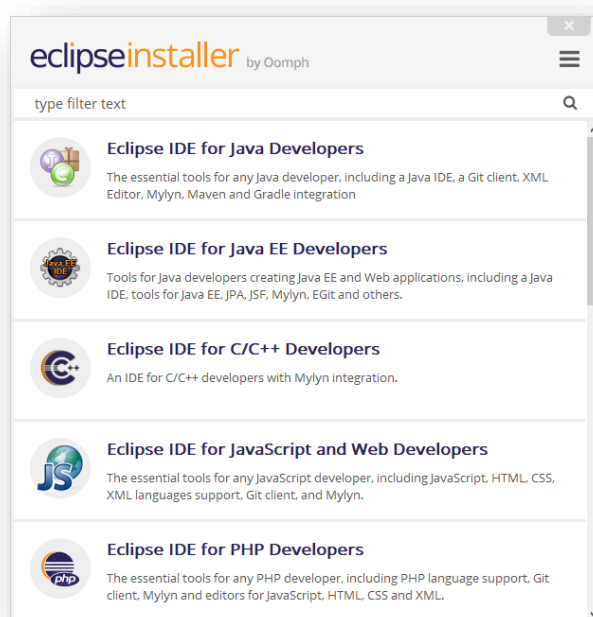


The screenshot shows the Eclipse Foundation download page. It features a 'Download' button and a 'Download from: Switzerland - SWITCHmirror (http)' link. Below this, it says 'File: eclipse-inst-win64.exe SHA-512' and '>> Select Another Mirror'. A section titled 'OR Get It Faster from our Members' lists three options: Genuitec (Native installers for all OS platforms, lightning fast ...), IBM (Blazingly fast downloads hosted by IBM Cloud), and Obeo (Download the latest Eclipse easily and discover how to ...). To the right, there is a 'Squish' logo and a section titled 'Other options for this file' (including All mirrors (xml), Direct link to file (download starts immediately from best mirror)). A 'Related Links' section at the bottom includes links for Donate, Becoming a mirror site, and Updating and installing Eclipse components.

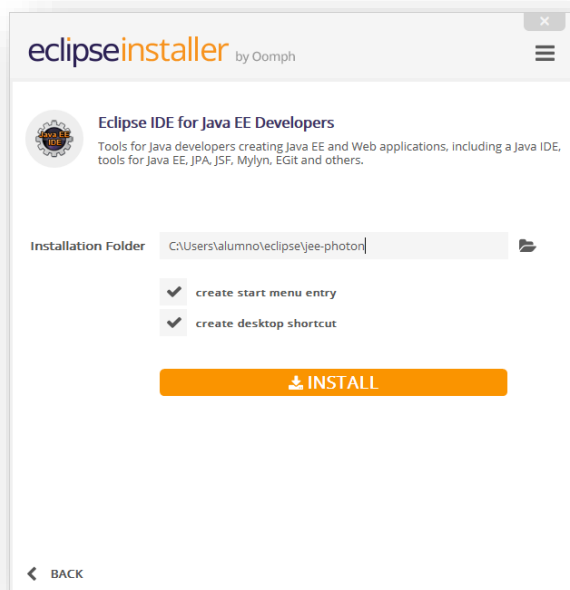
De nuevo se pulsa sobre el botón Download para obtener el archivo eclipse-inst-win64.exe de instalación del IDE Eclipse.



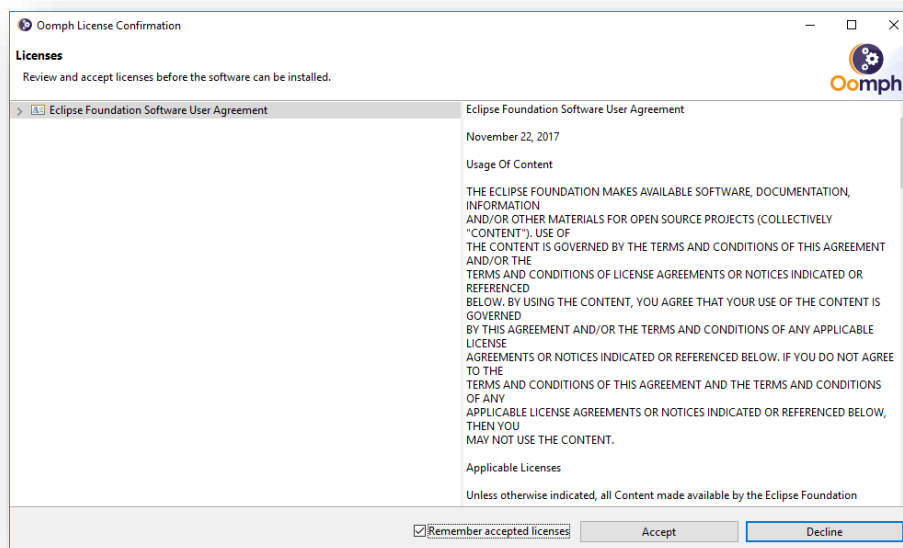
Se pulsará en Guardar archivo. Una vez descargado, se ejecutará el archivo para realizar la instalación.

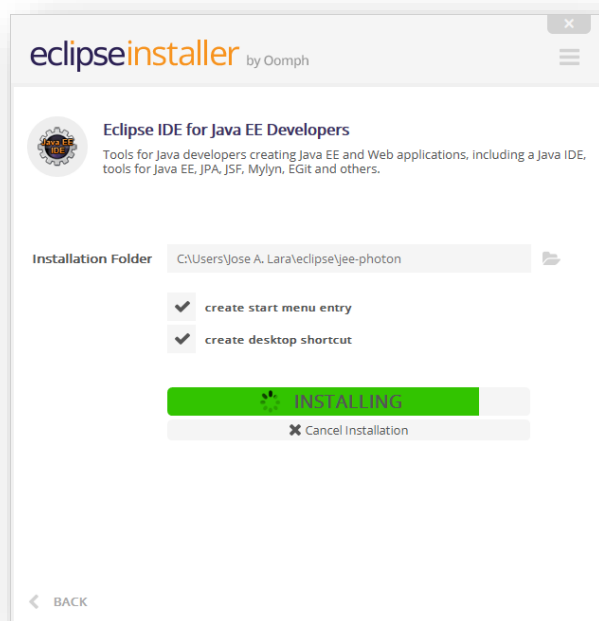


Se elige la versión que se desea instalar. El instalador pregunta la carpeta en la que se instalará Eclipse, y, finalmente, se pulsa en INSTALL.

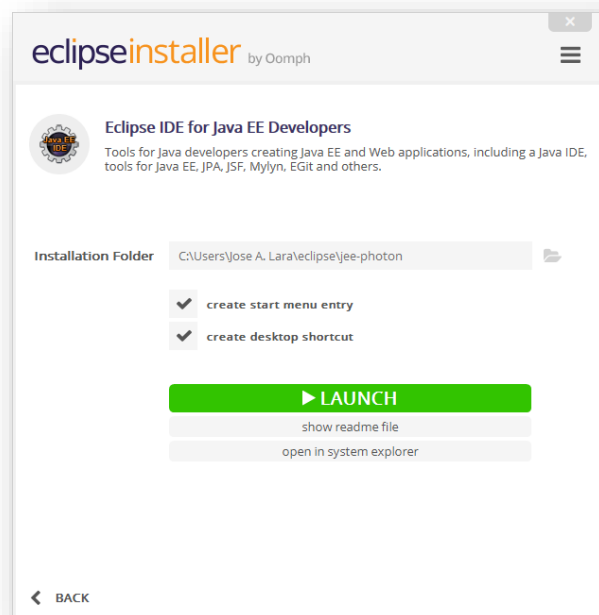


Se pedirá la aceptación de algunas licencias de software. Se marcará la opción Remember accepted licenses para no tener que volver a pulsar el botón Accept en cada una de las licencias.

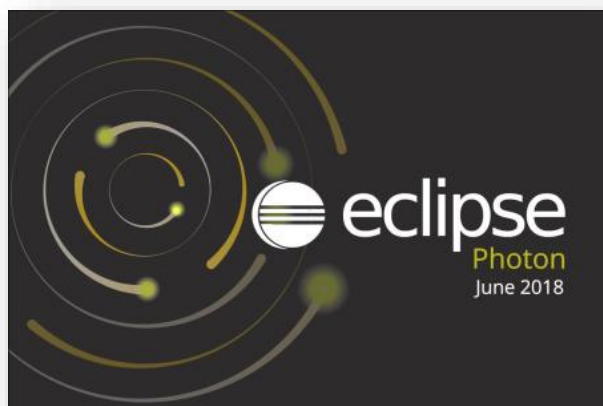




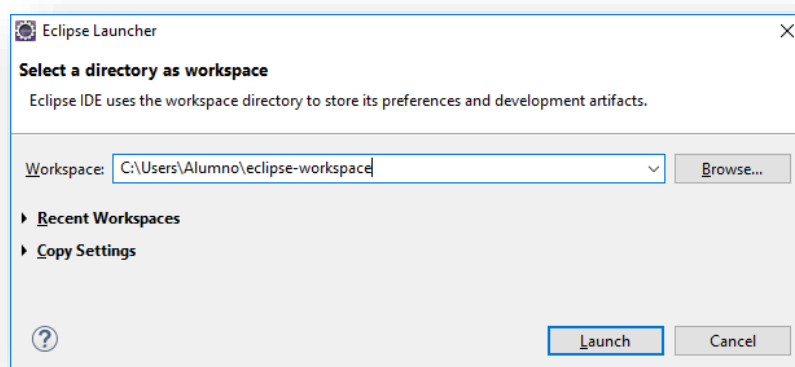
Continuará la instalación y finalmente mostrará la opción de ejecutar Eclipse. Para ello se pulsará en el botón Launch.



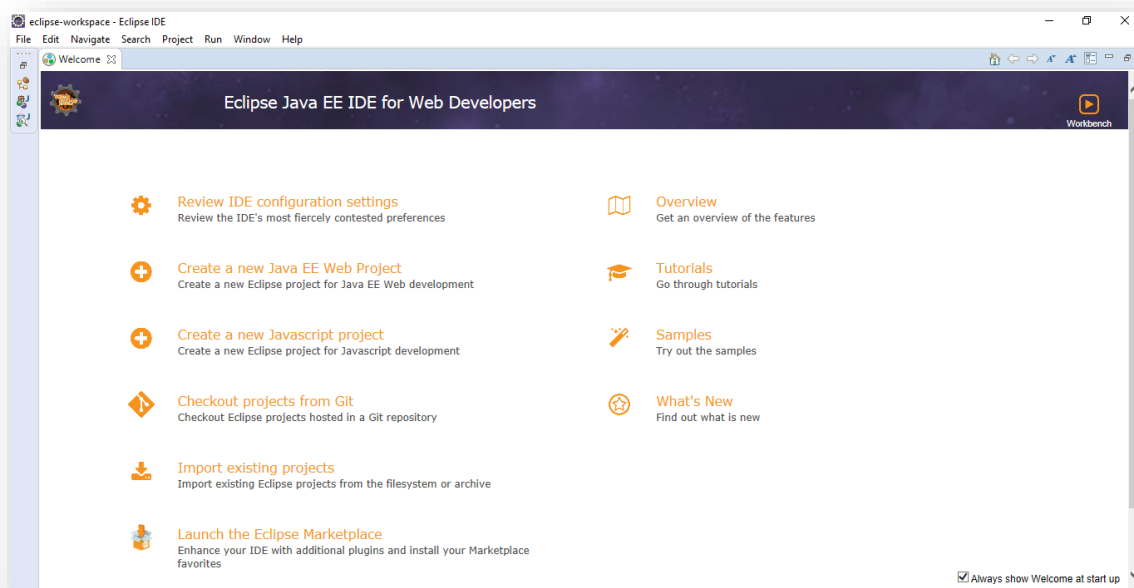
A continuación, se mostrará la llamada splash screen que especifica la versión de Eclipse que se va a ejecutar.



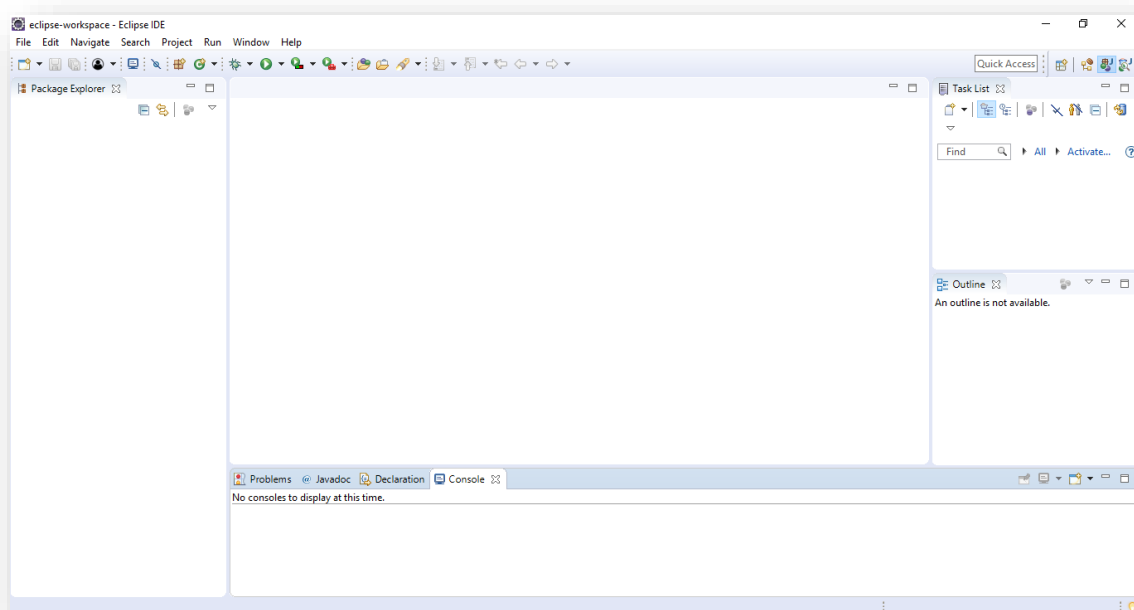
Solicitará que se le indique la carpeta en la cual se van a colocar los proyectos de Eclipse.



La primera vez que se ejecute Eclipse mostrará una ventana de bienvenida, en la cual se ofrecen varias opciones como revisar la configuración del IDE, crear nuevos proyectos, ver ejemplos y tutoriales, etc.

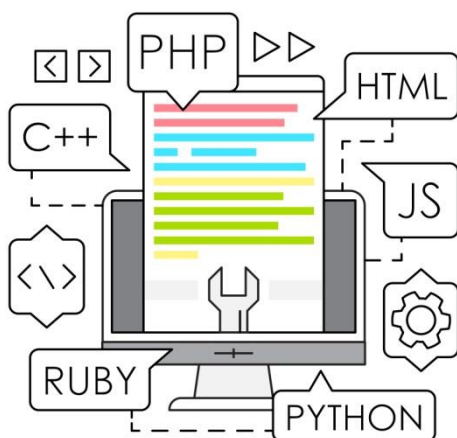


Se cierra la ventana de bienvenida, y ya se puede empezar a utilizar el entorno de desarrollo.



1.2 Estructuras selectivas o alternativas

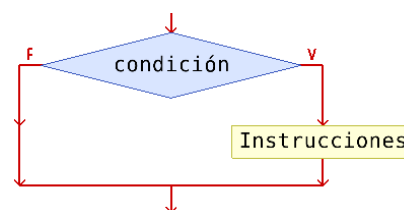
Son aquellas que permiten la **ejecución de unas u otras acciones** dependiendo de que se cumpla o no una condición o dependiendo del valor que tome una determinada variable



1. Si: permite ejercer una u otra acción dependiendo de que se cumpla o no una condición.

a. Simple

```
Si condición Entonces
    instrucciones;
Fin Si
```



ESTRUCTURA if

La estructura if se denomina estructura de selección única porque ejecuta un bloque de sentencias solamente cuando se cumple la condición del if.

- Si la condición es verdadera se ejecuta el bloque de sentencias.
- Si la condición es falsa, el flujo del programa continúa en la sentencia inmediatamente posterior al if.

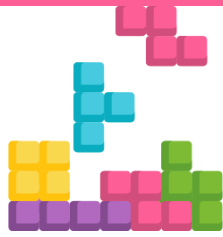
La sentencia if en Java tiene la siguiente sintaxis:

```
if (condición) {
    bloque-de-sentencias
}
```

La condición es una expresión que evalúa un valor lógico, por lo que el resultado solo puede ser true (verdadero) o false (falso). La condición siempre se escribe entre paréntesis. La selección se produce sobre el bloque de sentencias delimitado por llaves.

Si el bloque de sentencias solo tiene una sentencia, entonces se puede escribir sin las llaves, tal y como se muestra a continuación.

```
if (condición)
    sentencia;
```



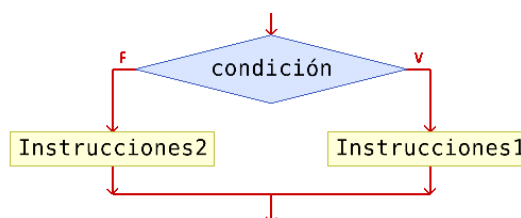
EJEMPLO PRÁCTICO

En el ejemplo, se comprueba si la edad es mayor o igual a 18 años. Si se cumple la condición (true), imprime por pantalla Es mayor de edad y, además, establece el valor de la variable booleana mayorEdad a true.

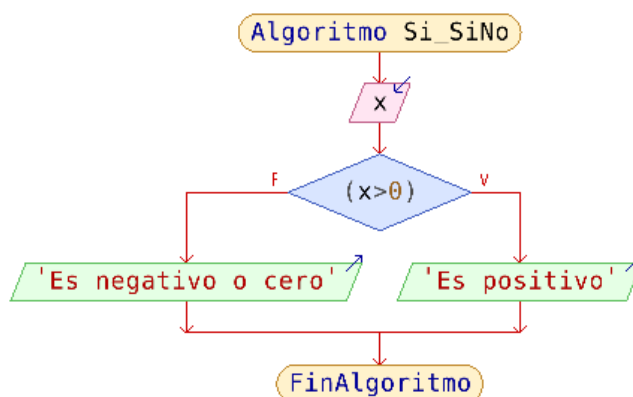
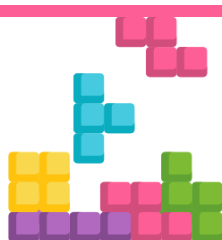
```
if (edad >= 18) {
    System.out.println("Es mayor de edad");
    mayorEdad = true;
}
```

b. Doble

Si *condición* Entonces
Instrucciones1;
 Si no
Instrucciones2;
 Fin Si



EJEMPLO PRÁCTICO



```

Algoritmo Si_SiNo
  Leer x
  Si (x>0) Entonces
    ..... Escribir 'Es positivo'
  SiNo
    ..... Escribir 'Es negativo o cero'
  Fin Si
FinAlgoritmo
  
```

Estructura if else

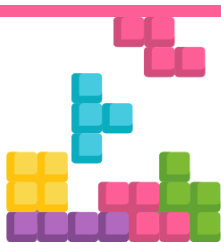
La estructura if-else se denomina de selección doble porque selecciona entre dos bloques de sentencias mutuamente excluyentes.

- Si se cumple la condición, se ejecuta el bloque de sentencias asociado al if.
- Si la condición no se cumple, entonces se ejecuta el bloque de sentencias asociado al else.

La sentencia if-else tiene la siguiente sintaxis:

```

if (condición) {
    bloque-de-sentencias-if
} else {
    bloque-de-sentencias-else
}
  
```

EJEMPLO PRÁCTICO

En el ejemplo, se comprueba si la edad es mayor o igual a 18 años. Si se cumple la condición (true), imprime por pantalla Es mayor de edad y, además, establece el valor de la variable booleana mayorEdad a true. Si no se cumple la condición (false), imprimirá Es menor de edad y, además, establece el valor de la variable booleana mayorEdad a false.

```
if (edad >= 18) {
    System.out.println("Es mayor de edad");
    mayorEdad = true;
} else {
    System.out.println("Es menor de edad");
    mayorEdad = false;
}
```

c. Múltiple

Si *condición1* Entonces

Instrucciones1;

Si no si *condición2* Entonces

Instrucciones2;

Si no si *condición3* Entonces

Instrucciones3;

Si no

Instrucciones4;

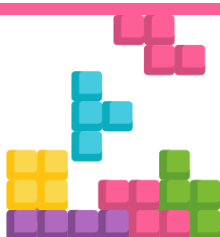
Fin Si

Estructura if else if

La estructura if-else-if se puede aplicar en los mismos casos en que se utiliza un if-else anidado. Esta estructura permite escribir de forma abreviada las condiciones de un if-else anidado.

Una sentencia if-else-if tiene la siguiente sintaxis:

```
if (condición-1) {
    bloque-de-sentencias-condición-1
} else if (condición-2) {
    bloque-de-sentencias-condición-2
} else {
    bloque-de-sentencias-else
}
```



EJEMPLO PRÁCTICO

En el ejemplo, se comprueba si un número es positivo, cero o negativo

```
if (numero>0) {
    System.out.println("Es positivo");
    negativo=false;
} else if (numero==0) {
    System.out.println("Es cero");
    negativo=false;
} else {
    System.out.println("Es negativo");
    negativo=true;
}
```

2. Según sea: se usa cuando dependiendo del valor de una variable queramos ejecutar una u otras acciones.

Según *variable_numerica* Hacer

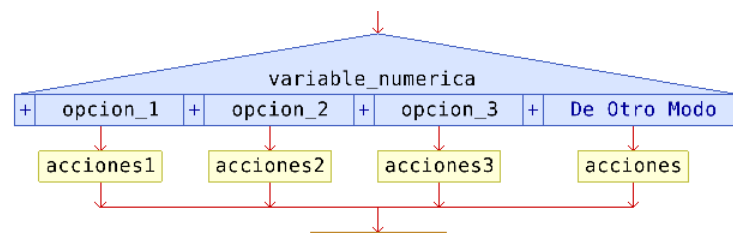
opcion_1:
 secuencia_de_acciones_1

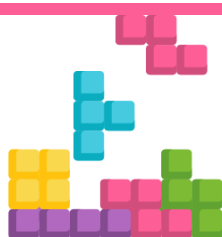
opcion_2:
 secuencia_de_acciones_2

opcion_3:
 secuencia_de_acciones_3

De Otro Modo:
 secuencia_de_acciones

Fin Según

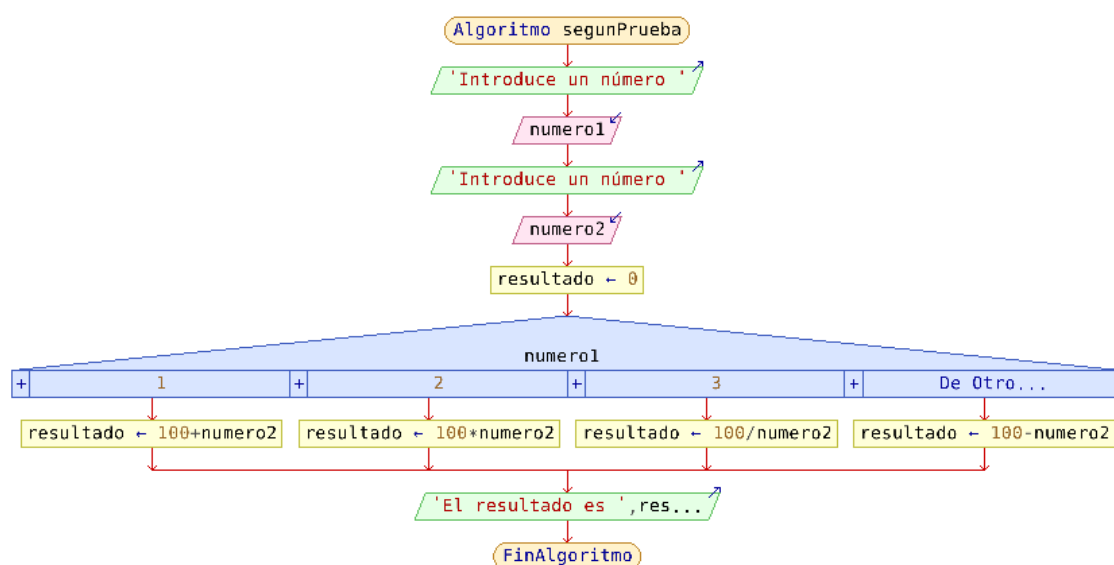




EJEMPLO PRÁCTICO

Diseñar un algoritmo tal que, dados como datos dos variables de tipo entero, obtenga el resultado de la siguiente función:

$$\text{Resultado} = \begin{cases} 100 + \text{numero2} & \text{si } \text{numero1} = 1 \\ 100 * \text{numero2} & \text{si } \text{numero1} = 2 \\ 100 / \text{numero2} & \text{si } \text{numero1} = 3 \\ 100 - \text{numero2} & \text{para cualquier otro valor} \end{cases}$$



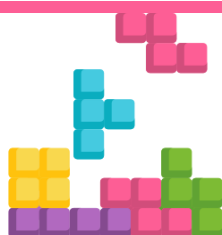
Estructura switch

La estructura switch es una estructura de selección múltiple que permite seleccionar un bloque de sentencias entre varios casos. En cierto modo, es parecido a una estructura de if-else anidados. La diferencia está en que la selección del bloque de sentencias depende de la evaluación de una expresión que se compara por igualdad con cada uno de los casos.

La estructura switch consta de una expresión y una serie de etiquetas case y una sola opción default. La sentencia break indica el final de la ejecución del switch.

Una sentencia switch tiene la siguiente sintaxis:

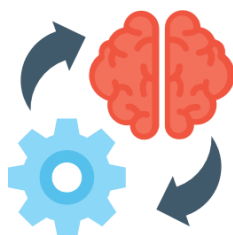
```
switch (expresión) {  
    case valor1:  
        bloque-de-sentencias-1;  
        break;  
    case valor2:  
        bloque-de-sentencias-2;  
        break;  
    case valor3:  
        bloque-de-sentencias-3;  
        break;  
    case valor4:  
        bloque-de-sentencias-4;  
        break;  
    default:  
        bloque-de-sentencias-default;  
}
```



EJEMPLO PRÁCTICO

Ejemplo de estructura switch en Java:

```
int numero=1;  
String cadena;  
switch (numero) {  
    case 0: cadena="Es cero";  
        break;  
    case 1: cadena="Es uno";  
        break;  
    case 2: cadena="Es dos";  
        break;  
    default: cadena="Es distinto de 0, 1 ó 2";  
}  
System.out.println(cadena);
```



RECUERDA

Las condiciones que controlan las estructuras selectivas devolverán siempre un valor booleano: verdadero o falso.



ARTÍCULO DE INTERÉS

Puede ampliar información sobre las estructuras de control selectivas y su sintaxis visitando la web:

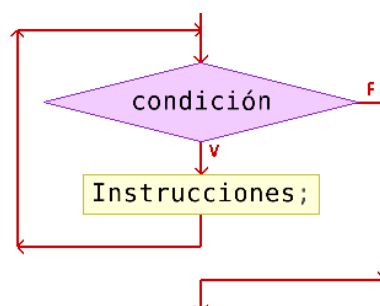
<http://www.desarrolloweb.com/articulos/2225.php>

2. ESTRUCTURAS REPETITIVAS

Estas estructuras permiten repetir un **número determinado** de instrucciones un **número finito** de veces.

- a) **Mientras:** se usa siempre que se quiera repetir una serie de instrucciones mientras se cumpla una determinada condición.

Mientras *condición* Hacer
 Instrucciones;
Fin Mientras



Número de iteraciones: de 0 a n

Estructura while

La estructura de repetición while repite el bloque de sentencias mientras la condición del while es verdadera. La condición se

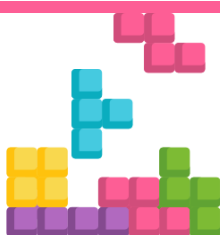
comprueba justo al principio. Si el resultado es **falso**, entonces **no se ejecuta** el bloque de sentencias. Por eso se dice que se puede ejecutar el bloque de sentencias de 0 (ninguna) a n (muchas) veces.

La sintaxis de while en Java es la siguiente:

```
while (condición) {  
    bloque-de-sentencias;  
}
```

La condición del while se escribe obligatoriamente entre paréntesis. Cuando el programa ejecuta un while, lo primero que hace es evaluar la condición. Si es verdadera ejecuta el bloque de sentencias, si es falsa finaliza el while.

En cada iteración, cuando finaliza la ejecución del bloque de sentencias, vuelve a evaluar la condición. De nuevo, si es verdadera, ejecuta el bloque de sentencias, si es falsa finaliza el while. Cuando esto último se produce, el flujo del programa continúa en la sentencia inmediatamente posterior al while.



EJEMPLO PRÁCTICO

Ejemplo de estructura while en Java. Este programa muestra la tabla de multiplicar del número introducido por teclado.

```
Scanner sc = new Scanner(System.in);  
int numero, contador, resultado;  
System.out.print("Introduzca un número: ");  
numero = sc.nextInt();  
contador=1;  
while (contador<=10) {  
    resultado=contador*numero;  
    System.out.println(numero+" * "+contador+" = "  
                        +resultado);  
    contador=contador+1;  
}
```

Resultado obtenido al introducir el número 4 por teclado:

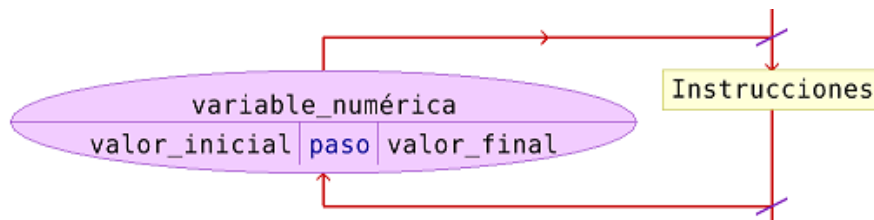
```
4 * 1 = 4  
4 * 2 = 8  
4 * 3 = 12  
4 * 4 = 16  
4 * 5 = 20  
4 * 6 = 24  
4 * 7 = 28  
4 * 8 = 32  
4 * 9 = 36  
4 * 10 = 40
```

- b) **Para**: se usa cuando se conozca de antemano el número de veces que se quiere repetir algo.

Para *variable_numerica* < -*valor_inicial* Hasta *valor_final*
 Con Paso *paso* Hacer

bloque-de-sentencias

Fin Para

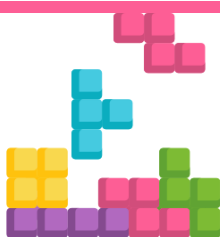


Estructura for

La estructura de repetición for repite el bloque de sentencias mientras la condición del for es verdadera. Un for es un caso particular de la estructura while. Sólo se debe utilizar cuando se sabe el número de veces que se debe repetir el bloque de sentencias.

Un for tiene la siguiente sintaxis:

```
for (inicialización; condición; actualización) {
    bloque-de-sentencias;
}
```



EJEMPLO PRÁCTICO

Ejemplo de estructura for en Java. Este programa muestra la tabla de multiplicar del número introducido por teclado.

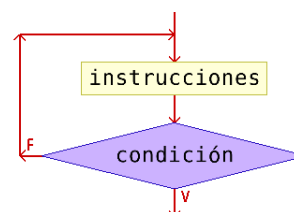
```
Scanner sc = new Scanner(System.in);
int numero, resultado;
System.out.print("Introduzca un número: ");
numero = sc.nextInt();
for (int contador = 1; contador <= 10; contador++) {
    resultado=contador*numero;
    System.out.println(numero+" * "+contador+" = "+resultado);
}
```

Resultado obtenido al introducir el número 4 por teclado:

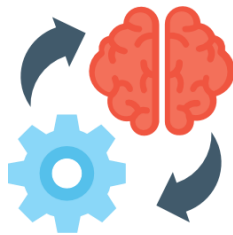
```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

c) **Repetir:** se usa siempre que se quiera repetir una serie de instrucciones hasta que se cumpla una determinada condición.

Repetir
instrucciones
Hasta Que condición



El cuerpo del bucle como mínimo se repite 1 vez. Número de iteraciones de 1 a n.

**RECUERDA**

Es muy importante a la hora de programar el orden y claridad del código. Sobre todo, cuando se trabaja con varios desarrolladores colaborando en el mismo equipo de programación.

**ARTÍCULO DE INTERÉS**

Puede ampliar información sobre las estructuras de control repetitivas y su sintaxis visitando la web:

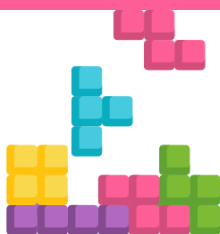
<http://www.desarrolloweb.com/articulos/2249.php>

La estructura de repetición do-while ejecuta el bloque de sentencias al menos una vez. Después comprueba la condición y repite el bloque de sentencias mientras la condición es verdadera.

Un do-while tiene la siguiente sintaxis:

```
inicialización;  
do {  
    bloque-de-sentencias;  
    actualización;  
} while (condición);
```

Esta es la sintaxis general. La condición se escribe obligatoriamente entre paréntesis.



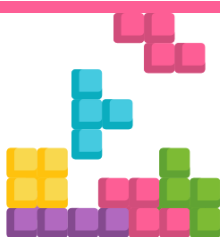
EJEMPLO PRÁCTICO

Ejemplo de estructura do-while en Java. Este programa muestra la tabla de multiplicar del número introducido por teclado.

```
Scanner sc = new Scanner(System.in);  
int numero, resultado, contador;  
System.out.print("Introduzca un número: ");  
numero = sc.nextInt();  
contador=1;  
do {  
    resultado=contador*numero;  
    System.out.println(numero+" * "+contador+" = "+resultado);  
    contador=contador+1;  
} while (contador<=10);
```

Resultado obtenido al introducir el número 4 por teclado:

```
4 * 1 = 4  
4 * 2 = 8  
4 * 3 = 12  
4 * 4 = 16  
4 * 5 = 20  
4 * 6 = 24  
4 * 7 = 28  
4 * 8 = 32  
4 * 9 = 36  
4 * 10 = 40
```



EJEMPLO PRÁCTICO

Algoritmo que muestra en pantalla los números del 1 al 50, realizado con bucle mientras, para y repetir.

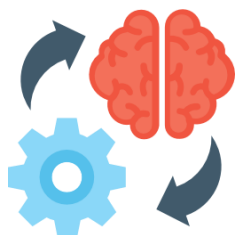
Algoritmo bucles

```
x<-1
Mientras x<=50 Hacer
    Escribir x
    x<-x+1
Fin Mientras
```

```
Para x<-1 Hasta 50 Con Paso 1 Hacer
    Escribir x
Fin Para
```

```
x<-1
Repetir
    Escribir x
    x<-x+1
Hasta Que x>50
```

FinAlgoritmo



RECUERDA

Algunas de las estructuras estudiadas son equivalentes entre sí y un mismo problema se puede resolver utilizando varias de ellas.

RESUMEN FINAL

En esta unidad se han estudiado las distintas estructuras de control y cuándo se utilizan. Además, a través de ejemplos prácticos en Java, se ha aprendido su sintaxis.