

UD01.Conceptos básicos

Apuntes

DAM1-Programación 2023-24

Introducción	2
Qué es la programación	2
Lenguajes de Programación	3
Instalar IDEs	4
Conceptos Básicos	5
“Hola Mundo”. Ejemplos en distintos lenguajes.	6
Componentes básicos de un lenguaje de programación	7
Programación en Java	8
El programa principal	8
Identificadores y Palabras reservadas	9
Tipos de datos y Variables	11
Literales y Constantes	12
Comentarios	13
API de Java	15
Operaciones básicas	18
Conversión de Tipos	18
Ejercicios	20
Anexo. Buenas Prácticas	22
Anexo. Instalar y configurar Git y GitHub	23
Crear y Configurar un repositorio remoto	23

Introducción

Qué es la programación

Las **computadoras** son una parte esencial de nuestra vida cotidiana. Casi todos los aparatos que usamos tienen algún tipo de computadora capaz de ejecutar ciertas tareas: lavarropas con distintos modos de lavado, consolas de juegos para momentos de entretenimiento, calculadoras súper potentes, computadoras personales que se usan para un montón de propósitos, teléfonos celulares con un sinfín de aplicaciones y miles de cosas más.

Todos estos dispositivos con computadoras de distinto tipo tienen algo en común: alguien “les dice” cómo funcionar, es decir, les indica cuáles son los pasos que deben seguir para cumplir una tarea. De eso se trata la **programación: es la actividad mediante la cual las personas le entregan a una computadora un conjunto de instrucciones para que, al ejecutarlas, ésta pueda resolver un problema**. Quienes realizan esta actividad reciben el nombre de **programadores**. Sin las personas que las programen, las computadoras dejan de ser útiles, por más complejos que sean estos aparatos. Los conjuntos de **instrucciones** que reciben las computadoras reciben el nombre de **programas**.

La programación es un proceso creativo: en muchas ocasiones la tarea en cuestión puede cumplirse siguiendo distintos caminos y el programador es el que debe imaginar cuáles son y elegir uno. Algunos de estos caminos pueden ser mejores que otros, pero en cualquier caso la computadora se limitará a seguir las instrucciones ideadas por el programador.

Desafortunadamente, las computadoras no entienden español ni otro idioma humano. Hay que pasarles las instrucciones en un lenguaje que sean capaces de entender. Para eso debemos aprender algún **lenguaje de programación**, que no es más que un lenguaje artificial compuesto por una serie de expresiones que la computadora puede interpretar. Las computadoras interpretan nuestras instrucciones de forma muy literal, por lo tanto a la hora de programar hay que ser muy específicos. Es necesario respetar las reglas del lenguaje de programación y ser claros en las indicaciones provistas.

Fuente: [1 Introducción a la Programación](#)

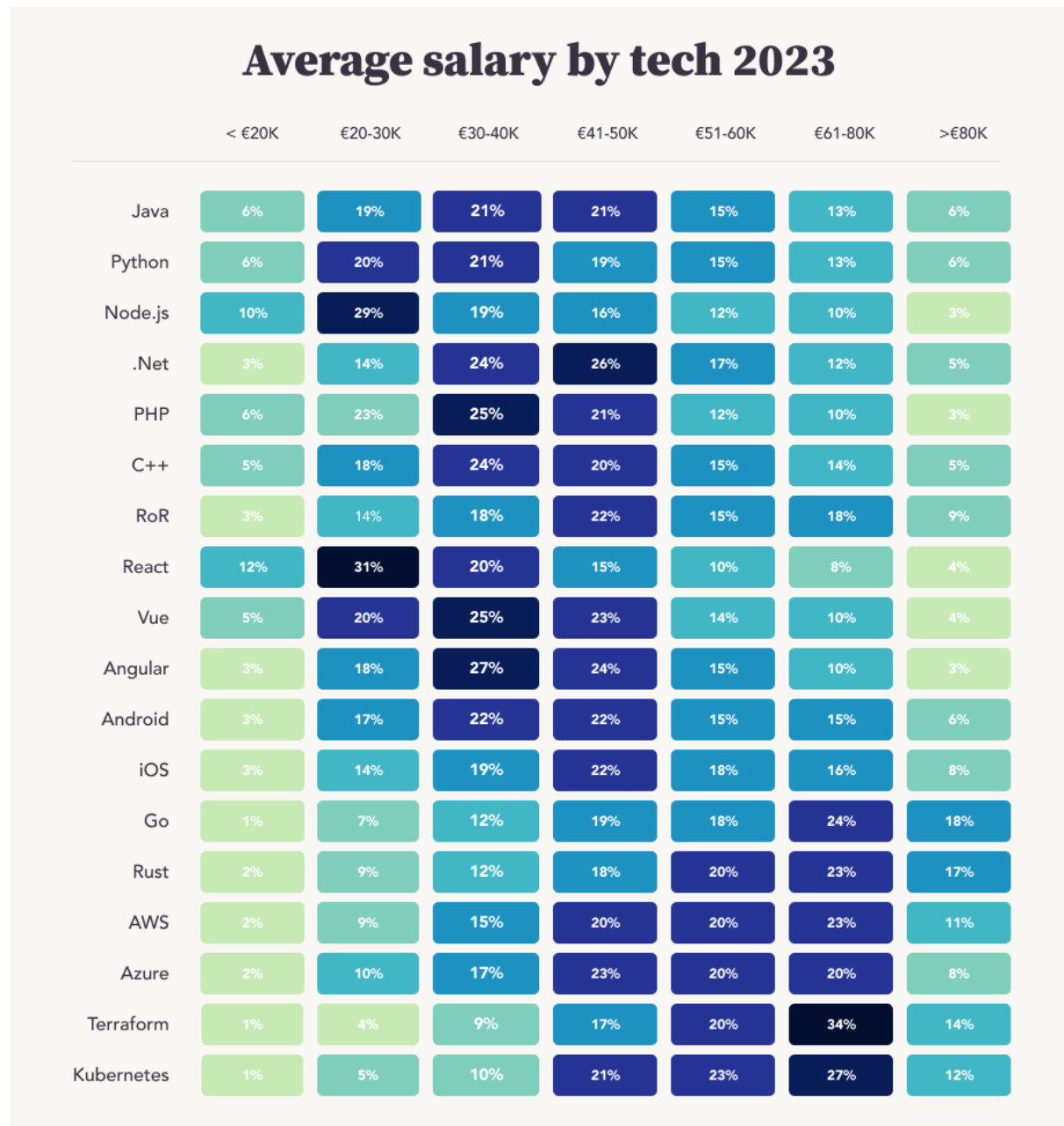
Cualidades clásicas de un/a buen/a programador/a:

- [Cómo ser un buen programador: Aptitudes y habilidades](#)
- [7 cualidades para ser un buen programador](#)
- [Cómo ser un buen programador. Aptitudes que te serán útiles](#)

Lenguajes de Programación

Comparativas de popularidad, empleabilidad y uso de lenguajes de programación:

- [TIOBE Programming Community Index](#)
- [PYPL PopularitY of Programming Language](#)
- [Top Programming Languages 2022 - IEEE Spectrum](#)
- [Stack Overflow Annual Developer Survey](#)



Fuente: [Salarios en tecnología 2023 \[España\] - Manfred](#)

Instalar IDEs

Instalaremos las versiones portables de [Microsoft Visual Studio Code](#) y [Apache NetBeans](#) en los equipos del centro educativo.

Pasos previos:

1. Crear carpeta personal en la unidad local D:/ (Guardar lo mínimo posible en c:/ o en las unidades de red)
2. Crear una carpeta para cada módulo (por ejemplo D:\Programacion)
3. Crear subcarpetas para organizar el módulo:
 - a. una para el software?
 - b. Una para cada UD?
 - c. etc.
4. Configurar descargas del navegador para que siempre pregunte la carpeta de destino.

Paso 1: Descargar versión portable comprimida en .zip (Y guardar en la unidad D:/)

1. <https://code.visualstudio.com/#>
2. [netbeans-19-bin.zip](#)

Paso 2: Descomprimir (también en D:/)

Paso 3:

- En la carpeta donde descomprimiste los programas crea una subcarpeta de nombre “data”. en esa carpeta VScode guardará las extensiones y datos de usuario. Crear dentro otra de nombre “tmp” para archivos temporales.
- Ejecutar vscode: code.exe
- Ejecutar Netbeans: bin/netbeans64.exe

Paso 4:

- En NetBeans: Crear proyecto HolaMundo, Codificarlo y Ejecutarlo.
- En Visual Studio Code: Abrir carpeta src del proyecto anterior. Abrir fichero .java. Instalar plugins recomendados. Ejecutar programa.

Incidencias:

- Configurar proxy (192.168.0.11:3128) en IntelliJ
- En NetBeans, descarga e instalación de “feature” al crear el primer proyecto en 3 equipos.

Conceptos Básicos

- **Algoritmo**: conjunto de instrucciones bien definidas, ordenadas y finitas que permiten resolver un problema.
- **Lenguaje de programación**: idioma o lenguaje con reglas definidas que permite a una persona escribir un programa interpretable por una máquina.
- **Lenguaje o código máquina**: Lenguaje compuesto por un conjunto o lista de instrucciones de bajo nivel que son interpretables directamente por el hardware de un ordenador y dependiente de él.
- **Lenguaje de programación de alto nivel**: Lenguaje de programación más cercano al lenguaje natural de las personas que permite al programador una mayor flexibilidad y abstracción a la hora de expresar las instrucciones pero que requiere de una traducción a lenguaje máquina antes de poder ser ejecutada por la computadora.
- **Programa**: secuencia de instrucciones escritas para realizar una tarea específica en un computador.
- **Código fuente**: Es el texto, instrucciones o líneas de código de un programa escrito en un lenguaje de programación de alto nivel.
- **Compilador**: es un programa informático que traduce el código fuente de un programa escrito en un lenguaje de alto nivel al código máquina ejecutable por un ordenador. El proceso de traducción se denomina *compilación*. En el caso del lenguaje de programación Java el resultado de la compilación es un código denominado **bytecode** ejecutable por la Máquina Virtual Java en cualquier plataforma hardware que la soporte. El tiempo que se tarda en compilar un programa se denomina *tiempo de compilación*.
- **Código objeto**: código resultado del proceso de compilación de código fuente.
- **Intérprete**: programa informático capaz de analizar y ejecutar el código fuente de otros programas, normalmente instrucción a instrucción.
- **Ejecución de un programa**: proceso por el cual una máquina (su procesador) ejecuta las instrucciones de un programa informático previamente traducido a código máquina y cargado en memoria. El tiempo durante el que se ejecuta un programa se denomina *tiempo de ejecución*.

Investiga: Ventajas y desventajas de lenguajes compilados e interpretados. El caso “especial” de Java.

“Hola Mundo”. Ejemplos en distintos lenguajes.

Terminal de Windows:

Ejecutar “cmd”

- **cd**: Cambiar directorio
- **dir**: listar directorio
- **d:** cambiar a la unidad de disco d:

...

C

HolaMundo.c

```
#include <stdio.h>

int main()
{
    printf( "Hola mundo!!!" );

    return 0;
}
```

Compilar en Linux: `gcc holaMundo.c`

Ejecutar en Linux: `./a.out`

Compilar en Windows: [TCC : Tiny C Compiler](#) ([tcc-0.9.27-win64-bin.zip](#)) Descomprimir y añadir al path. Compilar: `tcc HolaMundo.c`

Ejecutar en Windows: `HolaMundo.exe`

Java

HolaMundo.java

```
package holamundo;

public class HolaMundo {

    public static void main(String[] args) {
        System.out.println("Hola Mundo!!!");
    }
}
```

Compilar en Windows: `javac HolaMundo.java`

Ejecutar en Windows:

JavaScript

HolaMundo.html

```
<script>
alert("Hola Mundo!!!");
</script>
```

Ejecutar en Windows: Abrir fichero desde el navegador web.

PHP

HolaMundo.php

```
<?php
echo "Hola Mundo!!!";
```

```
?>
```

[PHP 8.2 para Windows](#): descargar, descomprimir y añadir al path. Ejecutar `php HolaMundo.php`

Python

HolaMundo.py

```
print ("Hola Mundo!!!")
```

Python para [Windows embeddable package \(64-bit\)](#): descargar, descomprimir y añadir al path.

Ejecutar `python HolaMundo.py`

Componentes básicos de un lenguaje de programación

- Conjunto de caracteres
- [Palabras reservadas](#)
- Literales
- Variables
- Identificadores
- Tipos de datos
- Constantes
- Comentarios
- Operadores
- Funciones
- Librerías
- etc.

Investiga: Características principales de los lenguajes más populares.

Práctica: Instala entornos de desarrollo:

- Microsoft Visual Studio Code
- NetBeans

Programación en Java

- [Introducción al lenguaje Java](#)
 - [Descarga JDK](#)
 - [Java API](#)
 - [Documentación JAVA 20](#)

El programa principal

- [Mi primer programa Java](#)

El siguiente código representa el típico programa inicial “HolaMundo” escrito en lenguaje Java:

holamundo\HolaMundo.java

```
package holamundo;

public class HolaMundo {

    public static void main(String[] args) {
        System.out.println("Hola Mundo!!!");
    }
}
```

Un programa Java:

1. Estará formado por uno o varios archivos fuente con extensión .java
2. Estos archivos pueden estar organizados en carpetas o paquetes (*package*)
3. Cada archivo fuente puede contener una o varias clases (*class*). El contenido de cada clase se define entre llaves { }.
4. En cada archivo fuente sólo una clase puede ser pública (*public*). En ese caso el nombre del fichero .java deberá coincidir con el nombre de la clase.
5. Para que un programa Java se pueda ejecutar debe contener una clase principal que tenga un método *main*, que se declara de la siguiente forma:

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

    // Aquí va el código del método main que iniciará el programa.

}
```

6. El contenido del método *main* (las instrucciones o sentencias que forman el programa) o de cualquier otro método se encierran entre llaves { }.
7. En general, cada instrucción o sentencia Java termina en un punto y coma (;)
8. Todo programa Java empieza a ejecutarse a partir del método *main()*.

Todos estos conceptos (clases, paquetes, métodos, etc.) se estudiarán en profundidad a lo largo del curso. No te preocupes si aún no te han quedado claros.

Identificadores y Palabras reservadas

- [Caracteres y secuencias de escape de Java](#)
 - [ASCII - Wikipedia](#)
 - [Unicode](#)
- [Java: Identificadores y palabras reservadas](#)

En todo lenguaje de programación existen una serie de palabras que tienen un significado especial y definen la gramática del lenguaje. Estas palabras no pueden usarse para nombrar variables, constantes, etc. Se denominan **palabras reservadas**. En Java son las siguientes:

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

Además existen tres valores literales: `true`, `false` y `null` que tienen un significado especial en el lenguaje y un estatus similar al de una palabra reservada.

Investiga: Averigua y compara las palabras reservadas de los lenguajes más populares: [C](#), [JavaScript](#), [PHP](#), [Python](#).

Los **identificadores** son los nombres que el programador o programadora asigna a variables, constantes, clases, atributos, métodos, paquetes, etc. a lo largo de un programa.

En Java deben seguir las siguientes reglas:

1. Comienzan siempre por una letra, un subrayado (`_`) o un dólar (`$`).
2. Los siguientes caracteres pueden ser letras, dígitos, subrayado o dólar.
3. Se distingue entre mayúsculas y minúsculas.
4. No pueden ser una palabra reservada.

Es importante que los identificadores elegidos sean significativos, que representen el contenido al que hacen referencia, que sean claros y, si es posible, breves.

Existen reglas de estilo en Java que recomiendan utilizar distintas notaciones según el tipo de identificador que se va a crear. Por ejemplo, se suele utilizar la [notación lowerCamelCase](#) para formar identificadores de variables y métodos que consten de más de una palabra o la notación `UpperCamelCase` para las clases, interfaces, etc.

*Una **guía de estilo** es un conjunto de normas que aplicar a la hora de escribir código fuente en el lenguaje de programación al que van dirigidas. Su objetivo es crear estructuras de código fáciles de entender, no sólo para sí mismo sino también para cualquier programador, hacer más eficiente el proceso de desarrollo y conseguir que los programas sean más robustos y fáciles de mantener.*

Fuente: [guias/java/Guia-codigo-java.md at master · carm-es/guias · GitHub](#)

- [GUÍA DE ESTILO EN JAVA](#) (Escuela Superior de Informática-UCLM)
- [Google Java Style Guide](#)

Algunos ejemplos de identificadores de variables son: edad, maxValor, numCasasLocalidad, notaMediaTercerTrimestre, etc.

Ejemplos y ejercicios de creación de identificadores válidos, que siguen las reglas de estilo de Java. Detección de identificadores no válidos, poco significativos, etc.

Indica si los identificadores son válidos o no, si cumplen con las reglas de estilo de Java. Cuando no sea así, propón identificadores adecuados.

1. numeroDeTelefono
2. \$totalVentas = totalVentas
3. mi_nombre => miNombre
4. 4everYoung
5. _variable
6. #precioProducto
7. usuario-email
8. \$saldoCuenta
9. primeraClase
10. mi_123_variable
11. _nombreUsuario
12. 12345_valor
13. &variable
14. \$precioUnitario
15. #contador_de_inventario
16. miVariable%descuento
17. producto1
18. 3añosDeExperiencia
19. _mi_numero_de_cuenta
20. mi_salario\$\$
21. \$nombreCliente
22. variable#temporal
23. _variable99
24. mi_@direccion
25. precio-total
26. 5HorasTrabajo
27. !descuento
28. codigoDeProducto
29. _Mildentificador
30. valor#unitario

[IA] Puedes usar [chatGPT](#) o equivalente para generar más identificadores y ponerte a prueba.

Prompt (de ejemplo):

Para un ejercicio de programación presenta una lista de identificadores, es decir, sólo los

nombres, algunos válidos y otros no, para que el alumnado compruebe si son identificadores correctos en Java, para que valore si son suficientemente significativos y para que compruebe si siguen las reglas de estilo de Java. No incluyas la solución.

Para continuar:

Añade más identificadores.

Importante: No confíes plenamente en los resultados de los modelos de lenguaje de inteligencia artificial. Complementa con otras fuentes para validarlos. En el caso del módulo de Programación revisa tú mismo el código utilizando el entorno de desarrollo (IDE).

Tipos de datos y Variables

- [Tipos de datos Java](#)

En un programa en ejecución los datos de las variables, constantes, etc. se almacenan en la memoria del ordenador. El tamaño de la memoria se mide en Bytes. Cada dato siempre lleva asociado un **tipo de dato**, que determina el conjunto de valores que puede tomar y, por tanto, el espacio que ocupará en memoria.

En Java existen los siguientes tipos de datos conocidos como *tipos primitivos*: `byte`, `short`, `int`, `long`, `float`, `double`, `char`, `boolean`.

Tipo	Uso	Tamaño (Bytes)	Rango de Valores	Valor por defecto
<code>byte</code>	Número entero corto	1	-128 a 127	0
<code>short</code>	Número entero	2	-32768 a 32767	0
<code>int</code>	Número entero	4	-2147483648 a 2147483647	0
<code>long</code>	Número entero largo	8	-9223372036854775808 a 9223372036854775807	0
<code>float</code>	Número real o Coma flotante de precisión simple	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0
<code>double</code>	Número real o Coma flotante de precisión doble	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0
<code>char</code>	Carácter Unicode	2	\u0000 a \uFFFF	\u0000
<code>boolean</code>	Dato lógico	1 bit	true ó false	false

Una **variable** es una zona de memoria que se referencia con un identificador, conocido como nombre de la variable, donde se almacena el valor de un dato que puede cambiar durante la ejecución del programa.

En Java debemos indicar a qué tipo pertenece cada variable antes de usarla. Esto se denomina *declaración de variables*. Ejemplo:

```
double importe;
```

Se puede *inicializar* asignar un valor inicial a una variable en el mismo momento de declararla:

```
double importe = 120.75;
```

Java no permite usar las variables de tipo primitivo hasta que el usuario las inicialice. Java impide que asignemos a una variable un valor fuera del rango de valores del tipo al que pertenece, produciendo un error.

Ejemplos y ejercicios de declaración de variables.

Asocia un tipo de dato adecuado a los identificadores del ejercicio anterior.

Comprueba la validez de los identificadores en un programa escrito en Java de nombre `Identificadores/identificadores.java`

Investiga: Averigua y compara los tipos de datos de los lenguajes más populares: C, JavaScript, PHP, Python.

Literales y Constantes

- [Programación Java: Literales en Java. Variables y constantes en Java](#)

Un **literal** es un valor de tipo entero, real, lógico, carácter, cadena de caracteres o un valor nulo (`null`) que puede aparecer dentro de un programa.

Los literales en Java, al igual que las variables, se asocian a un tipo de dato:

- **Números enteros:** pueden expresarse en base decimal, binaria (prefijo `0b` ó `0B`), octal (prefijo `0`) o hexadecimal (prefijo `0x` ó `0X`). Por defecto serán de tipo **int** salvo que se añada el sufijo “`le`” (“`l`” o “`L`”) para indicar que tipo **long**. Pueden contener el carácter `_` para facilitar la lectura del número.
- **Números reales:** Son números en base 10, que deben llevar una parte entera, un punto decimal y una parte fraccionaria. Si la parte entera es cero, puede omitirse. El signo `+` al principio es opcional y el signo `-` será obligatorio si el número es negativo. También pueden representarse utilizando la notación científica, incluyendo una `E` seguida del exponente. Por defecto estos literales serán de tipo **double**, salvo que se añada el sufijo “`efe`” (`F` ó `f`) para indicar tipo **float**.
- **Caracteres individuales:** Contienen un solo carácter encerrado entre comillas simples (`' '`). Pueden incluir secuencias de escape y códigos Unicode. Son de tipo **char**.

- **Cadenas de caracteres:** Está formado por 0 ó más caracteres encerrados entre comillas dobles (" "). Pueden incluir secuencias de escape. Las cadenas de caracteres en Java son objetos de tipo **String**.
- **Valores lógicos:** verdadero (**true**) o falso (**false**), de tipo **boolean**.

Los literales suelen aparecer en la asignación de valores a las variables o formando parte de expresiones aritméticas o lógicas.

Un programa puede tener valores que no cambian a lo largo de su ejecución que se denominan constantes.

Una **constante** es una zona de memoria que se referencia con un identificador, conocido como nombre de la constante, donde se almacena un valor que no puede cambiar durante la ejecución del programa.

Una constante en Java se declara de forma similar a una variable, añadiendo la palabra **final** al inicio.

Según las reglas de estilo de Java el identificador de una constante debe escribirse con mayúsculas y, si consta de varias palabras, separarlas con el carácter de subrayado (_).

Ejemplo:

```
final int NUMERO_ALUMNOS = 33;
```

Comentarios

- [Comentarios en Java](#)

Un comentario es un texto que se utiliza para explicar y documentar el código fuente y que el compilador no tiene en cuenta al ejecutar el programa.

En Java hay 3 tipos de comentarios:

- **Multilínea:** cualquier texto incluido entre los símbolos de apertura **/*** y cierre ***/** será considerado un comentario. Puede extenderse por varias líneas.
- **En línea (hasta el final de la línea):** todo lo que sigue al símbolo **//**
- de documentación: **/** ... */** hasta el final de la línea se considerará un comentario.
- **De documentación:** Como un comentario multilínea pero con el símbolo de inicio **/****. Es usado por algunas herramientas para generar documentación automática.

Completa el programa `identificadores.java` inicializando cada variable a un valor apropiado al tipo con el que la hayas definido.

Añade un comentario donde hayas modificado el nombre del identificador, indicando el nombre original.

Tipos Primitivos. En el mismo paquete, realiza un nuevo programa que declare una variable de cada uno de los 8 tipos primitivos de Java, que las inicialice a un valor a elección del alumno/a y que imprima sus valores, cada una en una línea distinta, con un texto explicativo apropiado e indicando también el espacio que cada una en memoria. (Por ejemplo: "La variable varDouble es de tipo double, tiene el valor 4.78 y ocupa 64 bits en memoria")

Preferiblemente imprime el resultado en formato tabla. Ejemplo:

Tipo	NumBits	Valor
byte	8	124
short	16	12234
...		

Sugerencia: usa el carácter de escape `\t` de tabulación para alinear las columnas.

Constantes. En el mismo paquete, realiza un nuevo programa que declare, inicialice e imprima por pantalla las siguientes constantes:

- Número de alumnos matriculados en el módulo de Programación (32 según el listado del aula virtual)
- Número total de sesiones del módulo de Programación en el curso lectivo.
- Número de sesiones semanales del módulo de Programación.
- Número de meses del año.

Entrega la carpeta del paquete comprimida en zip en el aula virtual: [UD01.Entregas](#)

API de Java

Una [API](#) (*application programming interface*), o interfaz de programación de aplicaciones, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software.

Java dispone de una amplísima biblioteca de herramientas para realizar tareas complejas facilitando la tarea del programador/a.

La mayoría de estas herramientas se denominan **clases** y están organizadas en **módulos** y **paquetes**. Cada clase se identifica por su nombre completo o cualificado, que incluye el nombre del paquete y de la clase. Por ejemplo, la clase **Math** contiene herramientas para cálculos matemáticos complejos y se identifica por el nombre cualificado `java.lang.Math`.

Algunas de las funcionalidades básicas que ofrecen estas herramientas son:

- **Salida de datos:** permiten la escritura de información en dispositivos de salida: monitores, impresoras, dispositivos de almacenamiento, etc. Ejemplo: clase [java.lang.System](#).
- **Entrada de datos:** permiten leer información desde teclado, de un fichero o desde otros dispositivos. Ejemplo: [java.util.Scanner](#).
- **Cálculos complejos:** realizan operaciones matemáticas como raíces cuadradas, logaritmos, cálculos trigonométricos, etc. Ejemplo: [java.lang.Math](#).
- **Fechas y horas:** permiten obtener la fecha y hora del sistema y realizar operaciones relacionadas con el tiempo. Ejemplo: [java.time.LocalDate](#), [java.time.LocalDateTime](#), [java.time.LocalTime](#).

Una clase proporciona funcionalidades, normalmente en forma de **métodos**. Por ejemplo la clase **Math** dispone de los métodos `abs()` o `sqrt()` para calcular el valor absoluto o la raíz cuadrada de un número, respectivamente. También puede incluir datos útiles, como por ejemplo la **constante PI** para cálculos trigonométricos.

Algunas clases útiles para trabajar con fechas y horas son las clases **LocalTime**, **LocalDate** y **LocalDateTime**. Todas ellas se encuentran en el paquete `java.time`. Un ejemplo de método útil que ofrecen todas ellas es `now()` que permite obtener la hora y/o fecha de sistema.

Para usar este y otros métodos podemos usar el nombre cualificado de la clase. Por ejemplo:

```
java.time.LocalTime.now()
```

Para hacer menos engorroso el uso de las clases de la API podemos indicar al inicio de nuestro código mediante la palabra reservada **import** que clases queremos importar y luego utilizarlas directamente con su nombre sencillo. Por ejemplo:

```
import java.time.LocalDateTime;
...
LocalTime.now()
```

Podemos importar varias clases de un mismo paquete usando el símbolo asterisco (*).

```
import java.time.*;
```

El paquete **java.lang** contiene las clases fundamentales para programar en Java sus clases se incluyen de forma automática sin necesidad de importarlas.

API de Java. Explora en Internet la [API de Java](#) correspondiente al JDK que estés utilizando y familiarízate con el formato de la documentación.

- Agrega el enlace a los marcadores de tu navegador.
- Busca el paquete java.lang y explora su contenido.
- Busca la clase System y su campo out que utilizamos para mostrar información por consola.
- Busca la clase Math y observa las propiedades y métodos que ofrece.
- Busca el paquete java.time
- Busca la clase Scanner en el paquete java.util.

En general, las clases de la API pueden utilizarse de dos formas:

- **De forma estática:** usamos directamente el método, constante o recurso, simplemente anteponiendo el nombre de la clase. Por ejemplo, los métodos y constantes que ofrece la clase [Math](#) se utilizan de forma estática. Por ejemplo:

```
double raiz = Math.sqrt(16);
double circunferencia = 2 * Math.PI * radio;
```

- **De forma no estática:** Para usar estos métodos o propiedades debemos antes crear un objeto de esa clase usando el operador **new** que se verá en profundidad más adelante. Un ejemplo de clase que usaremos de esta forma es la clase [Scanner](#).

Podemos utilizar la clase **Scanner**, del paquete **java.util**, para leer datos de teclado. Para utilizar sus funciones será necesario primero declarar (crear o instanciar) un objeto con la palabra reservada **new** de esa clase antes de poder usar sus métodos.

```
Scanner sc = new Scanner(System.in)
```

El código anterior declara una variable de tipo objeto (Scanner), de nombre `sc`, que permitirá leer datos de la entrada estándar del sistema (System.in), es decir, del teclado.

Los métodos más habituales de un objeto de tipo Scanner para leer datos son:

- **nextInt():** lee un número entero.
- **nextDouble():** lee un número real.

- **nextLine():** lee una línea entera como una cadena de caracteres.
- **next():** Lee una cadena de caracteres hasta que se encuentra con un separados (tabulador, espacio en blanco o salto de línea)

Método `useLocale(Locale.US)` . Permite leer números reales de teclado usando el punto (.), notación americana, como separador decimal.

- [Java Scanner para lectura de datos](#)
- [Leer un char por teclado en Java](#)
- [Java printf para dar formato a los datos de salida](#)
- [Utilizar DecimalFormat para dar formato a los datos de salida](#)
- [Utilizar String.format para dar formato a los datos de salida](#)

Crea los siguientes programas en un nuevo paquete de nombre `api`:

HoraLocal. Escribe un programa que utilice la clase `LocalTime` para imprimir por pantalla la hora del sistema.

LeerNumeroEntero. Solicita un número por teclado y muéstralo.

RaizCuadrada. Escribe un programa que utilice la clase `Scanner` para solicitar un número por teclado, que utilice la clase `Math` para calcular su raíz cuadrada y que imprima el resultado.

Investiga: ¿Cómo podrías formatear la salida por pantalla para presentar solo dos decimales?

LeerTiposPrimitivos. Escribe un programa que solicite al usuario diferentes datos para almacenarlos en variable de cada uno de los tipos primitivos. Es decir, el programa solicita al usuario introducir un dato de tipo `byte`, otro de tipo `short`, otro `int`, etc. y los almacena en variables del tipo adecuado.

Incluye también un dato de tipo `String` o cadena de caracteres.

A continuación el programa imprimirá los valores recibidos. Utiliza en cada caso el método apropiado de la clase `Scanner`.

PaquetesJava. Investiga en Internet y haz un listado de los principales paquetes de Java y su utilidad (entre 10 y 20). (`lang`, `io`, `util`, `applet`, `awt`, `swing`, `net`, `math`, `sql`, `security`, etc.)


Operaciones básicas

- [Operadores Java](#)

- Operadores de asignación
- Operadores aritméticos
- Operadores relacionales
- Operadores lógicos
- Operador ternario

Jerarquía de operadores (o precedencia)

Realiza los siguientes [ejercicios](#) en un nuevo paquete de nombre **expresiones**:

-  UD01.Ejercicios de Operadores y Expresiones

Ejemplos:

Crear expresiones.

A partir de fórmulas algebraicas

$$\frac{3}{2} + \frac{4}{3}$$

A partir de expresiones algorítmicas

Comprobar si la última cifra de un número entero N es par

Evaluar Expresiones.

Si $a=8$, $b=3$, $c=-5$, determina el resultado:

$$A. 2 * b + 3 * (a - c)$$

Puedes probar resultados en un programa Java.

Conversión de Tipos

Todos los datos y variables en Java tienen un tipo asociado. Cada tipo ocupa un tamaño en memoria. Cuando asignamos un valor a una variable, ambos deben ser del mismo tipo.

Si intentamos asignar un valor `double` a una variable `int`, por ejemplo:

```
int a = 5.6;
```

el compilador nos avisará de que estamos cometiendo un error y no permitirá ejecutar el programa. El motivo principal es que el valor 5.6 de tipo `double` ocupa 64 bits en memoria y no se puede almacenar en una variable `int` que ocupa 32 bits.

Sin embargo, Java sí permite la operación contraria, es decir, almacenar un valor `int` en una variable `double`, ya que la variable `double` de 64 bits sí tendría espacio suficiente para almacenar un valor `int` de 32 bits, por ejemplo:

```
double a = 56;
```

Para ello, el compilador realiza una **conversión automática (implícita o de ensanchamiento)** de tipos.

Sin embargo, en ocasiones sí es interesante poder realizar conversiones como las del primer ejemplo, es decir, de un tipo `double` o una variable más “grande” a un tipo `int` o una variable más “pequeña”. Un ejemplo de uso sería guardar la parte entera de un número con decimales en una variable entera. Este tipo de **conversión, explícita o de estrechamiento**, supone una pérdida de información, ya que los decimales desaparecerán, y requiere que el programador lo indique con una sintaxis específica que consiste en indicar un `cast` o molde indicando entre paréntesis el tipo de destino delante del valor que queremos asignar:

```
int a = (int) 5.6;
```

En el ejemplo anterior el valor `double 5.6` se convertirá (*estrechará*) a un tipo `int` antes de asignarlo a la variable `a`, evitando así el error del compilador.

Existen dos tipos de conversiones de tipos de datos en Java: implícitas y explícitas.

Conversiones implícitas.

Son aquellas que se realizan de forma automática cuando se asigna un valor o expresión de un tipo a una variable de un tipo diferente.

Para que una conversión implícita pueda realizarse el tipo de la variable de destino debe tener un rango o precisión igual o superior al del valor que se quiera guardar en ella.

Conversiones implícitas posibles:

Tipo del valor de origen	Tipo de la variable de destino
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

En el caso de las expresiones, Java automáticamente convierte todos los tipos de datos de los valores implicados al tipo de dato de mayor precisión.

Conversiones explícitas.

Cuando la conversión automática es imposible para el compilador, es posible forzarla mediante una conversión explícita (también conocido como *casting* o refundición). Para ello se utiliza la siguiente sintaxis:

```
(tipo_destino) variable_a_convertir
```

Las conversiones explícitas pueden producir una pérdida de información, ya que la precisión del tipo de destino es menor. Por eso estas conversiones también se conocen como “estrechamiento”.

En general, se recomienda evitar las conversiones de tipos siempre que sea posible.

Ejercicios

Crea los siguientes programas en un nuevo paquete de nombre operadores:

E0103. Solicita la edad del usuario y muestra la que tendrá el año que viene.

E0104. Solicita el año actual y el año de nacimiento del usuario y muestra la edad del usuario suponiendo que en el año en curso ya haya cumplido años.

Alternativa E0104b: usa la clase `LocalDate` para obtener el año actual.

E0105. Programa que muestre que el rango de valores de un tipo `short` se comporta de forma cíclica, es decir, que el valor siguiente al máximo es el valor mínimo.

E0106. Crea una aplicación que calcule la media aritmética de dos notas enteras. Ten en cuenta que la media puede tener decimales.

E0107. Aplicación que calcule el perímetro y el área de un círculo a partir del valor del radio introducido por teclado. El radio puede contener decimales.

E0108. Crea un programa que solicite al usuario su edad y muestre si es o no mayor de edad imprimiendo un literal booleano: `true` o `false`.

E0109. Programa que pida un número entero al usuario y que indique si es par mediante un literal booleano (`true` o `false`).

E0110. Diseña un algoritmo que indique si podemos o no salir a la calle, lo que se determinará en función de ciertas condiciones, y que lo muestre mediante un literal booleano (`true` o `false`). En principio, podremos salir a la calle siempre que no llueva y que hayamos terminado nuestras tareas. Además de lo anterior podremos salir a la calle si

tenemos que hacer algún recado, como ir al supermercado. Para obtener el resultado, el programa debe preguntar por consola y mediante valores booleanos si está lloviendo, si hemos terminado nuestras tareas y si tenemos que hacer algún recado.

llueve?	tengoTareas?	tengoRecados?	podemosSalir?
false	false	false	true
false	false	true	true
false	true	false	false
false	true	true	true
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	true

E0111. Un frutero necesita calcular los beneficios anuales que obtiene de la venta de manzanas y peras. Escribe una aplicación que solicite las ventas (en kilos) de cada semestre para cada fruta y que muestre el importe total siendo 2,35 € el precio del kilo de manzanas y 1,95 el del kilo de peras.

E0112. Escribe un programa que pida un número entero al usuario y muestre su valor absoluto.

E0113. Escribe un programa que solicite por teclado las notas del primer, segundo y tercer trimestre del curso, que serán números enteros. El programa deberá mostrar la nota media del curso como se utiliza en el boletín de calificaciones (sólo la parte entera) y como se usa en el expediente académico (con decimales).

E0114. Escribe un programa que solicite por teclado un número decimal y lo redondee al entero más próximo.

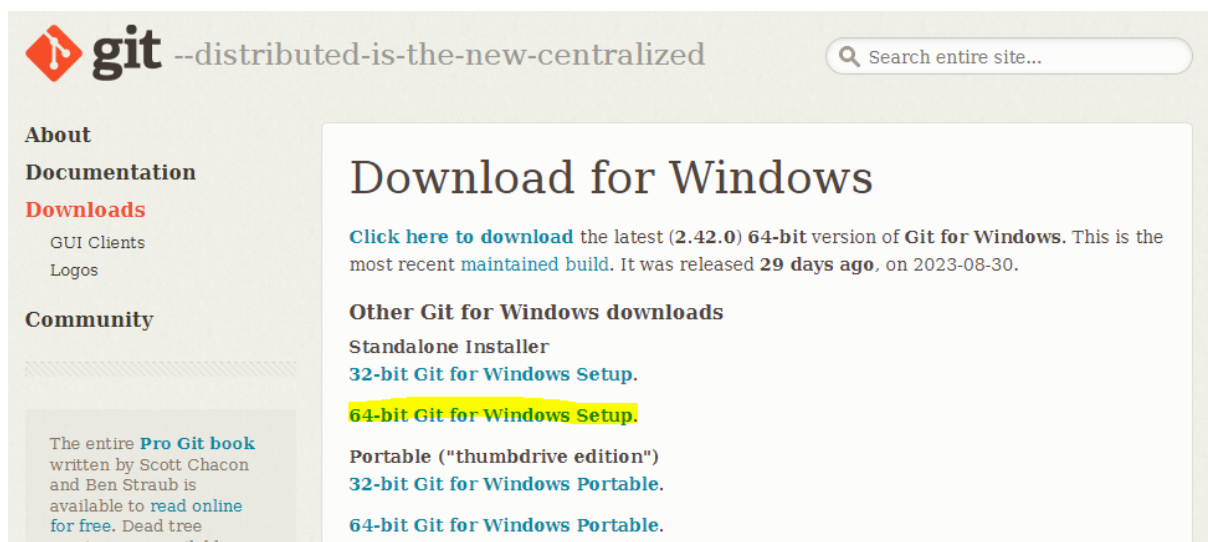
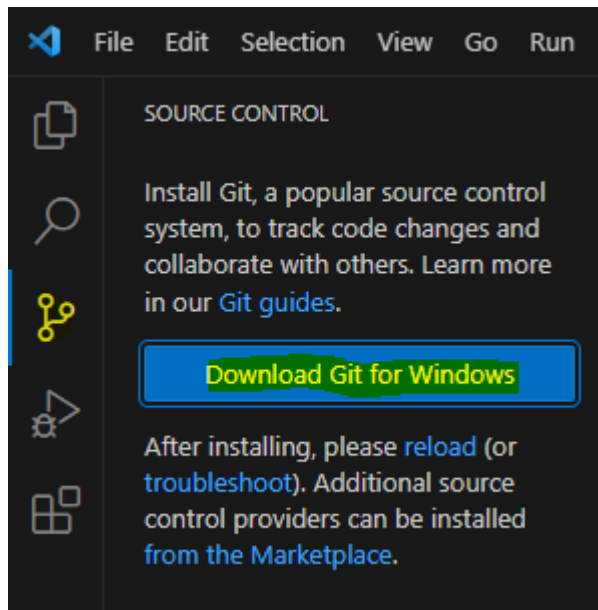
(Pendiente) Más ejercicios: [UD01.Ejercicios Propuestos](#)

Anexo. Buenas Prácticas

- [Consejos y Buenas Prácticas en Programación](#)

- Respeta la guía de estilo a la hora de escribir identificadores de variables, constantes, clases, paquetes, etc.
 - paquetes
 - Clases
 - variables
 - CONSTANTES
 - etc.
- Evita identificadores de una única letra, salvo excepciones bien justificadas.
- Evitar uso de “ñ” y otros caracteres especiales.
- Procura que los identificadores sean significativos y autoexplicativos.
- Escribe una instrucción por línea de código, no más.
- Una instrucción compleja puede alargarse en más de una línea.
- Evita las líneas demasiado largas (más de 80 a 100 columnas).
Tener que desplazar la pantalla horizontalmente para leer el código dificulta la legibilidad.
- Rodea los operadores por espacios.
- Separa bloques con líneas en blanco cuando sea conveniente.
- Utiliza el sangrado dentro de los bloques.
- Usa comentarios cuando sea necesario para indicar qué hace el código y cómo lo hace.
- Depuración: Usar comentarios para aislar los errores.
- ...

Anexo. Instalar y configurar Git y GitHub



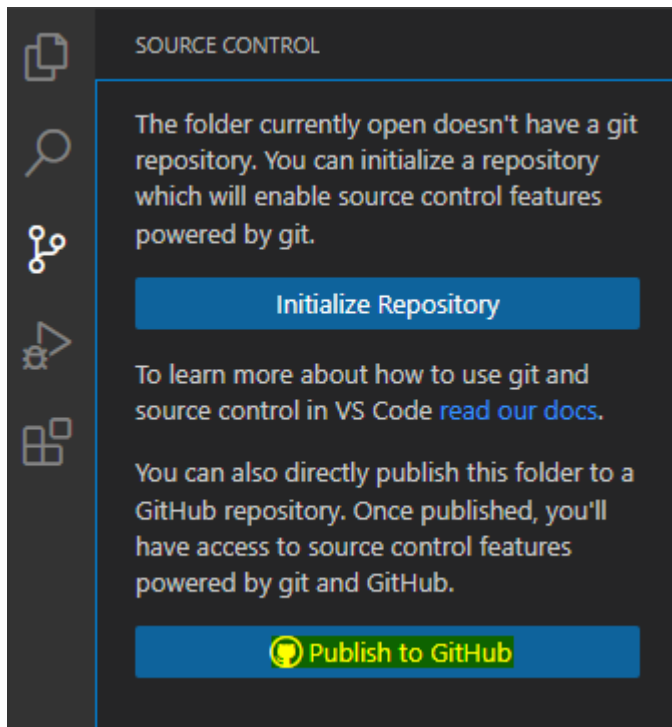
[En CDM] Descargar en unidad D:/

Instalar (puedes pulsar "Siguiente / Next" en casi todas las pantallas del asistente)

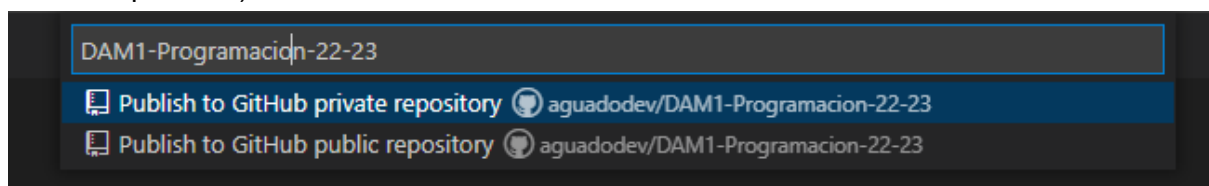
Crear y Configurar un repositorio remoto

1. Requisitos previos: tener cuenta en [GitHub](#) y tener instalado [Git](#) en el equipo local.
2. Partimos de una carpeta de proyecto en la que crearemos el repositorio.
 - a. Si ya tenemos un repositorio creado podemos eliminarlo borrando la carpeta oculta ".git" desde el explorador de archivos.

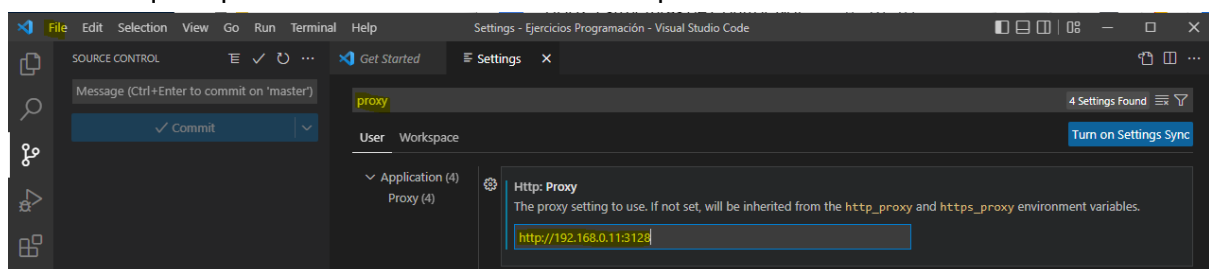
- Desde Ms Visual Studio en el apartado “Source Control” elegimos “Publish to GitHub” para crear y publicar el repositorio directamente.



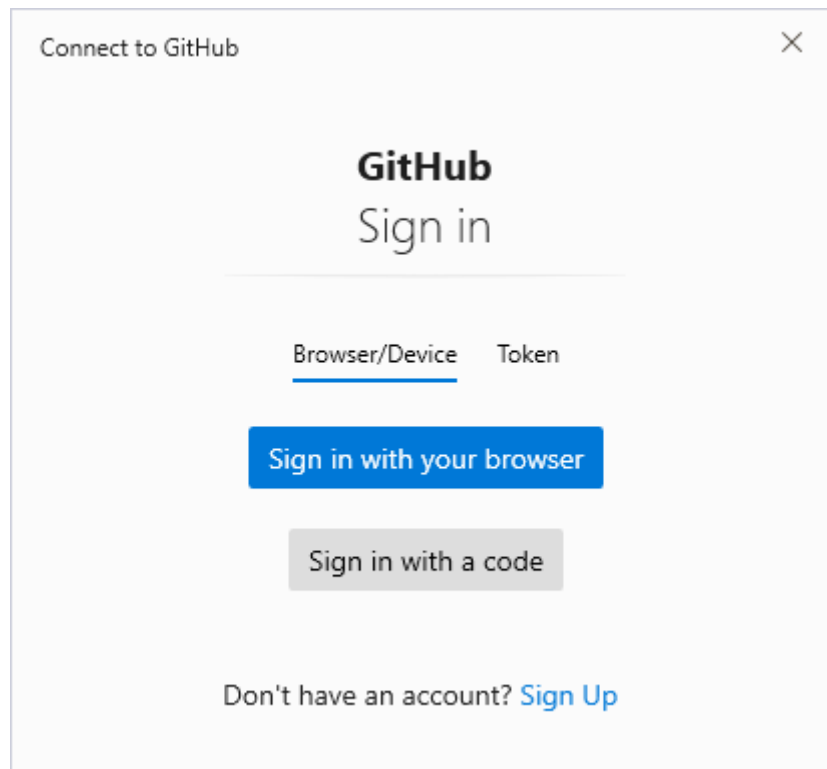
- Damos nombre al repositorio remoto (ojo con acentos, eñes y otros caracteres especiales):



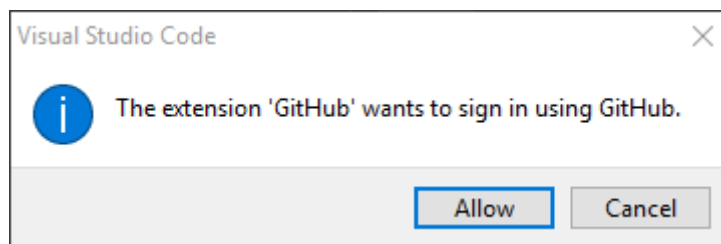
- Si tenemos problemas de conexión puede que tengamos que configurar el proxy de la organización en Visual Studio Code accediendo a File > Preferences y buscando el término “proxy”. O quizá incluso crear las variables de entorno que se indican: **http_proxy** y **https_proxy**. Si no tenemos permisos de administrador sobre la máquina podemos crear las variables solo para la cuenta de usuario actual.



- Para confirmar los cambios en el repositorio por primera vez será necesario identificarnos como autor/a con los comandos
git user.email ..
git user.name...
- En algún momento Visual Studio Code nos pedirá autenticarnos en GitHub



8. La extensión de GitHub de Visual Studio Code puede también solicitarnos autenticarnos en GitHub.



9. ...

Preguntas test con ChatGPT

Prompt de ejemplo para generar preguntas test de los contenidos del tema.

Crea un listado de preguntas tipo test de respuesta múltiple en formato AIKEN para Moodle sobre el tema "Introducción a la programación en Java", que incluya preguntas sobre la estructura básica del programa, entrada y salida básicas, uso básico de la api y operadores aritméticos, relacionales, lógicos y ternario.

...

Incluir la respuesta correcta.

...

Genera más preguntas en el mismo formato de la misma temática