👋

Intro: Programación

En esta clase veremos lo que es la **programación.**

Desde como funciona un lenguaje de programación (en concreto uno que se llama Java) — que es uno de los principales y como poder trabajar y hacer diferentes cosas programando.

⚠️ **Lo que no quiero que a la gente se le quede programando:**

> *"Ah es que voy a programar en Java y solo me quedo programando en Java..."*

No, ni mucho menos.

Vamos a hacer la asignatura de programación programando en Java — pero si luego te quieres cambiar a cualquier otro lenguaje **no te va a costar mucho.**

Recursos

Nosotros como recursos tenemos el **aula virutal** (donde tenemos todos los contenidos):

Y tenemos **teams** para conectarnos y ver las clases o contactar con el tutor (Borja).

No debemos preocuparnos si no podemos ver las clases en las horas indicadas en el horario, porque podemos enviarle un mensaje directo y el en cuanto pueda se va a acoplar a nosotros.

Contenidos

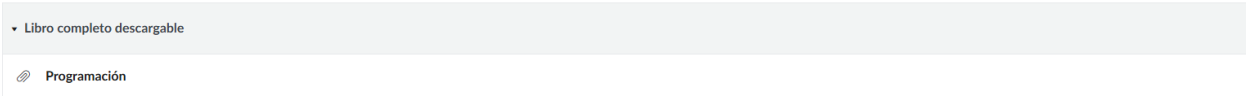
Vamos a hablar un poco sobre los contenidos.

https://campus.europaeducationgroup.es/courses/114658/modules

Todo lo que tenemos en estos **contenidos** es lo que vamos a dar en clase.

Y en concreto, todo lo que tenemos en cada uno de los módulos, pues tiene su contenido, scorn, ejercicios, etc...

Y si vamos abajo del todo veremos que tenemos una cosa que se llama **Libro:**



Donde nos podemos bajar el **PDF** entero donde vamos a ver todo lo que va poniendo en el scorn incluso un poquito más.

Esto es todo el contenido que se dara en la asignatura.

El profesor (Borja) generara sus propios contenidos, no dependeremos solo del libro. Si que es verdad que tenemos todos los módulos y todo ubicado en el libro.

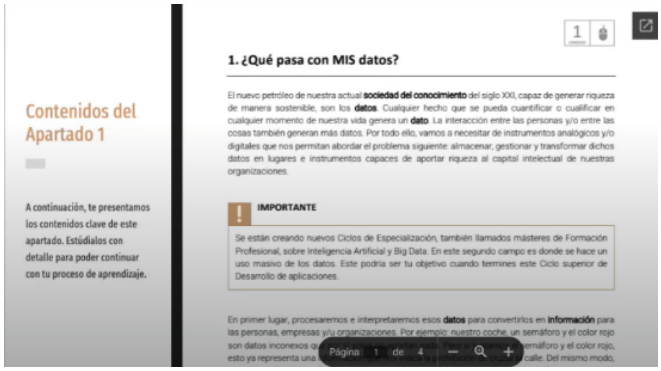
Pero además de esto, Borja nos da una serie de materiales adicionales.

Dinamica de las clases

Antes de meternos en todo esto, vamos a hablar de **como serán las clases.** Bien, pues tenemos una **tutoria colectiva** cada 15 días. Que es de `18:30` a `19:30` → lo único que la vamos a empezar un poquito antes y terminarla un poquito más tarde.

Si Borja (el profe) ve que hace falta dar una tutoria colectiva adicional, puede avisarnos y enviarnos enlace para que nos conectemos, como un bonus extra.

Las clases como tal, el profesor no se pondra a leer las unidades una a uno, no tiene sentido eso podemos hacerlo nosotros en casa.



Como van a ser las clases entonces?

Borja abrira el **IDE** (Entorno de Desarrollo) → y se va a poner a programar. Nos pondremos manos a la obra.

Y podremos ir practicando tantas veces como necesitemos. La recomendación es ir viendo lo que hara Borja y entenderlo. Si hace falta incluso preguntarselo, pero **núnca con un PDF delante**, siempre sera con un IDE abierto.

> 👨‍💻 **A programar se aprende *programando***
> No se aprende leyendo documentos.

Borja nos explicara la parte de programación enseñandonos directamente sobre el IDE, y luego nosotros tendremos que ir programando.

0 teoría, y la teoría la explicara según vayamos picando.

Actividades evaluables

A lo largo del curso, hay 3 actividades evaluables:

1. Una que está a ~ finales de **Noviembre**
2. Otra a ~ mitad de **Febrero**
3. Y una última ~ a mitad de **Abril**

Esos son los ejercicios evaluables que **hay que entregar** — que el nos puede evaluar.

Ahora, Borja nos va a proponer una cantidad **ingente** (muy grande) de ejercicios para que nosotros practiquemos.

No solo los que vamos a hacer en clase, pero sino que el nos dara *muuuchiiisimos* ejercicios para que podamos ir practicandolos.

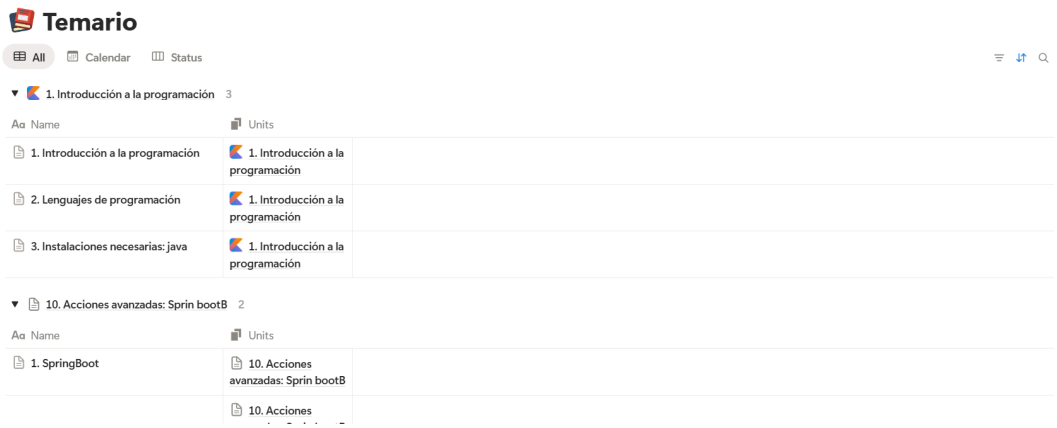
No podremos resolver todos en clase, porque nos va a dar tropecientos mil.

Sobretodo entender que **las clases no seran teorcias, seran practicas.**

Recursos adicionales

Los recursos adicionales los tenemos aquí:

Y podemos ver por ejemplo el **temario:**



Hay algunas cosas que no coincidaran con el temario del curso.

Esto es mucho más de lo que tenemos en el contenido de los scorns — es contenido adicional.

Pros y Contras

En este mundillo de la programación, hay dos cosas — una positiva y una negativa.

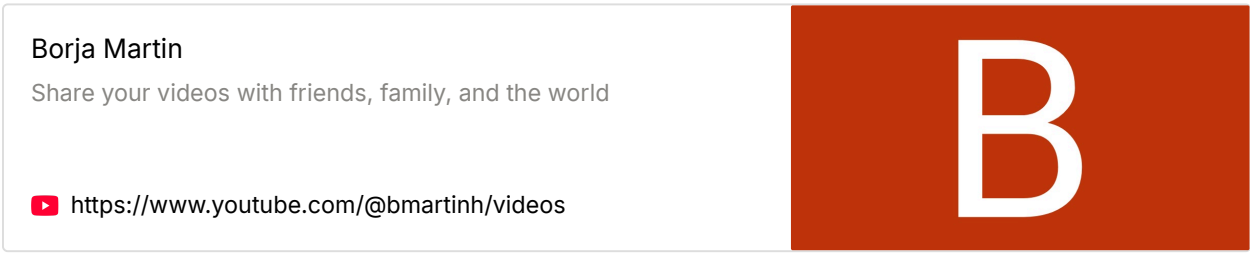
Positiva: vamos a entrar en un mundo en el que el trabajo no falta, que se cobra decentemente bien.

Negativa: empezamos a estudiar hoy 1 de octubre, pero no vamos a dejar de estudiar JAMAS — todos los dias van saliendo cosas buenas. Y lo que hoy vale, dentro de nada queda obsoleto.

Es un mundo de constante cambio. Lo malo es que vamos a tener que estar reciclandonos todo el dia.

Canal de YouTube de Borja

Tenemos un canal de YouTube de nuestro profesor **Borja** en el que repasa temas concretos



Aunque salga en YouTube, solo puede verlo gente que tenga el enlace. Podemos acceder desde aquí:

Sobre preguntar

Antes de empezar con los **contenidos** — tenemos que saber que lo más básico de la programación son solo 2 cosas:

1. **A programar se aprende programando:** No solo viendo... por mucho que vengamos a todas las clases y nosotros veamos todos los códigos que Borja

hace, si nosotros no picamos teclas, no aprendemos. Es imposible. La única manera de aprender es programando.

2. **No se da por supuesto NADA:** Aquel que empiece de 0, fenomenal. Aquel que sepa algo, genial. Pero Borja va a empezar de 0. Porque no va a dar por sentado que nadie sabe nada. El que empiece de 0 que no se agobie.

Y como se parte de 0, lo más importante de esto — aplicado a cualquier cosa — es **preguntar**.

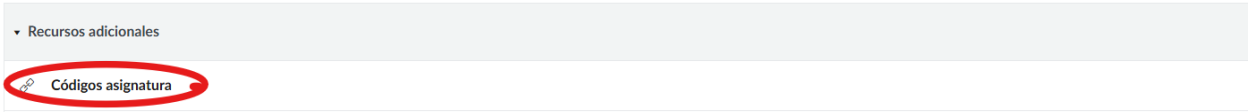
Si Borja explica algo, y no lo entendemos, si alguien no lo entiende que levante la mano o que corte y pregunte. Que no se corte.

> No hay preguntas tontas, sino tontos que no preguntan.

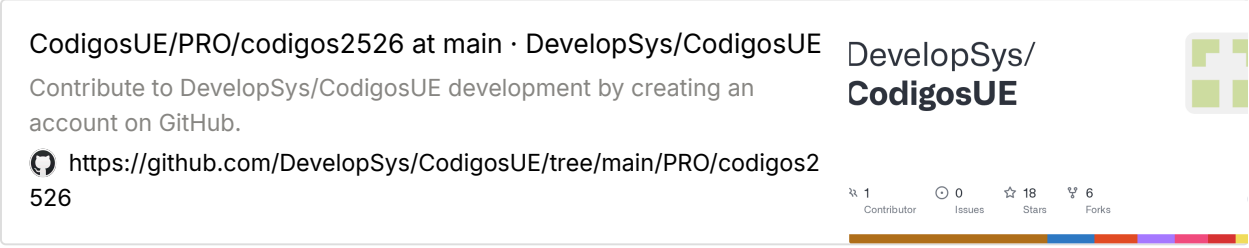
La única manera de ganarte la vida de esto, es aprendiendo. Y la forma de aprender es preguntando.

Códigos de asignatura: repo de Github

Todos los códigos de la asignatura estaran aquí:



Almacenados en una repo de Github:



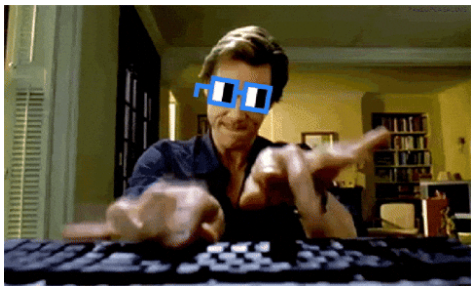
Es el repositorio de la asignatura.

Esto es una cosa que se llama **Github** (explicada en Entornos de Desarrollo) — una plataforma donde los programadores dejamos códigos en una especie de nube.

Todos los códigos estarán aquí — y todos los códigos iran ahi puestos.

1) Que es la programación?

En el primer tema veremos lo que es la **programación** — que leches es esto de programar? Porque la gente se piensa que es programar es darle a 4 teclas, escribir código en una pantalla negra a lo Mr. Robot y darle al play.



Ya te digo yo que **no.**

Olvidate de eso.

Porgramar **NO ES FÁCIL.** Pero tampoco es **díficil.**

Hablaremos un poco de que es programar, que son los algoritmos, cuales son los lenguajes de programación que existen, y una breve introducción a que es JAVA.

2) Introducción a JAVA

Una vez visto que es la programación, hablaremos en profundidad sobre **JAVA.**

Crearemos un **proyecto**, hablaremos de las **estructuras fundamentales** del proyecto, que es una **clase**, que es un **método**, que es una **variable**... Como se utilizan, dentro de un `main`...

Vamos a crear unas cuantas variables, iremos haciendo cositas.

3) Estructuras de control

Una vez visto esto, vamos a hablar de las **estructuras de control** — tenemos un código de programación y no queremos que todo se ejecute de arriba a abajo.

Queremos que se vaya ejecutando de arriba a abajo, pero en un momento determinado, enfrentando una decisión, igual queremos ejecutar otra línea o otro código dependiendo de la condición.

- Si la persona es mayor de 18 años → le dejamos beber alcohol
  - Si no → fuera
- Si el precio es más de 10 € → le aplicamos descuento
  - Si no → lo dejamos igual
- Repetir este código x veces...
- Sigue ejecutando esto mientras van pasando cositas.

Veremos que hay **tres tipos de estructuras de control:**

1. Las de selección
2. Las de repetición
3. Las de salto

4) Estructuras de datos estáticas

En vez de guardar un número en una variable, guardaremos unos cuantos números. Como una colección. En vez de guardar una mascota, guardamos varias.

Y es estática porque no la puedo hacer ni crecer, ni decrecer. Es el tamaño que le hayamos dicho. A un armario no le puedo hacer ni más grande ni más pequeño, es el tamaño que hay y punto.

5) Programación modular

Hablaremos un poco de la **programación modular** — que es esto de los métodos, como podemos crearlos, reutilizarlos, que devuelva cosas, que nos pida parametros para utilizarlos....

6) POO: Programación orientada a objetos

Hablaremos de la **programación orientada a objetos (POO)** → que es uno de los *principales paradigmas* de la programación hoy en dia.

> POO = Programación orientada a objetos (de forma abreviada)

Hasta este momento, habremos ejecutado de línea en línea nuestro código — pero cuando hablemos de **POO** la mente se nos va a abrir.

Y aquí es donde empezaremos a hacer el click.

Ejemplo de POO en Minecraft:
En Minecraft hay ***bloques*** que hay que ir picando e ir recolectando. El oro, la hierba, el cemento... Y esos bloques son los mismos.
Es un bloque, lo que pasa es que va cambiando su aspecto. Uno es verde, otro es ázul, otro es amarillo, algunos se caen, otros tienen diferentes durezas. Algunos tienen diferentes valores.
Pero esos 800 bloques, estan representados por un objeto → que es el objeto de tipo **bloque** — es decir abstraemos el concepto de “bloque”. Eso es una clase.

Ejemplo puesto por Adrian:

Podriamos decir que la clase es como una fabrica de objetos. Es decir, imaginad un cortador de galletas.

Este cortador de galletas, seria la **clase**.

Y luego gracias a ese cortador, podemos ir creando **galletas** de tipo **Galleta** — suena confuso ahora pero es simplemente abstraer el concepto de “Galleta” (que seria el “cortador”) para crear galletas:

La clase seria el **cortador metalico** — que fabrica las “instancias” o objetos de tipo galleta que queramos:

Pero el “fabricador” osea la clase — seria siempre la misma.

7) Utilización de objetos y desarrollo de clases

Una vez hayamos aprendido sobre la POO — podemos empezar a ver buenas practicas, creación de clases más complejas, etc...

8) Utilización avanzada de clases

Herencia, polimorfismo, abstracción... Temas más avanzados sobre la programación orientada a objetos entorno a las clases. Que ahora mismo serian complicados de explicar sin haber cubierto antes POO básica.

9) Colecciones dinámicas de datos

Muy parecido a lo que hablamos en la **unidad cuatro** (estructuras estáticas) pero que ahora **si que pueden mutar de tamaño.** Como por ejemplo un carton de pelotas de tenis, en el que podemos ir metiendo y sacando pelotas.

Por eso se le llama una ***estructura dinámica*** (y no estática) → porque puede ir cambiando de tamaño.

- Sin embargo en las estáticas decimos “es un armario de 4 baldas” ya ahi se queda. No pueden haber ni más ni menos. Es estático.

10) Control y manejo de excepciones

Básicamente son momentos en el que el código o la estructura de un programa pueden fallar, no porque yo lo tenga mal escrito (sintaxis de Java) → sino porque por cuestiones de lógica, conexión a internet, factores externos puede fallar.

El programa compila (se ejecuta) perfectamente, pero algo que no controla JAVA en si lo ha hecho fallar. Pues igual es que hay algo adicional que esta fallando y no lo estamos controlando.

Lo controlaremos con el **manejo de excepciones.**

11) Lectura y escritura de información

Gran parte del curso, nosotros lo que vamos a ir viendo es **leer y escribir** cosas, pero no todo el rato por la consola. En este tema veremos como exportar los datos del programa que nosotros eligamos a un fichero externo.

O por lo contrario, tenemos un fichero, queremos que el programa lo lea y ejecute una serie de instrucciones con el.

Es un flujo de datos **entrante** y **saliente**.

Veremos ficheros de tipo **texto plano** y ficheros de tipo **objeto** — veremos ambos casos.

12) Gestión de bases de datos relacionales

De hecho tenemos una asignatura llamada **Bases de Datos** (Victor Bullido) — ahi aprendemos lo que es una base de datos relacional (SQL).

Y nosotros hasta este momento tenemos la información en nuestro programa, pero en este tema veremos como interaccionar con esas bases de datos, y manipular la información que hay.

Que es más cercano al mundo real. No tan de “laboratorio” o academico.

13) Interfaz Gráfica de Usuario

Es díficil que lleguemos aquí, pero seran las **interfaces gráficas de usuario.** En vez de trabajar todo por consola, tendremos un programa con UI más tradicional.

Si no llegamos a este punto no debemos preocuparnos, porque el profesor (Borja) nos dara información acerca de como se hacen programas con UI.

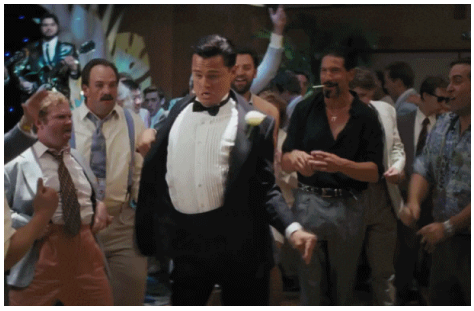
Utilizaremos una libreria Open-Source de Java para crear interfaces gráficas:

> JavaFX
>
> JavaFX runtime is available as a platform-specific SDK, as a number of jmods, and as a set of artifacts in Maven Central.
>
> https://openjfx.io/

Frase epica de Borja

> *"En la vida lo único que NO se negocia (en serio) son dos cosas: uno son las ganas, y dos el intentarlo"*

Si tienes esas dos cosas (las ganas y la determinación de intentarlo) te lo sacas. Quizá peor o mejor pero te lo sacas.



Un grande.

Breve introducción a la programación

👨‍💻 Vamos a ver un poco que es esto de la programación muy por encima.

Programar:

- Escribir intrucciones que lee la máquina para realizar tareas.

Indicar a la máquina mediante una serie de líneas de código tareas / acciones que tiene que realizar, para llegar a un resultado concreto.

No vamos a utilizar lenguaje natural (el castellano) para comunicarnos con la máquina. La máquina habla en **código binario** (0s y 1s). Electricidad, puertas lógicas, transistores...

> 👉 Existen de hecho lenguajes de bajo nível, como Assembly...
> También conocidos como ***lenguajes máquina***

Pero nosotros, que no queremos crear chips sino webs y aplicaciones, no vamos a escribir en lenguajes de bajo nível.

Sino que programaremos en **lenguajes de alto nível**. Que es muy parecido al lenguaje natural que utilizamos en nuestro día a día.

Aunque se asemeje NO es 100% natural obviamente, sino que tenemos que seguir unas pautas.

El ordenador no tiene razonamiento, es más tonto que Dory de la pelicula Nemo. Lo bueno que tiene es que sigue INSTRUCCIONES, tiene una capacidad de procesamiento alucinante.

El ordenador puede procesar cientos de miles de ejecuciones al segundo. Un humano apenas puede hacer 2 cosas a la vez.

Una máquina puede hacer la *tira de cosas* al MICROSEGUNDO. Pero es muy tonto. No sabe que hacer, alguien tiene que decirselo (los programadores).

Las cosas que le digo a la máquina, se las tengo que dar MUY MUY MUY ***detalladas***. Porque no intuye que tiene que hacer... No se para a pensar.

Se lo tenems que indicar de forma muy específica y detallada.

Esto es a lo que llamamos ALGORITMOS:

Un algoritmo (lo veremos más en fundamentos de programación) → es una consecución de ejecuciones con unas entradas que las procesa, y devuelve una salida.

La máquina habla de bajo nível, en 0s y 1s (electricidad y circuitos) pero nosotros en esta asignatura utilizaremos un lenguaje de alto nível (llamado JAVA).

Que es lo que necesitamos para programar en un lenguaje de alto nível? → Pues depende del lenguaje de programación con el que te quieras pegar.

Hay infinidad de lenguajes (Python, PHP, Golang.....) hay muchísimos, y cada día salen nuevas variantes. Pero si que es verdad que existen los **principales.**

En nuestra vída profesional, núnca seremos expertos en todos. Como máximo de 3. Y ser experto no quiere decir "lo se todo, soy un Guru".

Pero no hay persona en el mundo que se sepa todos y de memoria la sintaixs.

Nuestra solución:

Ser experto en uno, dos o tres lenguajes y conocerlos en profundidad, y lo que es la base de programar tenerla clara y una vez tengamos eso claro, no nos va a costar casí nada cambiarnos de lenguaje.

> 🚗 **Una vez aprendes a conducir, puedes conducir otros coches con más fácilidad, pero tienes que empezar con uno.**
> Y si sabes conducir un coche, luego aprender a conducir un autocar es más fácil.

No vamos a saberlo todo NUNCA — vamos a seguir aprendiendo toda la vida, y nos tenemos que mantener como novatos permanentemente.

Existen principales lenguajes, nosotros trabajaremos con JAVA — hay años en los que esta como el primero más importante e utilizado.

Gran parte de los programas y software que existe de hoy en día, están hechos en Java. Es muy fiable, muy seguro y sobretodo *multiplataforma.*

Porque tiene una cosa llamada JVM (Java-Virtual-Machine) que permite a los programas ejecutarse en cualquier dispositivo que tenga la JVM instalada. Lo que hace es independizar el lenguaje de programación del hardware de la máquina. Le da igual que sea una nevera, un reloj, un ordenador, un móvil.

Además que al ser más restrictivo nos dara un salto para luego aprender otros lenguajes más sencillos.

Es uno de los lenguajes que se suelen utilizar más para aprender.

También hay otros lenguajes como **Python**, que de hecho este año es el más popular. Esta muy de moda porque es más fácil la sintaxis, no hay que compilarlo, simplemente ejecutarlo, y se utiliza mucho para IA y Data Science.

Y otro que tenemos es **JavaScript** — que es *muy orientado o totalmente* orientado a web.

Aquí hay un enlace con el ranking de los lenguajes más vigentes a día de hoy:

> TIOBE Index - TIOBE
>
> Let op! Internet Explorer wordt niet meer ondersteund. Hierdoor kan de website mogelijk niet goed functioneren, gebruik een alternatieve browser om optimaal gebruik te maken van deze website. Klik hier om een alternatieve browser te downloaden.
>
> https://www.tiobe.com/tiobe-index/

Hay un lenguaje mejor que otro? No.

Que depende en que lenguaje utilice en un proyecto? → Depende del cometido del proyecto. Elegiremos un lenguaje u otro.

No existe un lenguaje de programación universal. Utilizaremos el lenguaje dependiendo de que queramos hacer.

Nosotros utilizaremos JAVA para está asignatura y para adentrarnos en el mundillo de la programación.

Y que tiene JAVA en particular? → pues que es un **lenguaje compilado.**

JAVA es similar al lenguaje natural... pero... no hemos dicho que la máquina habla en 1s y 0s? Si. Pero lo que pasa es que el lenguaje se compila y devuelve un ejecutable y lo ejecuta la JVM (Java-Virtual-Machine) → escribimos nuestro código, lo compilamos, le da el ejecutable a la JVM.

Da igual si estas en Linux, Windows, MacOS, Android...

Esto es lo bueno de JAVA.

Que necesitamos para programar en JAVA?

1. **JAVA** (xd) → como lo descargamos? Pues vamos a su portal de ORACLE (la empresa que mantiene JAVA) y nos descargamos el JDK.

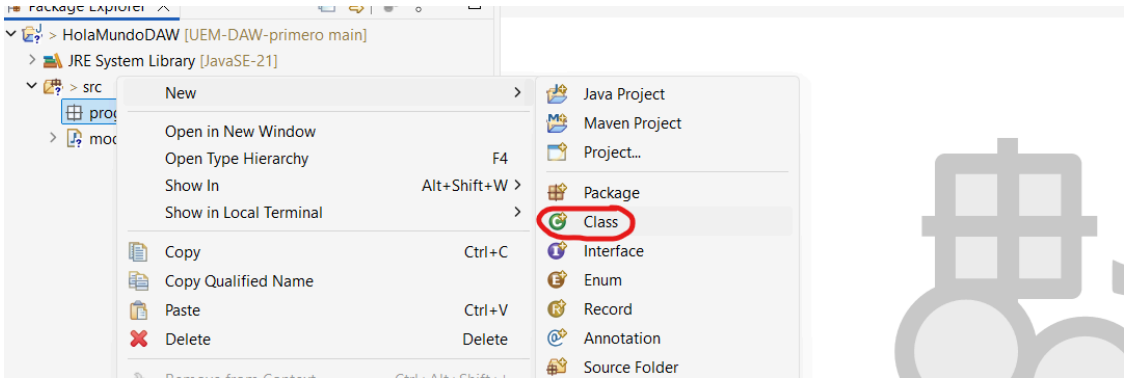
2. **Un IDE** (entorno de desarrollo) → es como un bloc de notas pero con esteroides para escribir nuestros programas.

Yo utilizare ECLIPSE pero el profesor recomienda utilizar INTELIJ IDEA.

Resolución de ejercicio

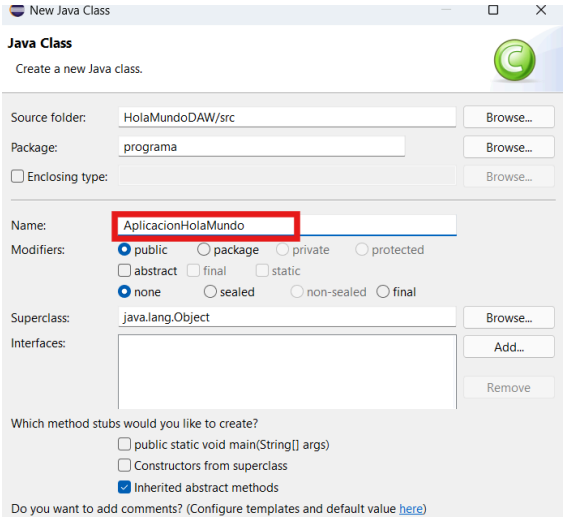
Vamos a crear un **Hola Mundo** en JAVA.

Lo primero que tenemos que hacer es **crear una clase:**

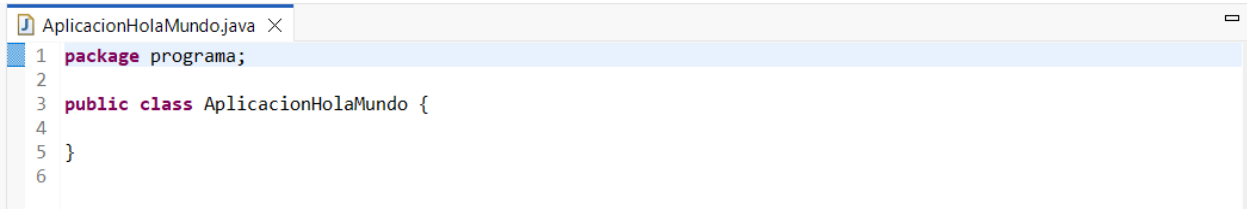


Atención: yo estoy utilizando el IDE de Eclipse, pero si eres otro alumno viendo mis apuntes tendrás que hacerlo en tu IDE correspondiente. Aunque el código es el mismo.

Yo voy a llamar a mi clase **AplicaciónHolaMundo:**



Nos encontramos con este código inicial:



Lo que tenemos que hacer, aunque aún no lo entendamos muy bien, es implementar un método main. El método main para que me entendais, es como el **punto de entrada del programa**, es decir lo que JAVA va a mirar para ejecutar.

Digamos que es como la puerta principal de tu casa por la que tiene que entrar y salir todo el mundo siempre. No hay otra.

Y el método main se escribe así:

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

}
```

No hace falta entender ahora mismo que significa cada palabra (public, void, String...) pero es una sintaxis que memorizaras con el tiempo y se te quedara despues de repetirla mil veces, y eventualmente cuando pasen unos meses y hayamos cubierto otros temas (como modificadores de acceso, tipos de datos etc... será más claro).

Así queda en el programa:

```
package programa;

public class AplicacionHolaMundo {

    public static void main(String[] args) {

    }
}
```

Si os fijais esta ***dentro*** de los corchetes que se abren y cierran en `public class AplicacionHolaMundo`.

Dentro de el, le indicaremos a JAVA la siguiente instrucción:

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

El Hola Mundo suele ser el primer programa que se crea para enseñar al alumno a como escribir en consola. Ahora se vera más claro.

Podriamos decir que **System.out.println** son “”carpetas”” (entre comillas porque no son carpetas) que contienen un conjunto de instrucciones.

Y dentro de las comillas dobles `""` ponemos el texto que queremos que se muestre en pantalla.

Y al final de la instrucción, tenemos que poner un punto y coma `;` para indicar que ahi termina la instrucción, esto en otros lenguajes como Python no es necesario, pero en JAVA es obligatorio. Aunque veremos que hay excepciones (bloques de código, estructuras de control etc.)

Nuestro programa quedaría así:

```
package programa;

public class AplicacionHolaMundo {

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

Y si compilamos el programa, dandole al bóton de **RUN** de tu IDE, saldrá este texto en consola:

```
Hola mundo
```

Felicidades, has creado tu primer programa de muchos. Bienvenido al club de la lucha!

Pero oye, porque no pruebas experimentar con lo que has aprendido?

Por ejemplo como he hecho yo:

```java
package programa;

public class AplicacionHolaMundo {

    public static void main(String[] args) {
        System.out.println("Vamos a aprobar DAW todos");
        System.out.println("A darle mucha cañaaa joder");
        System.out.println(":)");
    }
}
```

Que crees que saldra en consola cuando ejecutemos esto?

▼ Piensalo y luego haz click aquí para verlo

```
Vamos a aprobar DAW todos
A darle mucha cañaaa joder
:)
```