UML

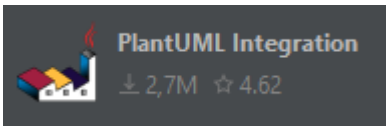
Curso	1W DAW
Alumno	Miguel Pozo Pérez
Profesor/a	Jaime García Quiles

Práctica

Repositorio GitHub: https://github.com/M-pozo/UMLClases_MiguelPozo

Primero instalamos un plugin llamado PlantUml en IntelliJ que nos ayudará a realizar los esquemas UML, Para ello vamos en IntelliJ a: **FIle > Editor > Plugins**

Buscamos en el buscador **PlantUMML Integration** y lo instalamos.



Creamos dos carpetas (Proyecto y UML) en **src** para tener un poco de orden, ahora hacemos clic derecho en la carpeta **UML > New > PlantUML File.**

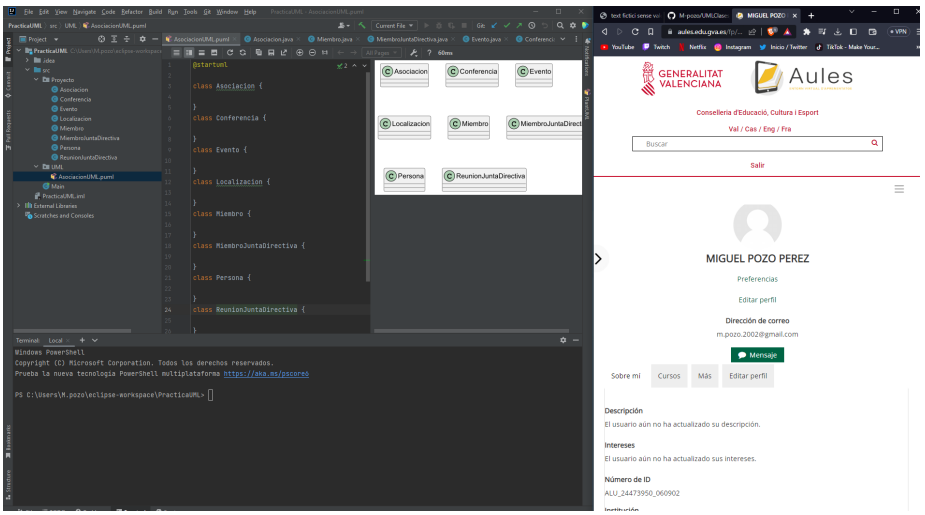
Una vez ya hemos hecho lo fácil vamos a comenzar a hacer diagramas para ayudarnos en nuestros proyectos, cuando tengamos claro qué programa queremos hacer los primero de todo va a ser identificar las clases del programa.

En nuestro programa actualmente hemos decidido implementar estas clases.
Asociación
Miembro
MienbroJuntaDirectiva
Evento
Conferencia
ReunionJuntaDirectiva
Localización
Persona

En UML las vamos a implementar con la siguiente sintaxis.

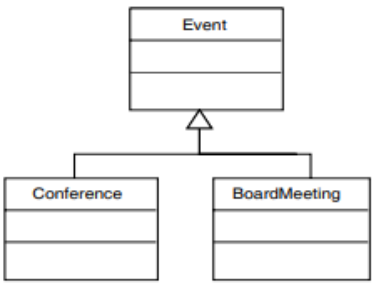
```
class NombreClase {

}
```

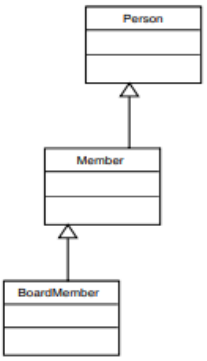
Para hacer el diagrama se dibuja un rectángulo dividido en otros tres rectángulos horizontalmente, en el de arriba se escribe el nombre de la clase, en el del centro las variables y en el de abajo los métodos.



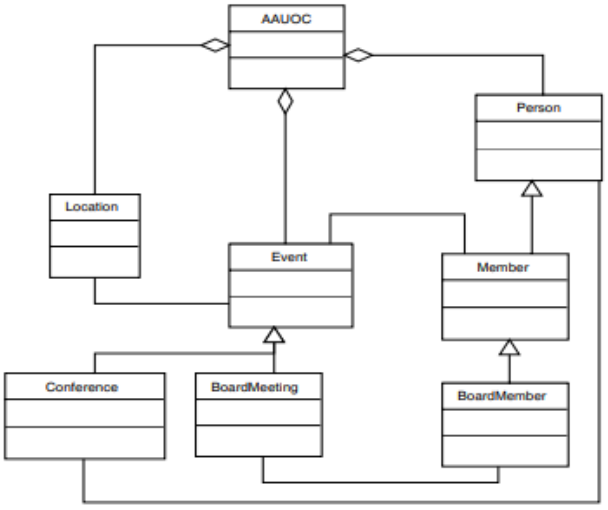
A continuación creamos el modelo de datos, como podemos observar hay una jerarquía de herencia entonces vamos a agrupar las clases para ir pillando la forma del diagrama final.

Tenemos las siguientes jerarquías:





Que iremos uniendo para desarrollar el diagrama final



Ahora vamos a ver como hacer las relaciones entre clases en el plugin de UML con la siguiente sintaxis;

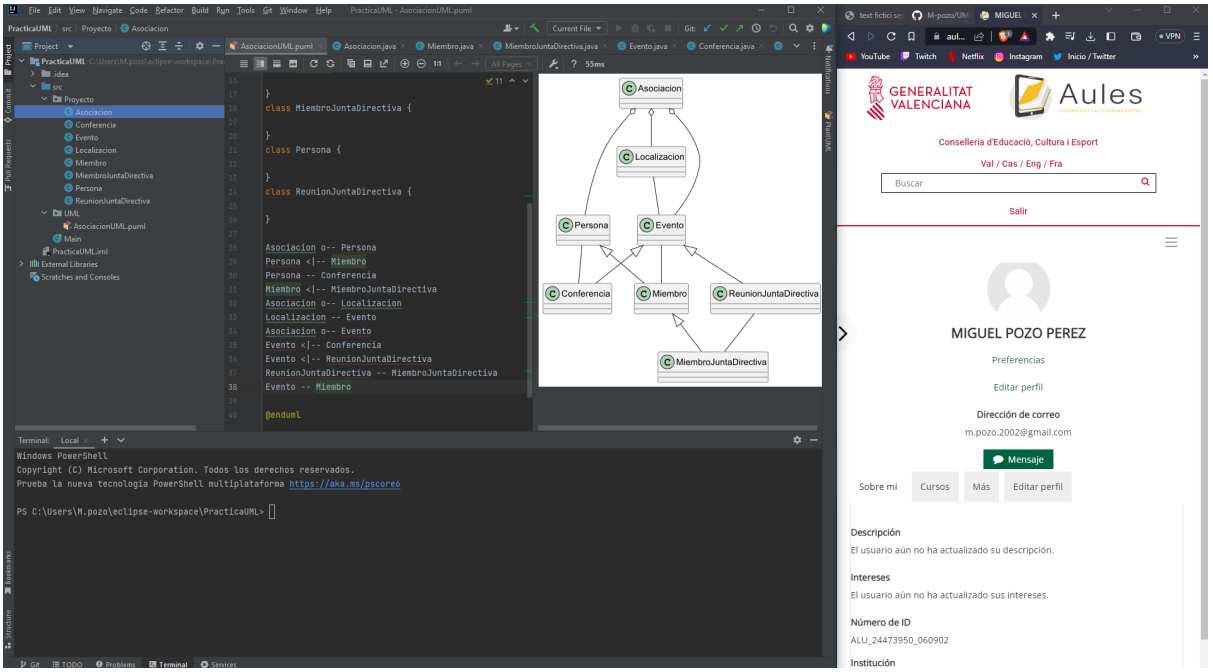
Clase “flechas” Clase Ejemplo Persona --> Miembro

Flechas:

```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```

```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```

```
@startuml
Class21 #-- Class22
Class23 x-- Class24
Class25 }-- Class26
Class27 +-- Class28
Class29 ^-- Class30
@enduml
```



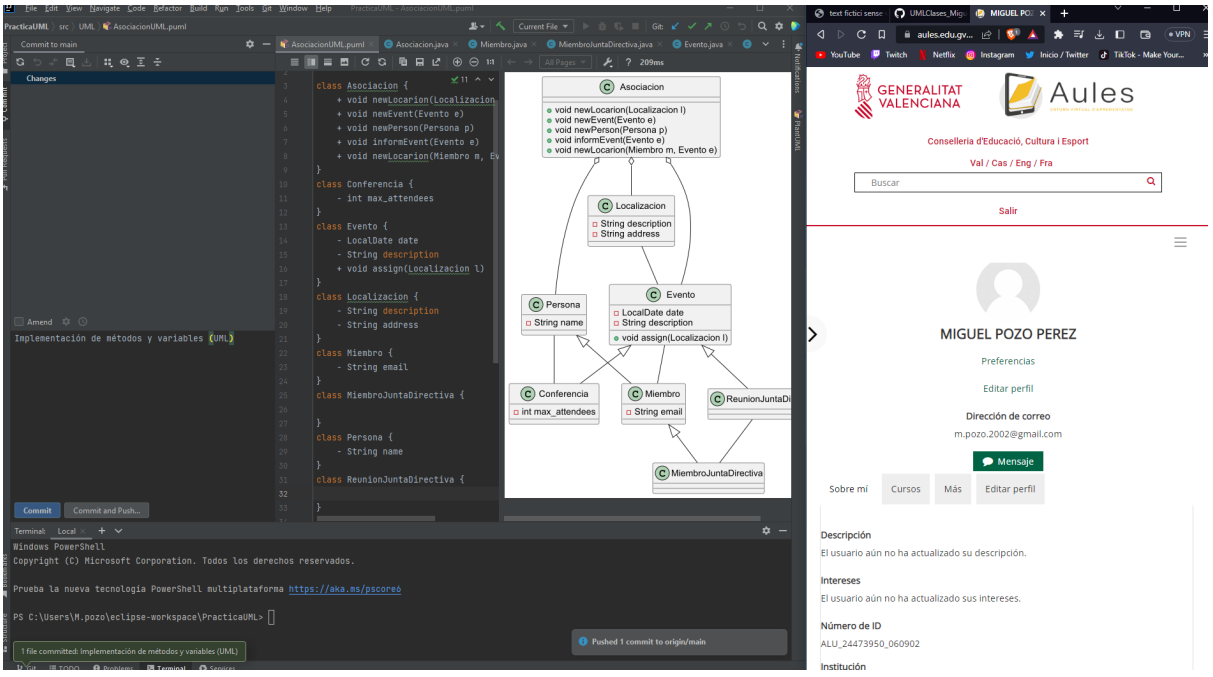
A continuación vamos a implementar los atributos y métodos para complementar las clases con los métodos más relevantes (obviamos getters, setters y constructores).

Para agregar atributos y métodos utilizaremos la siguiente sintaxis:

```
class NombreClase {

    Símbolo TipoVariable NombreVariable (Para variables) //borrar

    Símbolo TipoVariable NombreVariable (Para variables) //borrar

}
```

Símbolo:

Character	Icon for field	Icon for method	Visibility
-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public



Por último vamos a implementar las cardinalidades y etiquetas

Para ello utilizaremos la siguiente sintaxis:

Clase “CARDINALIDAD” flechas “CARDINALIDAD” : ETIQUETA

Ejemplo:

```
Localizacion "1" -- "0..*" Evento : isLocatedIn
```