

# Creando PlantUML en IntelliJ

## Enunciado:

La Asociación de Antiguos Alumnos de la UOC nos ha pedido si podemos ayudarles a confeccionar un programa que les permita gestionar a sus asociados, eventos y demás elementos relacionados.

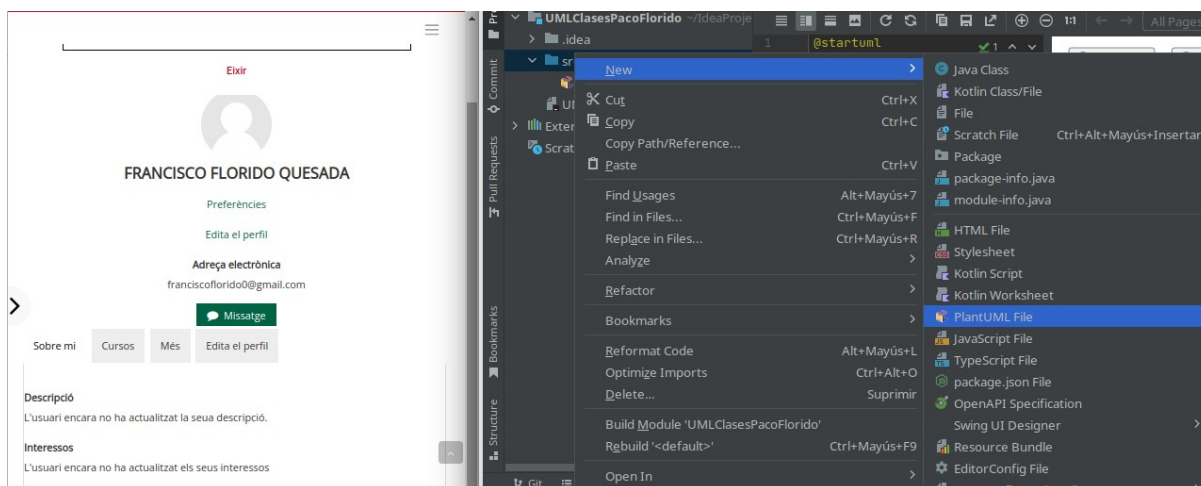
Los asociados se pueden dividir en miembros numerarios y en miembros de la junta directiva, que es elegida por votación en una asamblea general cada cuatro años. La única diferencia entre ellos es que los miembros de la junta directiva son convocados a las reuniones de junta y los demás miembros no, pero el resto de actividades que se realizan están abiertas a todos los miembros de la asociación.

La convocatoria de un evento se realiza por correo electrónico a todos los miembros activos en el momento del envío, recibiendo un enlace para aceptar su participación. En todos los eventos, la aceptación de los asistentes se realiza por orden de llegada ya que, en algún caso, se puede dar que el número de asistentes sea limitado, como en las conferencias.

En la convocatoria, también aparece información sobre el lugar que en muchos casos se repite, por lo que nos han dicho que quieren almacenar los datos para futuros usos.

## Solución:

Empezaremos creando un nuevo proyecto en Java, después, crearemos el archivo PlantUML que vemos en la siguiente imagen.



Una vez creamos el archivo, procedemos a crear las clases que necesitaremos para este programa.

```
class Member

class BoardMember

class Event

class Conference

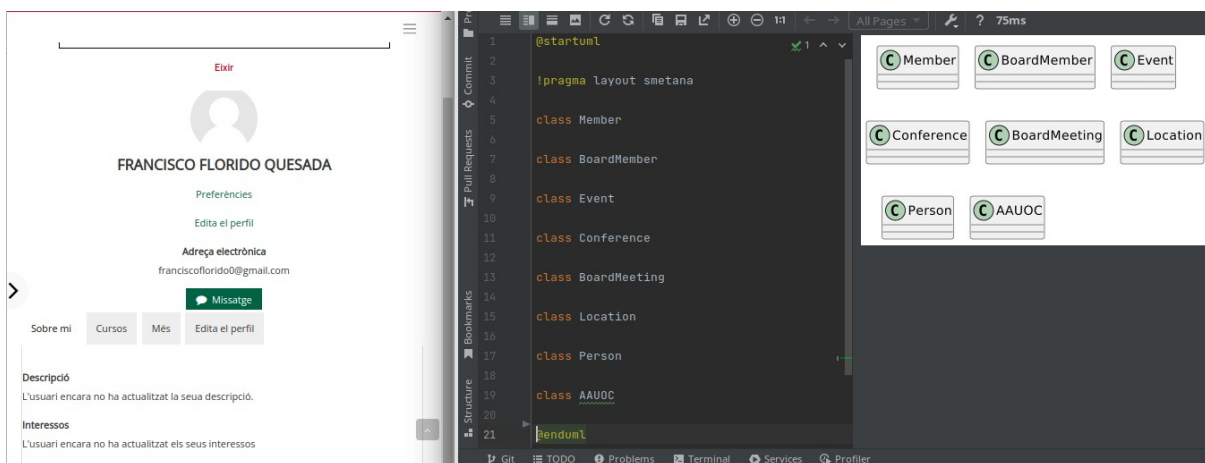
class BoardMeeting

class Location

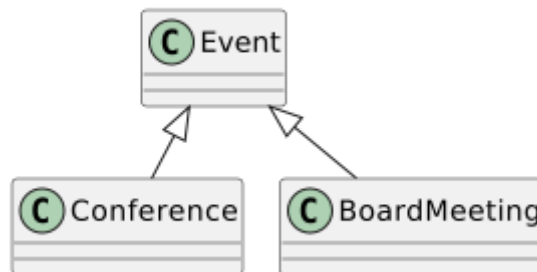
class Person

class AAUOC
```

Para que funcione lo que nos crea las clases automáticamente en UML tenemos que poner la línea de código que tenemos justo debajo del @startuml.

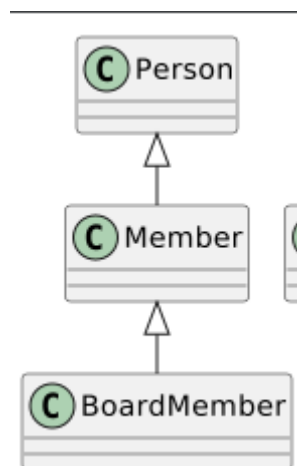


Ahora, vamos a empezar por las relaciones entre clases. En primer lugar, según el problema propuesto, la clase “Event” será una superclase de dos subclases que heredan de ella, estas subclases son la clase “Conference” y la clase “BoardMeeting”.



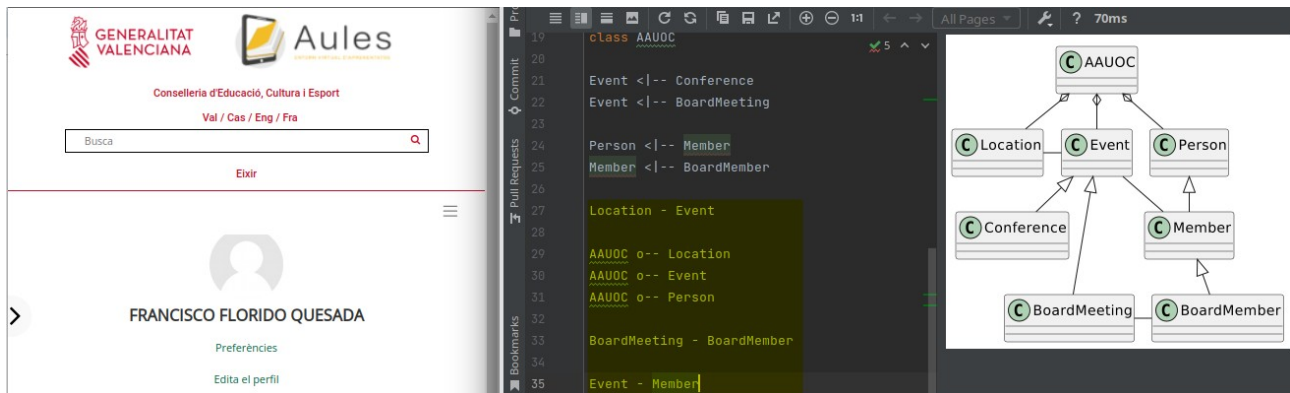
```
Event <|-- Conference
Event <|-- BoardMeeting
```

Las clases “Person”, “Member” y “BoardMember” también estarán relacionadas.

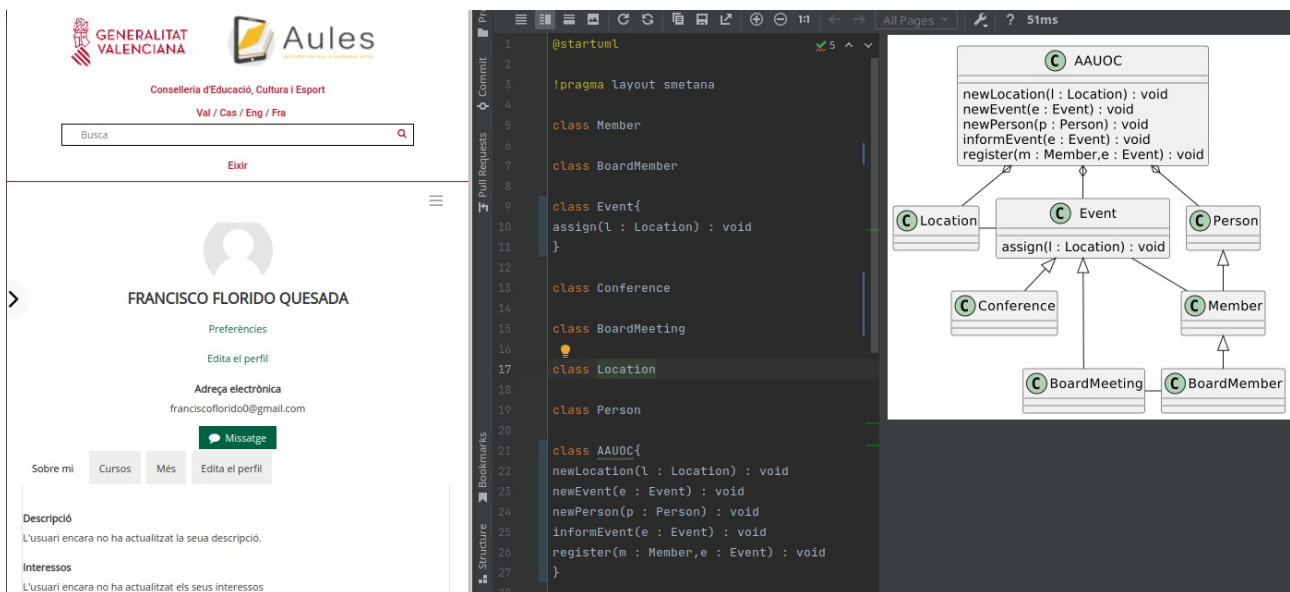


```
Person <|-- Member
Member <|-- BoardMember
```

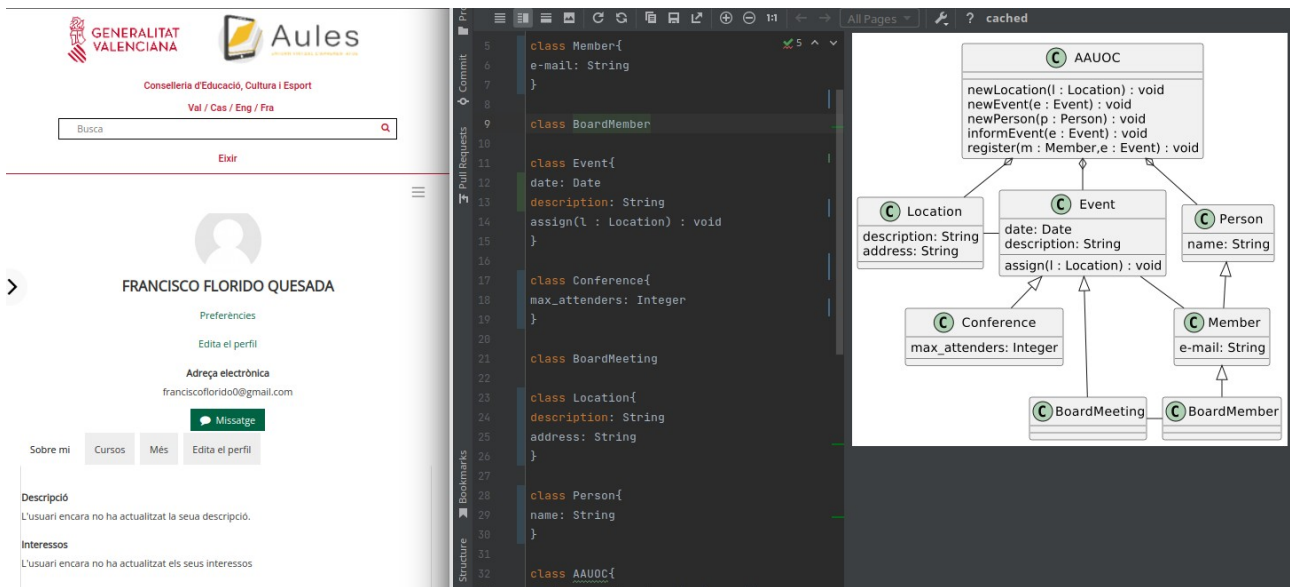
Además, tenemos las clases Localización (Location) y Asociación (AAUOC), que se relacionan con el resto de clases del siguiente modo:



Una vez creadas las relaciones pasamos con los atributos y los métodos. La clase AAUOC necesitará un conjunto de métodos para añadir nuevos eventos, personas y localizaciones al sistema (métodos *newX*), así como un método que permita informar a los miembros de la convocatoria de un evento (método *informEvent*). Además los miembros necesitarán confirmar su asistencia (método *register*).



También añadiremos los atributos a todas las demás clases.



The screenshot shows the Aules web application interface on the left, the code editor in the middle, and the class diagram on the right.

**Web Application Interface:** The header includes the 'GENERALITAT VALENCIANA' logo and 'Aules' branding. A search bar and language selector (Val / Cas / Eng / Fra) are present. The user profile for 'FRANCISCO FLORIDO QUESADA' is displayed with options for 'Preferències', 'Edita el perfil', 'Adreça electrònica', and 'Missatge'. Navigation tabs for 'Sobre mi', 'Cursos', 'Més', and 'Edita el perfil' are at the bottom.

**Code Editor:** The code defines several classes:
 

```

class Member{
    e-mail: String
}

class BoardMember

class Event{
    date: Date
    description: String
    assign(l : Location) : void
}

class Conference{
    max_attenders: Integer
}

class BoardMeeting

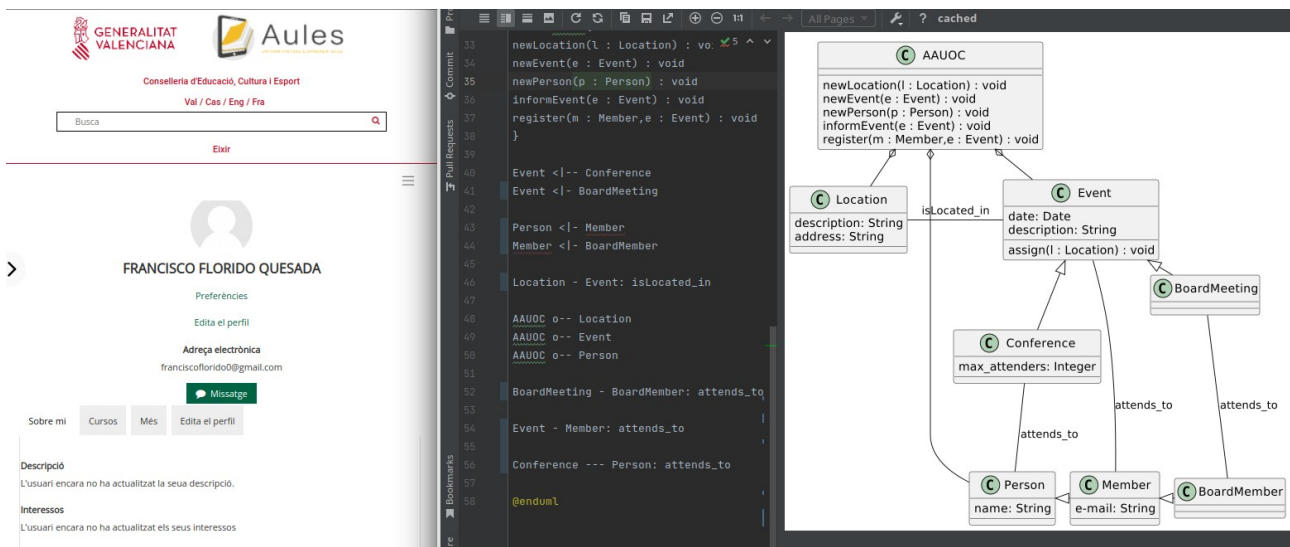
class Location{
    description: String
    address: String
}

class Person{
    name: String
}

class AAUOC{
    
```

**Class Diagram:** The diagram shows the relationships between the classes. AAUOC is the root class. It has associations with Location, Event, and Person. Location has an association with Event (labeled 'isLocated\_in'). Event has an association with BoardMeeting (labeled 'assign(l : Location) : void'). BoardMeeting has an association with BoardMember (labeled 'attends\_to'). BoardMember has an association with Member (labeled 'attends\_to'). Member has an association with Person (labeled 'attends\_to').

Ahora añadiremos una pequeña descripción de lo que hacen las relaciones.



The screenshot shows the Aules web application interface on the left, the code editor in the middle, and the updated class diagram on the right.

**Web Application Interface:** The interface is identical to the previous screenshot, showing the user profile for 'FRANCISCO FLORIDO QUESADA'.

**Code Editor:** The code now includes comments for the relationships:
 

```

newLocation(l : Location) : void
newEvent(e : Event) : void
newPerson(p : Person) : void
informEvent(e : Event) : void
register(m : Member,e : Event) : void

Event <|-- Conference
Event <|-- BoardMeeting

Person <|-- Member
Member <|-- BoardMember

Location - Event: isLocated_in

AAUOC o-- Location
AAUOC o-- Event
AAUOC o-- Person

BoardMeeting - BoardMember: attends_to

Event - Member: attends_to

Conference --- Person: attends_to

@enduml
    
```

**Class Diagram:** The diagram is updated to reflect the new relationships. AAUOC is associated with Location, Event, and Person. Location is associated with Event (labeled 'isLocated\_in'). Event is associated with BoardMeeting (labeled 'assign(l : Location) : void'). BoardMeeting is associated with BoardMember (labeled 'attends\_to'). BoardMember is associated with Member (labeled 'attends\_to'). Member is associated with Person (labeled 'attends\_to').

Y por último incluiremos la cardinalidad de las relaciones entre clases. Para esto quitaré los métodos y atributos para poder apreciar mejor la cardinalidad.

The screenshot shows a web application on the left and a code editor on the right. The web application is for 'Aules' (Generalitat Valenciana) and displays the profile of Francisco Florido Quesada. The code editor shows the following class definitions:

```

class AAUOC{
}
Event <|-- Conference
Event <|-- BoardMeeting
Person <|-- Member
Member <|-- BoardMember

Location "1" - "0..*"Event: isLocated_in

AAUOC o-- "0..*"Location
AAUOC o-- "0..*"Event
AAUOC o-- "0..*"Person

BoardMeeting"0..*" - "0..*"BoardMember: attends_to

Event "0..*" - "0..*"Member: attends_to

Conference"0..*" --- "0..*"Person: attends_to

@enduml
    
```

On the right, a UML diagram illustrates the relationships between these classes. AAUOC is associated with Location (0..\* to 1), Event (0..\* to 0..\*), and Person (0..\* to 0..\*). Location is associated with Event (1 to 0..\*) via 'isLocated\_in'. Event is a base class for Conference and BoardMeeting. BoardMeeting is associated with BoardMember (0..\* to 0..\*) via 'attends\_to'. Event is associated with Member (0..\* to 0..\*) via 'attends\_to'. Conference is associated with Person (0..\* to 0..\*) via 'attends\_to'. Member is a base class for BoardMember.

El diagrama completo quedaría así:

