Práctica Diagramas UML Clases

Daniel Vital Torres

DANIEL VITAL TORRES

Preferencias

Editar perfil

Dirección de correo

deadblood404@hotmail.com

Ciudad

Elche

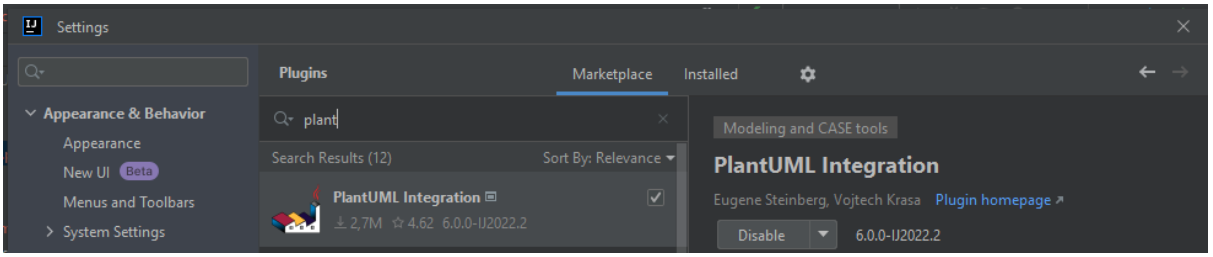
País

ES

Mensaje

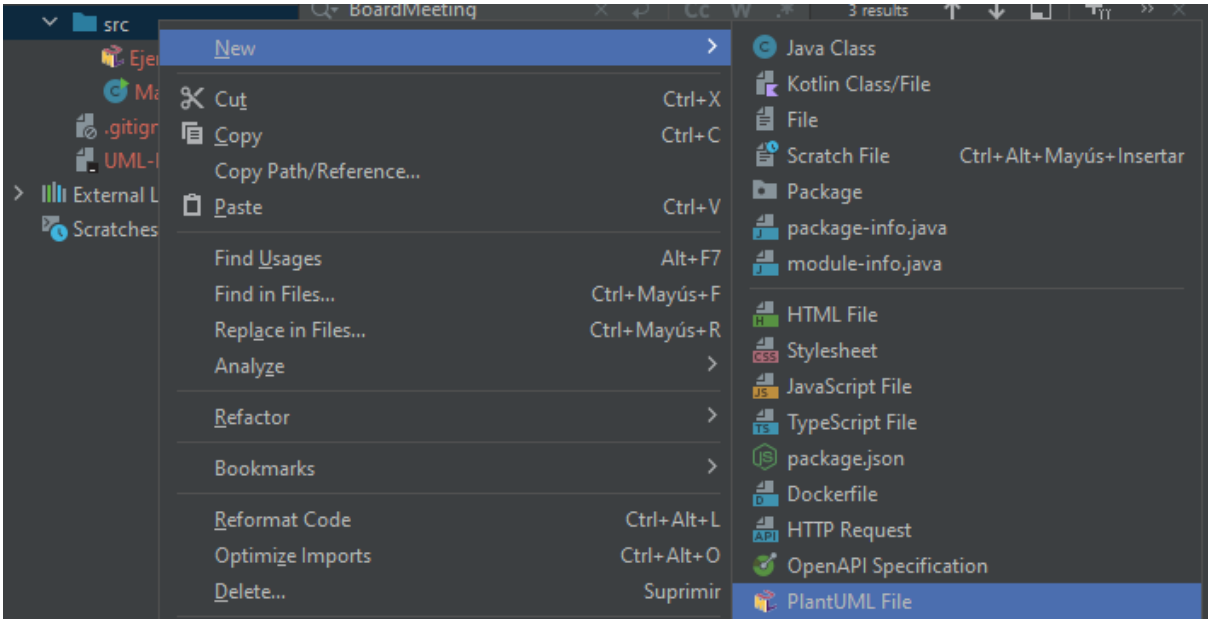
En esta práctica lo que haremos será seguir crear un diagrama UML con IntelliJ usando el plugin PlantUML siguiendo unos pasos concretos.

Lo primero que haremos será instalar, en IntelliJ, el plugin PlantUML. Muy sencillo.
Hay que acceder a File -> Settings -> Plugins -> Marketplace -> PlantUML.



Lo instalamos y pedirá reiniciar el IDE. Aceptamos y ya lo tenemos instalado.

Después de eso creamos un proyecto en IntelliJ, y en la carpeta src, creamos un fichero UML



Lo llamamos como veamos oportuno y ya tendríamos el fichero preparado.

[Manual de PlantUML]

Lo siguiente que tenemos que hacer es identificar, del enunciado que se nos presenta, las clases que va a haber.

> 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.

De este texto podemos sacar que las clases que necesitaremos son:

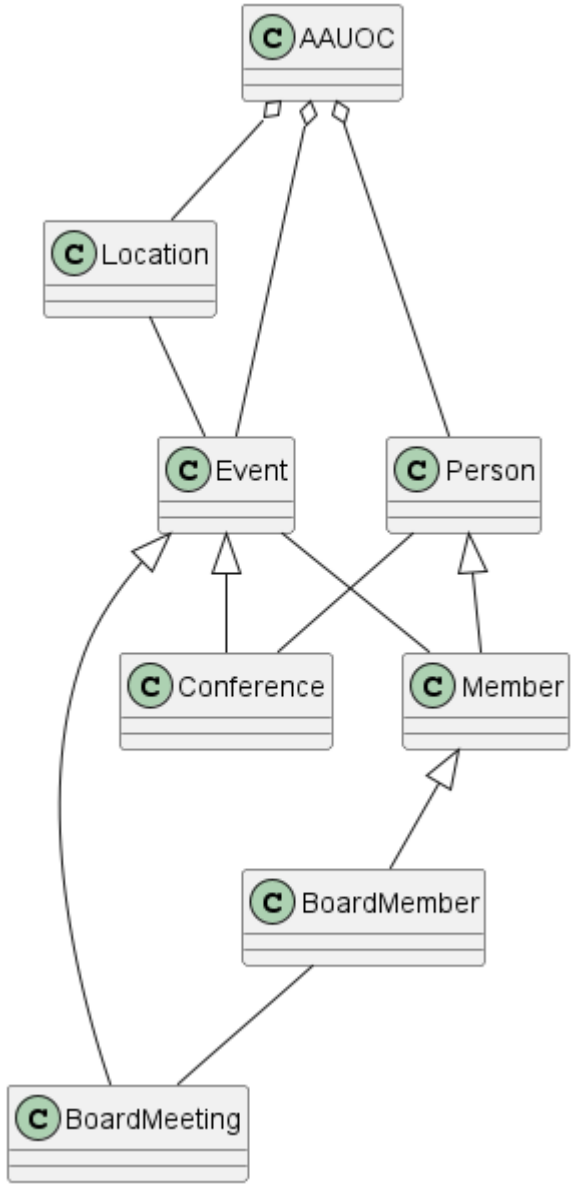
- Miembro (Member)
- Miembro de junta directiva (BoardMember)
- Evento (Event)
- Conferencia (Conference)
- Reunión de junta directiva (BoardMeeting)
- Localización (Location)
- Persona (Person)

y por otra parte la clase de la Asociación (AAUOC).

Lo siguiente que haremos será crear el modelo de datos. Se ha detectado, en el texto, una jerarquía de herencia que tiene como punto principal el evento. Básicamente todas las clases parten del evento, y estas clases, tienen subclases por debajo.

Si vamos hilando y vamos viendo qué clase es subclase de cuál, se nos queda el siguiente diagrama:

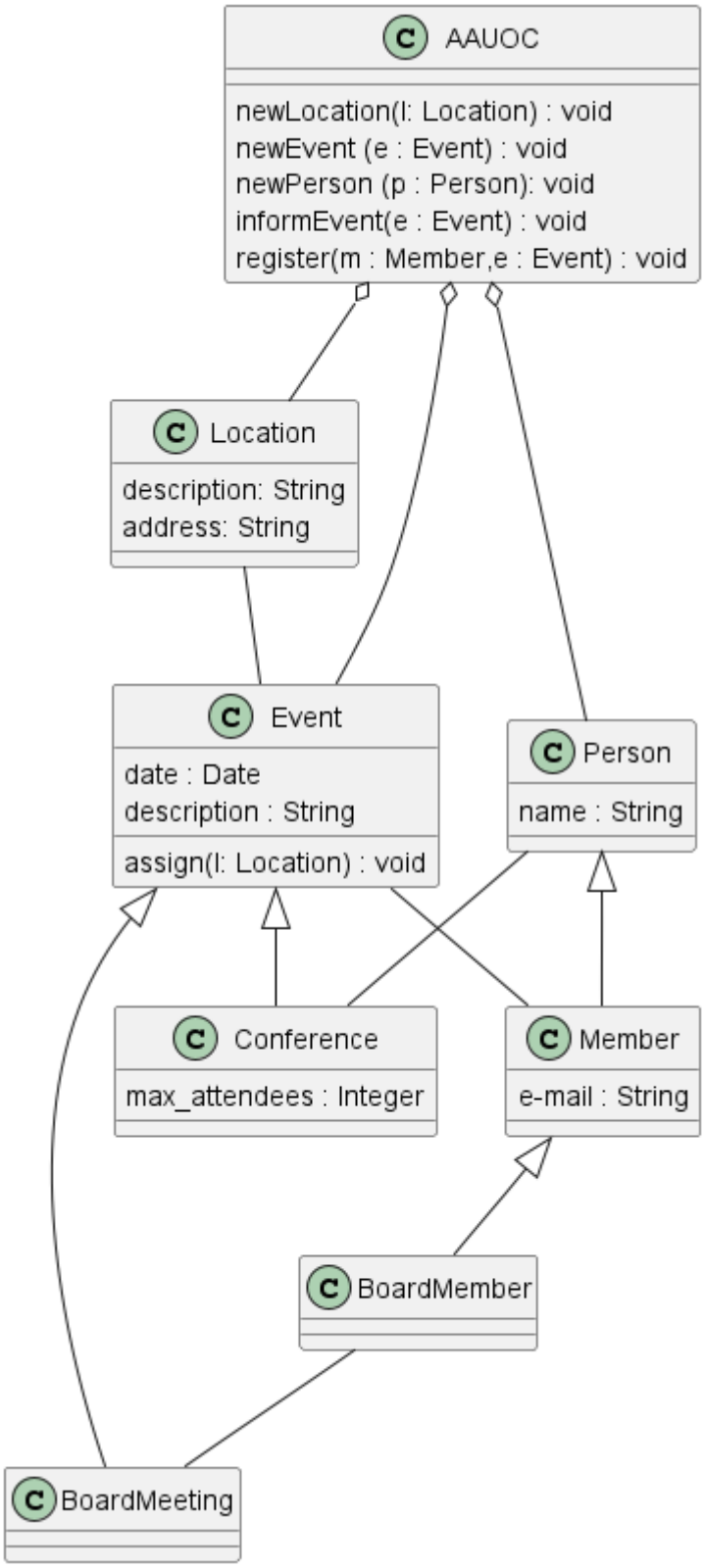
```
@startuml

class AAUOC

class Location
AAUOC o-- Location

class Event
Location -- Event
AAUOC o-- Event

class Person
AAUOC o-- Person

class Member
Person <|-- Member
Event -- Member

class Conference
Event <|-- Conference
Person -- Conference

class BoardMeeting
Event <|-- BoardMeeting

class BoardMember
BoardMember -- BoardMeeting
Member <|-- BoardMember

@enduml
```



Lo siguiente que tenemos que hacer, una vez que tenemos claras las clases y su órden, es la inclusión de atributos y métodos.

```
@startuml

class AAUOC{
newLocation(I: Location) : void
newEvent (e : Event) : void
newPerson (p : Person): void
informEvent(e : Event) : void
register(m : Member,e : Event) : void
}

class Location{
description: String
address: String
}
AAUOC o-- Location



class Event{
date : Date
description : String

assign(I: Location) : void
}
Location -- Event
AAUOC o-- Event

class Person{
name : String
}
AAUOC o-- Person

class Member{
e-mail : String
}
Person <|-- Member
Event -- Member

class Conference{
max_attendees : Integer
}
Event <|-- Conference
Person -- Conference

class BoardMeeting
Event <|-- BoardMeeting

class BoardMember
BoardMember -- BoardMeeting
Member <|-- BoardMember

@enduml
```

El diagrama en este paso se quedaría así:



Ahora haremos las relaciones entre clases. Qué hace qué con qué clase, cardinalidades, etc…

```
@startuml

class AAUOC{
newLocation(I: Location) : void
newEvent (e : Event) : void
newPerson (p : Person): void
informEvent(e : Event) : void
register(m : Member,e : Event) : void
}

class Location{
description: String
address: String
}
AAUOC o-- "0..*" Location

class Event{
date : Date
description : String
}
Location "1" -- "0..*" Event : isLocatedIn
AAUOC o-- "0..*" Event

class Person{
name : String
}
AAUOC o-- "0..*" Person

class Member{
e-mail : String
}
Person <|-- Member
Event "0..*" -- "0..*" Member : attendsTo

class Conference{
max_attendees : Integer
}
Event <|-- Conference
Person "0..*" -- "0..*" Conference : attendsTo

class BoardMeeting
Event <|-- BoardMeeting

class BoardMember
BoardMember "0..*" -- "0..*" BoardMeeting : attendsTo
Member <|-- BoardMember

@enduml
```

El diagrama completo quedaría así: