Pràctica 4: Manual de la GUI

Codi que es proporciona amb la pràctica 4:

Per tal que pugueu dissenyar més ràpidament la pràctica 4, us proporcionem ja una interfície gràfica implementada en Java FX, que podeu modificar al vostre gust. Podeu també fer-la des de zero si aquesta interfície no us convenç, però recordeu que la finalitat de la pràctica 4 és veure el disseny intern i no la interfície gràfica per si mateixa.

El codi base està en el github classroom i podeu clonar-lo al vostre github. Està compilat en la versió 17 de Java.

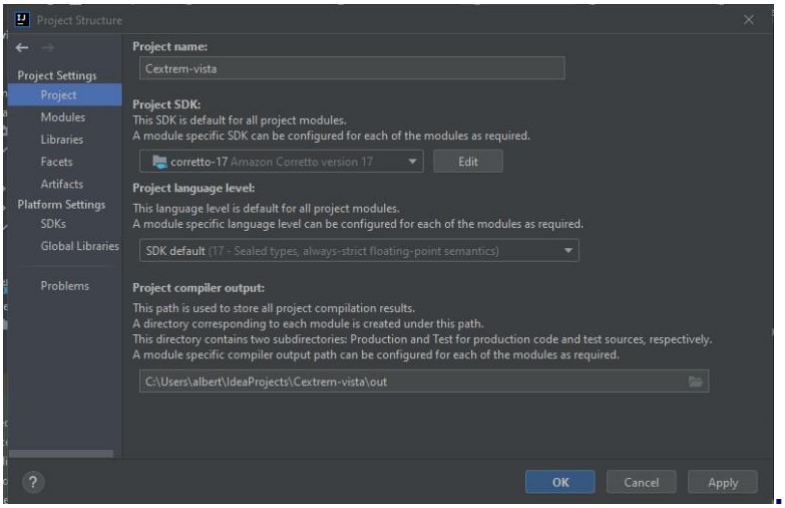
Projecte base (clonat en el github)

1. **Baixa el projecte p4-view-CeXtrem de la classroom de github:**

   https://classroom.github.com/a/1XuQ_XQW

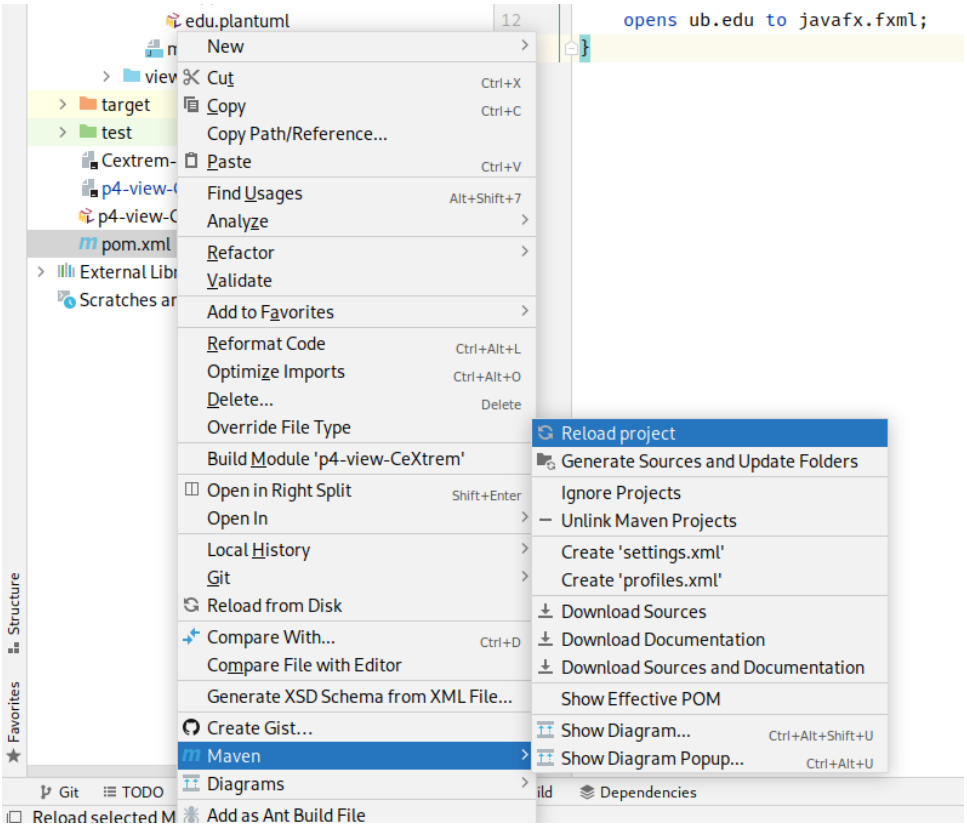
2. **Configura el projecte per a que s'executi la interfície gràfica**

Vigila que la versió de Java ara ha de ser la 17, per a poder compilar amb JavaFX.



3. **Instal·la les dependències**

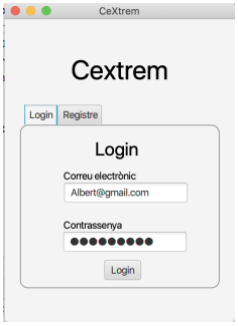
Per a instal·lar les dependències del projecte el primer cop que l'executes, fes clic amb el boto dret del ratolí sobre el arxiu pom.xml, obre el menú Maven, i selecciona la opció de recarregar projecte.



4. Executa el projecte

p4-view-CeXtrem de la pràctica 4 està formada per un projecte que conté una classe main: AppMain que crea directament la vista, sense tenir ni controlador ni model.

Quan executeu el codi, surt una primera finestra per entrar un nom de client:



En aquesta versió buida, podeu entrar amb qualsevol client i contrasenya i us deixarà entrar a l'aplicació general. La finestra principal (formulari **CeXtrem-view**) té el següent aspecte:

Per ara hem posat dades a les llistes de les excursions a mode d'exemple de com omplir les llistes de la interfície. Fixeu-vos que hi ha llistes que es poden carregar en la vista des de l'inici, i d'altres que cal anar actualitzant a mesura que es valoren o es paguen les activitats

En aquesta finestra ja podeu veure les llistes que us comentava l'enunciat:

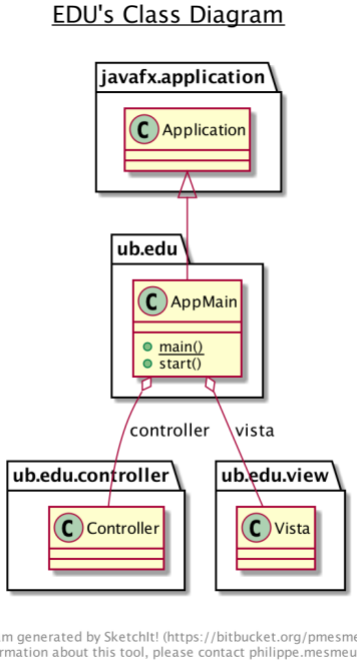
1. La Llista de les excursions disponibles que es poden llistar ordenades per nom o per data.
2. Vista del Top 10 de les activitats a les què s'han apuntat més socis.
3. Vista del Top 10 de les activitats més valorades (es pot seleccionar per estrelles o per likes/unlikes) o bé les Top 10 de les activitats que han tingut més socis que les han pagades (es pot seleccionar pel tipus de pagament), segons sigueu del grup de dimarts o de dijous.

Quan es selecciona una activitat, la interfície ja porta a la finestra on es mostren els seus detalls i els botons de Apuntar-se, Pagar i Valorar.

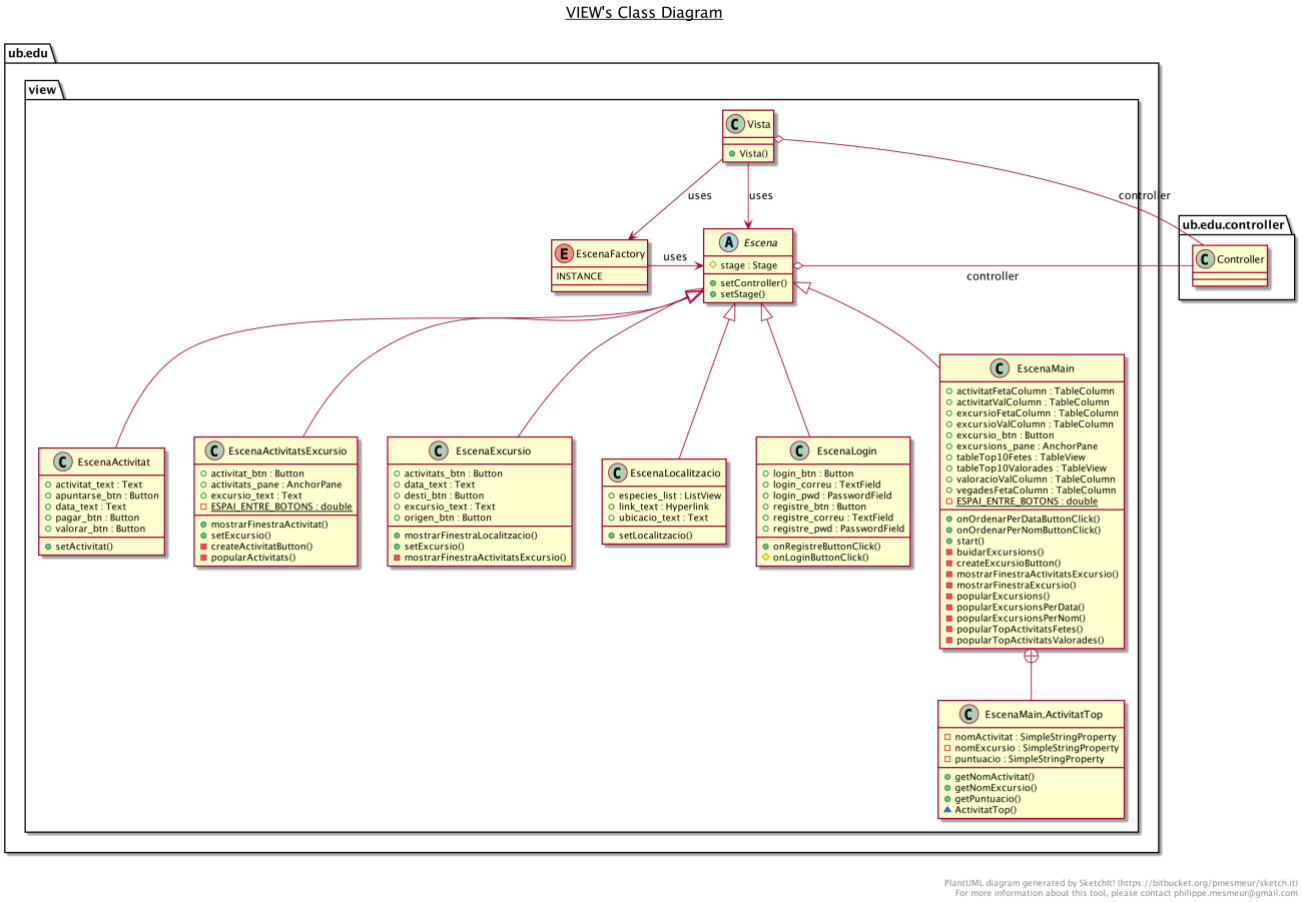
Recordeu que en la finestra principal de l'aplicació, la part de 10+Apuntades (i les 10+Valorades o les 10+Pagades, si la implementeu) **sempre** estan visibles en la pantalla de l'aplicació, actualitzant-se quan hi hagi canvis en les valoracions o en les visualitzacions (ja sigui de valor o d'ordre), sense necessitat que l'usuari provoqui el refresc, sinó just en el moment que es detecti un canvi en les dades.

5. Explora les classes de la carpeta view per a veure com es programa la interfície gràfica en JavaFX

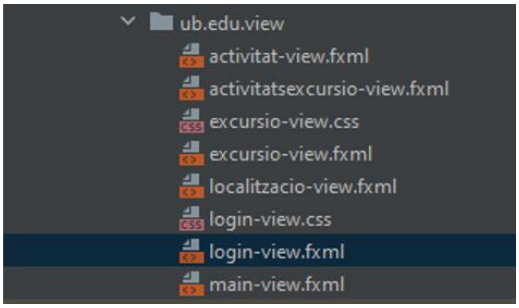
El teu main es troba a la classe AppMain que deriva de la classe Application de la llibreria de JavaFX. Explora el main del projecte i distribueix les responsabilitats de crear el Controlador, el Model i la Vista segons convingui. El codi que se't dóna és només una versió per a que funcioni. Proposa qui hauria de cridar o crear a qui...



El paquet de la vista està organitzat segons el següent diagrama de classes. Executa el projecte per veure a partir de quina finestra es creen les finestres que van apareixen a l'execució. En el següent diagrama, podem observar una classe Escena, de la qual hereten les diferents escenes que trobem a la interfície de l'aplicació.



Cada escena té associada un fitxer .fxml on es troba la distribució dels diferents components de l'escena (botons, camps de text, pestanyes...). Aquests fitxers es poden editar manualment, o bé utilitzant un *SceneBuilder* (consultar Apèndix). Addicionalment, també es pot incloure un fitxer *.css* per modificar l'estil dels components de l'escena (no es valorarà en aquesta pràctica).



Per crear una nova finestra amb una escena, cal crear un nou fitxer .fxml associat a aquesta, i utilitzar el mètode *creaEscena* de *EscenaFactory*, tot indicant per paràmetres el nom del fitxer (sense l'extensió), i el títol que volem que tingui la finestra. Mira per exemple la línia 17 de la classe Vista.java on es crea una finestra per mostrar la finestra de login:

```
Escena login = EscenaFactory.INSTANCE.creaEscena("login-view", "Login");
```

En el fitxer .fxml de cada escena, trobarem un paràmetre, *fx:controller*, on podem indicar quina és la classe associada a aquesta escena. Per exemple, en el fitxer *login-view.fxml*, trobem al final de la línia 14:

```
        fx:controller="ub.edu.view.EscenaLogin"
```

El mètode *creaEscena* de *EscenaFactory*, ens retorna la instància de l'Escena que acabem de crear (en aquest exemple, ens retornaria una instància de *EscenaLogin*), de manera que podem cridar a mètodes de l'Escena que acabem d'instanciar (línies 19 i 20 de la classe Vista):

```
        //Li enviem la finestra (stage) i el controlador a la nova escena
        login.setController(controller);
```

6. Comença a migrar el teu projecte de la pràctica 3 a aquest nou projecte

Per tal d'incloure el vostre codi de la pràctica anterior, podeu copiar en aquest nou projecte totes les vostres classes de la carpeta **src** de la vostra pràctica 3.

A continuació teniu una sèrie de passos a seguir per començar a desenvolupar el Mínim Producte Viable 1 d'aquesta pràctica 4:

1. El primer pas es analitzar des d'on es crea el controlador, qui crea la vista i qui crea la part del model. En aquesta pràctica teniu el main inclòs a la classe AppMain en la carpeta src. En el codi base aquesta classe crea el controller i després la vista per que no es té res més. Planteja't qui hauria de crear la vista, el controlador i el model, serà la classe main? Serà el controlador? Serà la vista?
2. Un cop hagis decidit qui crea els principals components del model-vista-controlador, mira com la vista transmetrà les peticions i com rebrà les respostes del model. Explora com s'està fent ara la primera història d'usuari de login de l'usuari.

3. A partir d'aquí, comença a fer la connexió de la interfície gràfica, per a visualitzar els detalls de les excursions. Veuràs que ara surten els llocs d'origen i destí de cada excursió i lligats a aquests llocs, estan les espècies. Canvia els DAO's o les inicialitzacions del model necessàries per suportar aquest canvi.
4. Hi ha algunes històries d'usuari que fan que el model canviï en certes parts i provoca que la vista s'hagi de refrescar automàticament, sense haver de passar pel controlador. Localitza-les i aplica el patró oportú.

Per a aquest lliurament és necessari que utilitzis **patrons** en les parts de la pràctica 3 que no hagis aplicat encara, així com a la part de la Vista, el Controlador i el Model. Analitza quins són els que has de fer servir per tal que un canvi de Vista, Controlador o Model, no impliqui canvis en la resta d'elements.

Apèndix

SceneBuilder

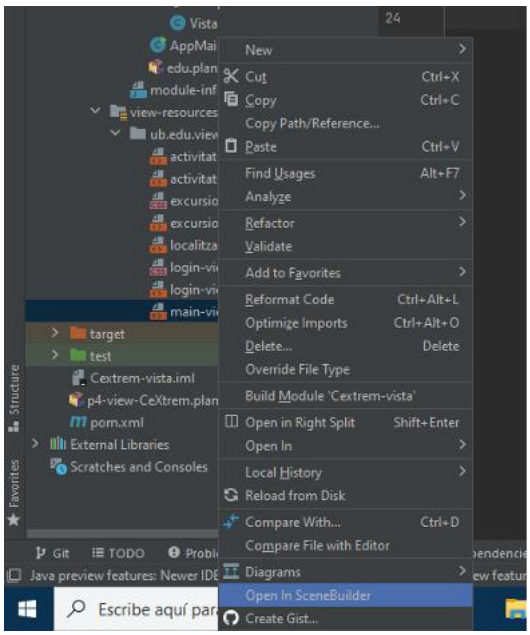
Els SceneBuilders són editors visuals dels fitxers .fxml. Hi ha dos principals que podeu descarregar:

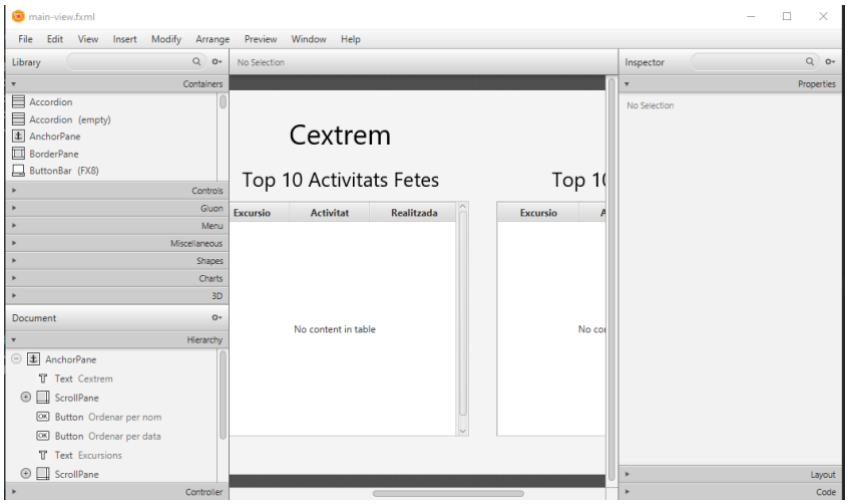
https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html

https://gluonhq.com/products/scene-builder/

Si descarregueu el *SceneBuilder* de *gluonhq*, us proporcionarà alguns components que no s'inclouen a les llibreries del projecte. Aquests components no s'han d'utilitzar (el programa us avisarà abans quan intenteu afegir aquests tipus de components).

Un cop descarregat, podreu modificar els fitxers .fxml fent click dret i seleccionant "Open in SceneBuilder":





Exposant components de la vista des dels fitxers .fxml a les Escenes

Per poder accedir des del codi als diferents components d'una Escena, un cop tingueu creat el component, haureu d'afegir un paràmetre *fx-id* indicant el nom de la variable que vulgueu que tingui el component. Per exemple, pel botó de Login en *login-view.fxml, trobem:*

```
<Button fx:id="login_btn" layoutX="109.0" layoutY="176.0" onAction="#onLoginButtonClick" text="Login" />
```

I, en *EscenaLogin.java,* tenim declarats tots els components que hem exposat en el fitxer *.fxml*:

```
public class EscenaLogin extends Escena {

    public Button login_btn;
    public TextField login_correu;
    public PasswordField login_pwd;
    public Button registre_btn;
    public TextField registre_correu;
    public PasswordField registre_pwd;
```

Fixem-nos també que, en el fitxer *.fxml*, hem posat la tag *onAction="#onLoginButtonClick".* Això vol dir que, en fer click en aquest botó, es cridarà al mètode *onLoginButtonClick* de la classe *EscenaLogin*.

```
@FXML
protected void onLoginButtonClick() {
```