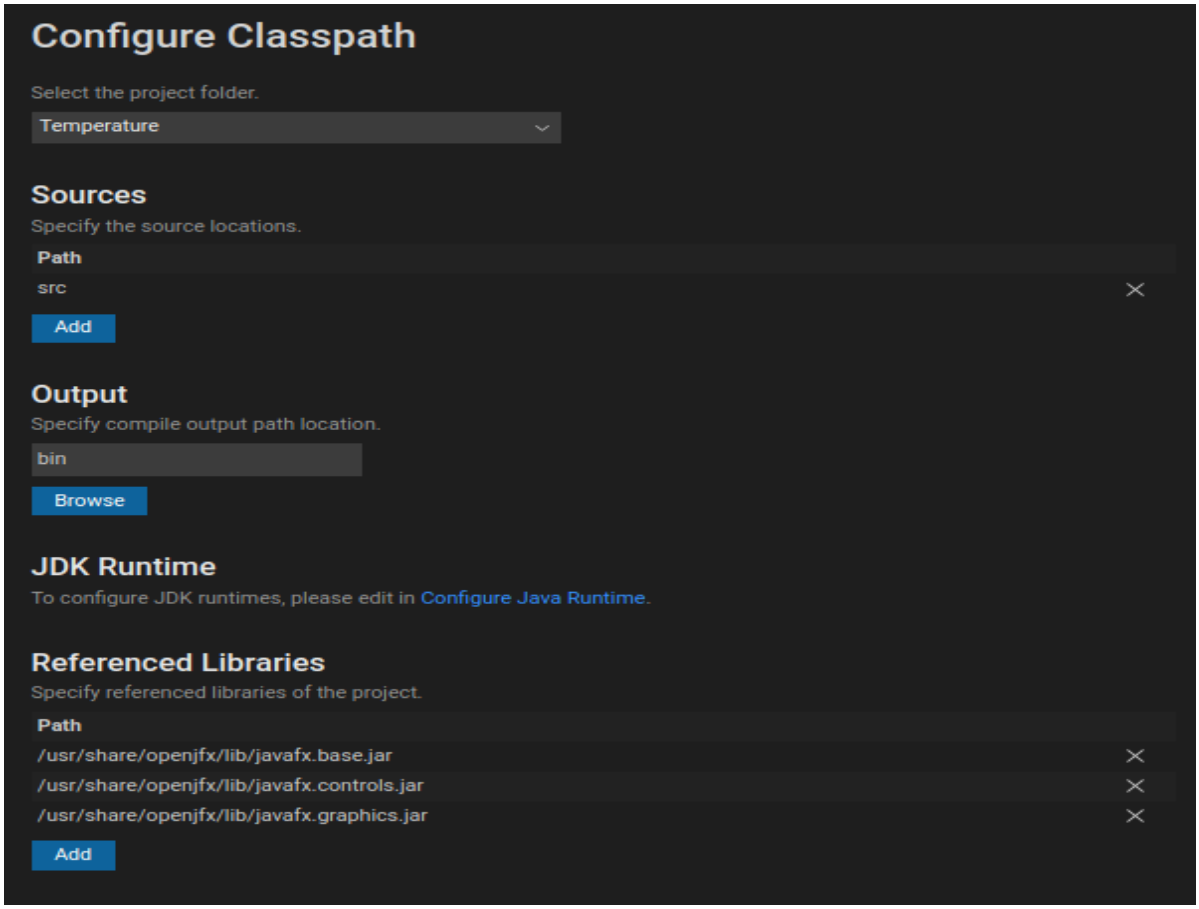
Objectifs

- Apprendre à paramétrer VSCode
- Implémenter Vue

Exercice 1 *Paramétrer le "bouton magique" de VSCode*

Il est possible de configurer le "bouton magique" de VSCode pour qu'il compile le code source et excécute le programme. Il faudra refaire cette manipulation **pour chaque projet** [1].

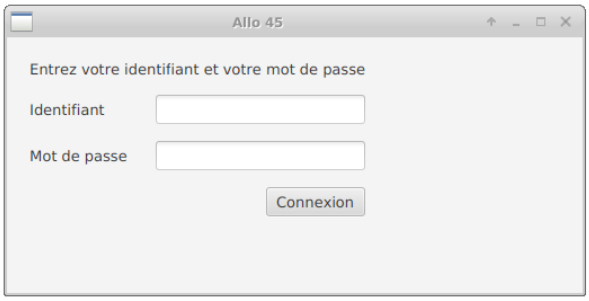
1. Ouvre un projet java (par exemple, le projet *Temperature* de la semaine dernière).
2. Dans le menu *Executer* > *Ajouter une configuration...* ou *Ouvrir les configurations*
   Après la ligne `"request": "launch",` ajoute la ligne :
   `"vmArgs": "--module-path /usr/share/openjfx/lib/ --add-modules javafx.controls,javafx.fxml",`
   N'oublie pas d'enregistrer le fichier `launch.json`
3. Dans le menu *Affichage*> *Palette de commandes...* > *Java : Configure Classpath*
   Ajoute le chemin des librairies javafx utilisées dans "Referenced Libraries".



4. Maintenant, tu peux appuyer sur le "bouton magique" pour compiler et exécuter le programme.

1. Si tu sais comment faire en sorte que cette configuration devienne la configuration par défaut, merci de me dire comment ;) !

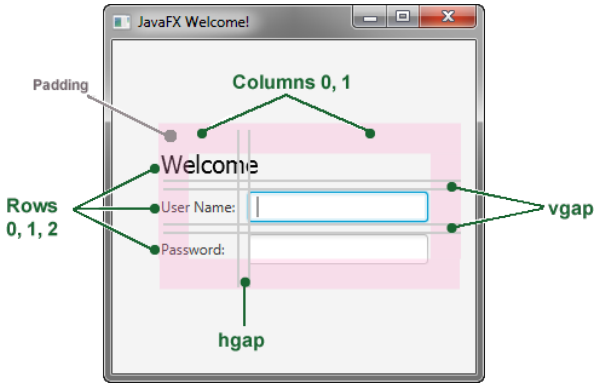
Exercice 2 *Une fenêtre de connexion*



On veut construire une Vue qui correspond à la maquette ci-contre. Comme racine, on utilisera un `GridPane` de quatre lignes et deux colonnes.

2.1 Identifie le objets graphiques de cette fenêtre

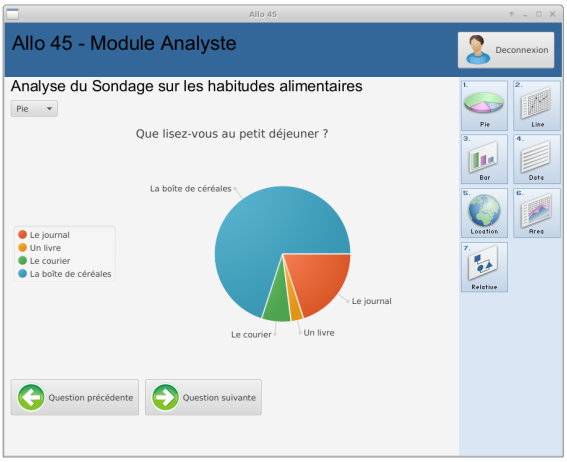
2.2 Crée une classe `FenetreConnexion` qui permet d'obtenir cette fenêtre.



2.3 Gère l'espacement entre les éléments de ce `GridPane`. Pour cela, tu peux utiliser les méthodes :

- `setVgap(int)` qui détermine l'espacement entre deux lignes,
- `setHgap(int)` qui détermine l'espacement entre deux colonnes
- `setPadding(Insets)` qui détermine l'espacement des éléments par rapport aux bords de la fenêtre.

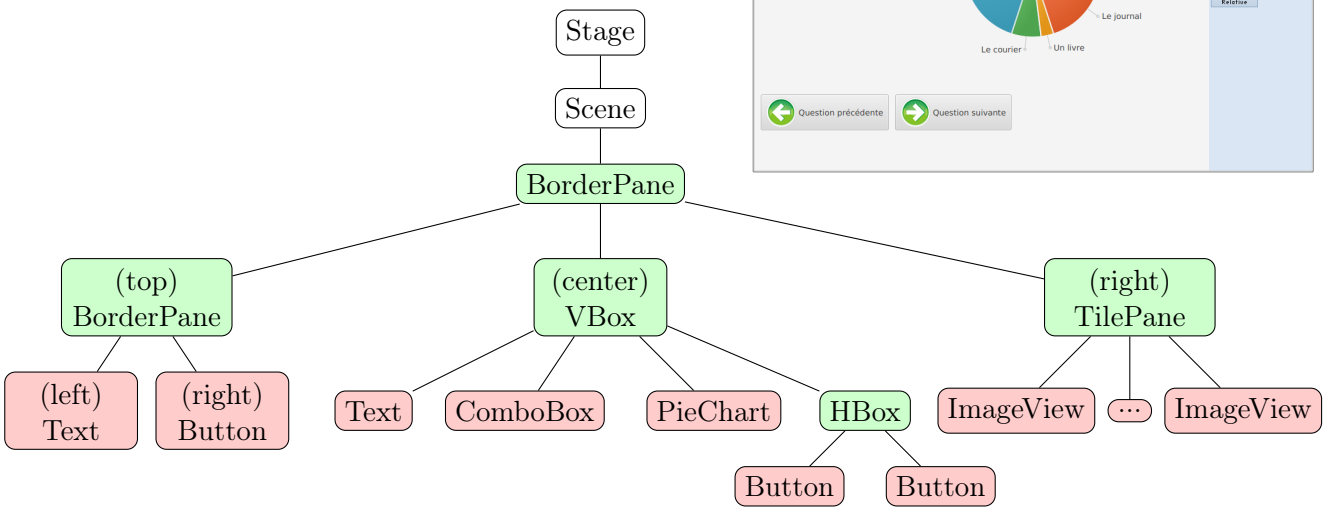
Exercice 3 *Une fenêtre plus compliquée*



On veut construire une Vue qui correspond à la maquette ci-contre et dont on donne le graphe de scène ci dessous.

3.1 Récupère les images sur Célène.

3.2 Crée une classe `FenetreAnalyste` qui permet d'obtenir cette fenêtre.



Quelques indications :

- Exemple de code pour modifier la police de caractère d'un texte :

```
Text text = new Text("Bonjour tout le monde !");
text.setFont(Font.font("Arial", FontWeight.BOLD, 32));
```

- Exemple de code qui permet de créer un diagramme circulaire (`PieChart`) :

```
PieChart chart = new PieChart();
chart.setTitle("Que lisez-vous au petit déjeuner ?");
chart.getData().setAll(
    new PieChart.Data("Le journal", 21),
    new PieChart.Data("Un livre", 3),
    new PieChart.Data("Le courier", 7),
    new PieChart.Data("La boîte de céréales", 75));
chart.setLegendSide(Side.LEFT); // pour mettre la légende à gauche
```

Exercice 4 *Une seule application, deux fenêtres*

4.1 Récupère le code le l'application *PlusieursFenetres* sur Celene et lance la pour voir ce qu'elle fait.

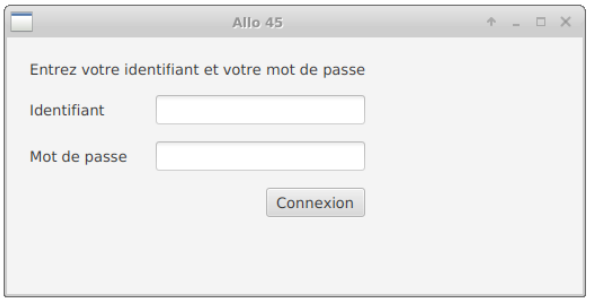
4.2 Modifie le code de façon à ajouter une nouvelle fenêtre (une VBox par exemple).

Exercice 5 *Une seule application, deux fenêtres : refactorisation de code*

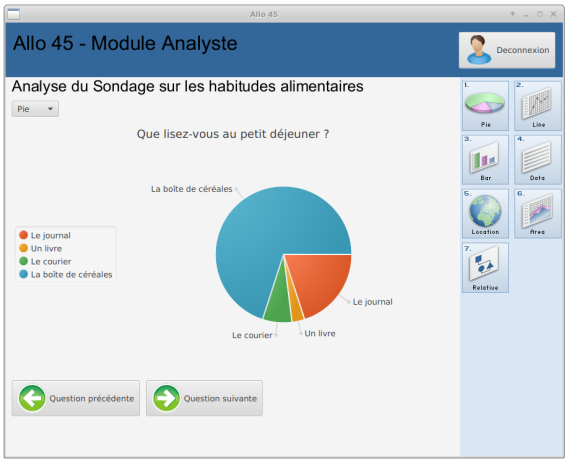
On veut maintenant construire une application dont la scène affiche la fenêtre de connexion quand on la lance. Quand on appuie sur le bouton *Connexion* la scène doit afficher la fenêtre du module analyste. Quand on appuie sur le bouton *Déconnexion* la scène doit afficher la fenêtre de connexion.

Pour cette application, nous n'utiliserons aucun modèle et seuls les deux boutons (Connexion et Déconnexion) seront fonctionnels pour permettre de passer d'une page à l'autre.

Utilise le code des exercices précédents pour coder une telle application.



Fenêtre de connexion



Fenêtre du module analyste

Exercice 6 *Graphe de Scène*

Construis une Vue qui correspond à la maquette ci-contre.