Objectif

Au cours des quatre prochaines séances de TP, vous allez fabriquer une interface graphique permettant de manipuler les données d'une table. Ces données seront stockées dans un fichier.

Cahier des charges

On souhaite représenter la relation « séance(matière, nature, date, durée, salle, enseignant) ». avec les contraintes suivantes (comme lors du TP6 AP1.2, `https://moodle.univ-lr.fr/2017/pluginfile.php/135564/mod_resource/content/1/sujet_AP12_TD6_2018.pdf`) :

- il n'y a que quatre matières : AP, BD, CDIN et SI ;
- il n'y a que trois natures : Cours, TD et TP ;
- la date est écrite en format normalisé « AAAA-MM-JJ » ;
- la durée est exprimée en minutes ;
- la salle est décrite dans une chaîne de caractères selon le format « département, étage, numéro de salle » comme par exemple « D204 ».

L'utilisateur pourra consulter l'ensemble du contenu de la table, regarder le détail d'une séance, la modifier, ajouter une nouvelle séance et sauvegarder les modifications dans un fichier.

Travail

Vous disposez de quatre semaines pour fabriquer cette application. Vous travaillerez en binôme.

Semaine 18

L'objectif de cette semaine est de concevoir l'interface graphique. Cette interface doit utiliser le composant graphique complexe `TableView`.

1. Analysez les besoins de l'utilisateur.
2. Proposez une interface graphique comprenant une ou plusieurs fenêtres.
3. Utilisez « Scene Builder » pour fabriquer cette interface.
4. Fabriquer les contrôleurs associés à vos fenêtres.

Annexe

Utilisation de `TableView`

Lorsque vous utilisez un objet `TableView` vous devez définir les colonnes que vous afficherez, avec l'objet `TableColumn`. Vous devez ensuite associer des données à ces objets.

Vous devez associer un objet de type `ObservableList<>` à l'objet `TableView`. Par exemple :

```
1  @FXML
2  private TableView tvSeance;
3
4  ObservableList<Seance> olDonnees = FXCollections.ObservableArrayList();
5
6  // Associer l'ObservableList<> au TableView
7  this.tvSeance.setItems(olDonnees);
8
9  // methode (que vous avez definie) qui affiche le detail d'une seance dans
10 // d'autres objets graphiques
11 showSeanceDetails(null);
12
13 // associe un gestionnaire d'evenements a un evenement de selection d'un
14 // element de la liste
15 this.tvSeance.getSelectionModel().selectedItemProperty().addListener(
16   (observable, oldValue, newValue) ->{
17     showSeanceDetails((Seance) newValue);
```

```
18      });
```

Vous devez, également, préciser les données que vous affichez dans les `TableColumn`. Par exemple :

```
1  // Acces aux objets graphiques
2  @FXML
3  private TableColumn<Seance, String> tcMatiere;
4  @FXML
5  private TableColumn<Seance, String> tcDate;
6
7  // Lien entre l'objet graphique et les donnees
8  this.tcMatiere.setCellValueFactory(cellData -> cellData.getValue().getMatiere());
9
10 this.tcDate.setCellValueFactory(cellData -> cellData.getValue().getDate().asString());
```

On utilise une « lambda expression », qui reçoit en paramètre un élément de la liste observable associée au `TableView` (un objet de type `Seance`), et qui précise quelle donnée afficher.

Utilisation de plusieurs fenêtres

Lorsque l'on utilise plusieurs fenêtres, l'objet qui étend le type `Application` joue le rôle de « contrôleur principal ». Il est alors nécessaire de transmettre des informations aux contrôleurs associés à une fenêtre.

La construction d'une fenêtre (chargement et instanciation des éléments graphiques et instanciation du contrôleur) doit se faire dans le contrôleur principal. Il faut garder une association entre les contrôleurs en utilisant la méthode « getController » de l'objet de type `FXMLLoader`. Par exemple :

```
1  // Attention, il existe plusieurs facons de charger un document fxml,
2  // le controleur associe n'est pas toujours correctement
3  // renseigne (nullPointerException)
4  FXMLLoader leLoader = new FXMLLoader(getClass().getResource("vue/Formulaire.fxml"));
5  Parent root = (Parent)leLoader.load();
6
7  // FormulaireController et le controleur associe au document fxml
8  FormulaireController leControleur = leLoader.getController();
9  if (leControleur == null){
10   System.out.println("Pas de controleur");
11 }
12
13 // Methode ajoute au controleur permettant de realiser une association
14 // du controleur de la fenetre vers le controleur principal
15 leControleur.setMainApp(this);
```

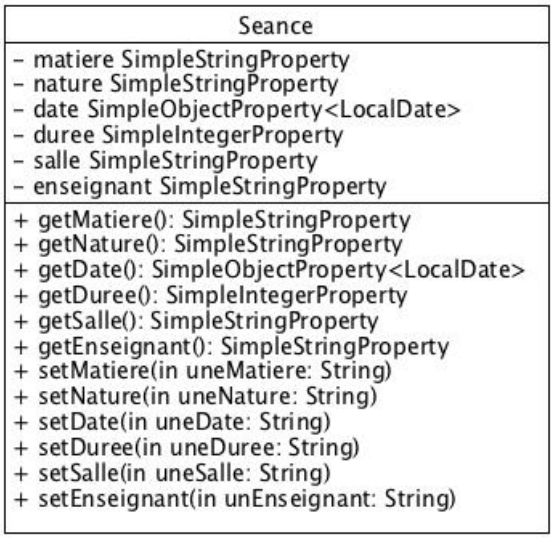

FIGURE 1 – Classe « Seance »

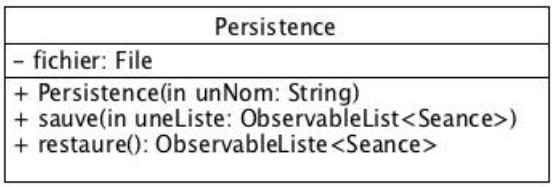

FIGURE 2 – Classe « Persistence »