TP 3
Evénements

Un évènement, en informatique, est une notification indiquant que "quelque chose" s'est produit. Cela peut-être l'utilisateur qui a cliqué sur un bouton, appuyé sur une touche de son clavier, a bougé sa souris, ou encore d'autres actions…

Dans le cadre de ce TP, vous allez utiliser ces évènements afin de pouvoir rendre vos interfaces interactives.

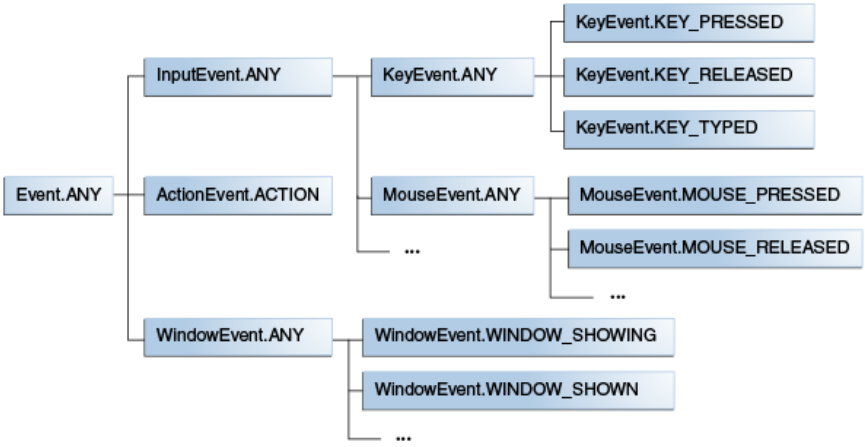
L'objectif de ce TP est d'étudier 2 manières différentes de gérer des évènements :

- ***La première est d'utiliser des « expressions lambdas »***
- ***La seconde est le référencement de méthode***

1) Fonctionnement des évènements

A savoir

Plus qu'une simple notification, un évènement est une structure de données (javafx.event.Event) contenant les données relatives à sa création. Celles-ci contiennent par exemple son type (ex : KeyEvent pour les évènements claviers, MouseEvent ceux de la souris ou encore ActionEvent pour les évènements produits lors du clic sur un bouton), la touche du clavier qui a été tapée ou encore la position de la souris.



Un gestionnaire d'évènements (EventHandler) est utilisé de pair afin de pouvoir définir des interactions ou des traitements particuliers suite à un évènement donné.

2) Gestion d'évènements en utilisant des « lambda expressions »

Téléchargez l'exemple de code JavaFX depuis le répertoire TP3 dans Chamilo.

Afin de gérer les évènements dans cette partie, vous allez utiliser les expressions lambdas qui permettent d'implémenter une interface à méthode unique (ici la méthode handle) par le biais d'une expression que l'on peut passer en paramètre. Elles évitent d'écrire une classe de gestion d'événements pour chaque événement.
L'exemple ci-dessous permet de mettre en place un gestionnaire d'évènements, lors du clic (.setOnAction) sur le bouton « monBoutonReset », afin de vider les champs concernés.

```
monBoutonReset.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        tfNom.setText("");
        tfPrenom.setText("");
        pfMdP.setText("");
    }
});
```

Etapes

1. Analysez le code de la classe TP3_Exo1App.java ainsi que l'IHM.
2. Commencez par ajouter dans la classe TP3_Exo1App.java un premier gestionnaire d'évènements pour le bouton **Reset** (voir exemple ci-dessus…).
3. Modifiez ensuite l'interface pour ajouter une nouvelle ligne entre prénom et mot de passe. Cette ligne doit contenir un label « **Login** : » à gauche, et un second label pour la partie droite. En effet, le login ne devra pas être saisi par l'utilisateur, mais calculé par votre programme, le champ ne doit donc pas être éditable.
4. Mettez en place les gestionnaires d'évènements nécessaires au calcul du **Login**. De manière simplifiée, un login est composé des 7 premiers caractères du nom de famille, et du premier caractère du prénom, et il est uniquement composé de caractères minuscules. Si le nom de famille fait moins de 7 caractères, il suffit de prendre tout le nom.
   Aide : événement setOnKeyTyped
5. Modifiez le gestionnaire d'évènements mis en place pour le bouton **Reset** afin de gérer le nouveau champ Login.
6. Ajoutez ensuite différents gestionnaires d'évènements afin de pouvoir reproduire le fonctionnement décrit ci-dessous :
   1. **Vérifier** : le bouton Vérifier colore en rouge le label des champs posant problème. Les règles sont : le nom et le prénom ne doivent pas être vides, et le mot de passe contient 8 caractères ou plus. (NB : pensez aussi au cas où un champ était précédemment invalide, mais qu'il respecte maintenant les règles).
      Aide : événement setOnAction
   2. **Quitter** : ce bouton ferme la fenêtre, et termine l'exécution du programme
   3. **TextArea** : le TextArea en bas de la fenêtre doit afficher les informations saisies dans le formulaire **uniquement** lorsque la souris est **au-dessus** de la zone
      Aide : événement setOnMouseEntered

3) Gestion d'évènements via référencement de méthodes et scene builder

Pour cet exercice, commencez par recréer l'interface de l'exercice 1 grâce au Scene builder (fichier TP3_Exo2-view.fxml).

Les évènements seront cette fois gérés grâce au référencement de méthodes au sein de Scene builder.

1. Pour cela, donnez un identifiant unique aux différents éléments de votre interface qui devront déclencher un évènement ou subir une modification (voir onglet Code, champ « fix:id »).

2. Dans votre Controller (TP3_Exo2Controller.java), vous pouvez maintenant référencer cet élément :

```
@FXML
public Label labelNom;
```

3. Toujours dans le Controller, ajoutez des méthodes pour le traitement de vos différents évènements :

```
@FXML
private void resetButtonAction(ActionEvent event) {
    tfNom.setText("");
}
```

4. Référencez enfin ces méthodes au sein de Scene builder, en cliquant sur les interacteurs concernés (voir onglet Code, champ « On Action »).

5. Testez votre interface et les différents évènements mis en place.

4) Création et affichage de nouvelles scènes

Pour cet exercice, repartez de l'interface que vous avez réalisé dans l'exercice 1 du TP2 :

Le but de cet exercice est :

- D'afficher une nouvelle scène dans la fenêtre principale lors du clic sur le bouton Valider. Cette nouvelle scène doit présenter un récapitulatif de toutes les informations saisies dans la scène précédente, ainsi qu'un bouton permettant de revenir à cette dernière.
- D'ouvrir une nouvelle fenêtre lors du clic sur le bouton Annuler, permettant de confirmer l'action (fermeture du programme) ou d'annuler et revenir à la fenêtre principale (fermeture uniquement de la fenêtre de confirmation).

1) Commencez par créer un nouveau FXML via sceneBuilder pour l'interface récapitulative. Pensez à affecter des id et des références de méthodes aux éléments nécessaires, dans Scene Builder.

Pour remplir la combobox avec la liste des lieux de résidence, il faut initialiser le composant dans une méthode initialize() du Controller. Puis utilisez les méthodes getItems() d'une combobox et setAll(…) pour affecter des valeurs.

De plus, dans votre fichier fxml (ouvert en mode « text »), supprimez l'affectation d'un contrôleur, présent en fin de ligne de votre container principal. =>fx:controller=« com.example.xxxx.HelloController"

Pour ouvrir ce nouveau fichier comme nouvelle scène de la même fenêtre (stage), suivez l'exemple de code ci-dessous, à mettre en place dans votre Controller.

```
@FXML
public void actionBoutonValider (ActionEvent event) throws IOException {
    FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("maSecondeScene.fxml"));
    fxmlLoader.setController(this);

    Stage fenetre = (Stage) btnValider.getScene().getWindow();
    Scene scene = new Scene(fxmlLoader.load(), 300, 275);

    // récupération des champs de la première scène et affectation aux champs de
la deuxième
    tfNomRecap.setText(tfNom.getText());
    // etc...

    fenetre.setScene(scene);
    fenetre.show();
}
```

La nouvelle interface à ouvrir devra être pourvue d'un bouton retour permettant de revenir à la scène principale. Pour cela, ajoutez dans le controller le code suivant (avec « monBoutonValider » qui est un bouton de la première scène et permet donc de retourner celle-ci) :

```
@FXML
public void actionBoutonRetour (ActionEvent event) throws IOException {
    Stage fenetre = (Stage) monBoutonRetour.getScene().getWindow();
    fenetre.setScene(monBoutonValider.getScene());
    fenetre.show();
}
```

2) Afficher les informations saisies dans la fenêtre du TP2.

Pour récupérer la valeur sélectionnée parmi des radio boutons, utilisez la méthode getSelectedToggle() de ToogleGroup.

Pour savoir si un choix (CheckBox) est sélectionné, utilisez la méthode isSelected().

3) Créer et afficher une nouvelle fenêtre

Le but de cet exercice est de :

- Apprendre à gérer les ouvertures et fermetures de fenêtres

1) Suivez les recommandations de l'exercice précédent, pour ouvrir une nouvelle fenêtre dans le même stage :

Spécifiez bien le contrôleur comme étant le contrôleur courant ; fxmlLoader.setController(this);

Attention : le contrôleur ne doit pas être défini dans le fichier fxml.

```
public void actionBoutonXXX(ActionEvent event) throws Exception {
    try {
        // Chargement de la nouvelle interface
        FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("ma-
seconde-scene.fxml"));
        Parent nouvelleScene = fxmlLoader.load();

        // Récupération du stage principal à partir du bouton cliqué (ou d'un
autre élément)
        Stage fenetre = (Stage) btnXXX.getScene().getWindow();

        // Ajout de la nouvelle scène au stage et affichage
        fenetre.setScene(new Scene(nouvelleScene));
        fenetre.show();

    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

2) Modifiez le code de manière à ne pas fermer la fenêtre principale.
Indication : pensez au Stage.

3) Complétez le code de manière à lancer une nouvelle fenêtre qui ne fermera pas la fenêtre principale. La nouvelle fenêtre demandera la confirmation du souhait de quitter (boutons oui/non).
Votre code devra permettre fermer l'application en cas de choix du bouton « Oui ».
Votre code devra permettre de revenir à la fenêtre principale en cas de choix du bouton « Non ».