M2105: Interfaces Homme-Machine

Feuille de TP n°2

Les événements

L'objectif de cette feuille est d'avoir une première approche sur la programmation événementielle en JavaFX. **Attention !** Le code que nous allons produire dans cette feuille ne sera pas obligatoirement un exemple à suivre ! Il s'agit ici de découvrir à la fois l'API de programmation et son fonctionnement en pratique.

Exercice 1 *Le B-A BA : clic ! clic !*

On va tout d'abord créer une application composée d'une fenêtre contenant un bouton. Créez la classe `Fomulaire` qui permet de d'obtenir cette fenêtre.

Nous allons considérer que le bouton a été créé par l'instruction :

```
Button b=new Button("Bouton 1");
```

Le principal événement qui peut arriver sur un bouton est `Action` qui signifie que l'on a cliqué dessus. On va créer une classe `ActionBouton` qui implémente un `EventHandler` pour gérer les événements sur ce bouton.

```
public class ActionBouton implements EventHandler<ActionEvent>{
}
```

1. Créez la classe `ActionBouton` et implémentez la méthode :

   ```
   public void handle(ActionEvent e) {
    System.out.println("Clic!");
   }
   ```

   Dans le constructeur de la classe `Formulaire` créez un objet de type `ActionBouton` après la création du bouton :

   ```
   ActionBouton abo = new ActionBouton();
   ```

   Si vous exécutez votre formulaire et que vous cliquez sur le bouton vous pouvez constater qu'il ... ne se passe rien !!! Effectivement, on a bien créé un gestionnaire pour l'événement `ActionEvent` mais on ne l'a pas enregistré auprès du bouton !

2. Pour remédier au problème, ajoutez dans la classe Formulaire après la création du bouton et du gestionnaire, l'instruction :

   ```
   b.setOnAction(abo);
   ```

   Maintenant, si vous relancez votre formulaire, vous devriez voir l'effet de votre gestionnaire.

Nous allons maintenant enrichir notre formulaire d'un `TextField` que nous allons appeler `t`.

L'`ActionEvent` qui correspond à l'appui sur la touche entrée dans le `TextField`. Vous pouvez essayer en ajoutant l'instruction

```
    t.setOnAction(abo);
```

après la création du `TextField` et du gestionnaire de l'exercice 1. En lançant votre application vous devez constater que l'appui sur la touche entrée dans le `TextField` a le même effet que le clic sur le bouton.

Exercice 2 *Spécialiser un gestionnaire d'événements*

Comme nous l'avons vu dans l'exercice précédent, On peut utiliser un même gestionnaire pour plusieurs objets ce qui peut devenir compliqué car nous devrons reconnaître quel objet a vraiment reçu l'événement ! Pour résoudre ce problème il y a plusieurs solutions qui ont toutes des avantages et des inconvénients :

1. Créer une classe gestionnaire par objet du formulaire. Dans notre cas cela reviendrait à implémenter deux `EventHandler<ActionEvent>`. Cette solution ne facilite pas la factorisation du code et peut générer une multitude de petites classes ! Pas très facile à maintenir.
2. Fournir des éléments permettant de retrouver l'objet qui a reçu l'événement. Pour cela on peut utiliser la méthode `getSource()` ou `getTarget()` des événements.

En général, on va trouver un juste milieu. Ici nous allons faire un gestionnaire d'événements action pour plusieurs boutons. Nous allons faire les actions suivantes :

1. Enlever l'instruction `t.setOnAction(abo);`
2. Ajouter un bouton dont le label sera `Bouton 2` pour lequel on va enregistrer `abo` comme gestionnaire d'événements action.
3. Modifier la méthode `handle()` de `ActionBouton` comme ci dessous.

```
public void handle(ActionEvent actionEvent) {
  Button b= (Button) actionEvent.getTarget();
  System.out.println("clic sur "+b.getText());
}
```

Essayez le formulaire. Vous pouvez constater que le même gestionnaire peut gérer ainsi plusieurs boutons.
Cependant, en général lorsque l'on clique sur un bouton on a souvent besoin d'avoir accès à d'autres informations que juste le nom du bouton qui a été cliqué comme par exemple le contenu d'autres champs du formulaire. Pour cela, on va effectuer les actions suivantes :

1. Ajouter un constructeur à la classe `ActionBouton` qui prend en paramètre l'application.

```
    private Formulaire appli;
    public ActionBouton(Formulaire appli){
        this.appli=appli;
    }
```

2. Modifier la classe `Formulaire` pour faire du champ de texte un attribut de la classe dont vous ajoutez le getter. Faites attention à ne pas redéclarer le champ texte dans la méthode de création de la scène!!!

```
private TextField tf;
public Scene formulaire(){
    ...
    this.tf=new TextField();
    ...
}
public TextField getTf(){ return this.tf;}
```

3. Modifier la méthode `handle()` de `ActionBouton` de la manière suivante

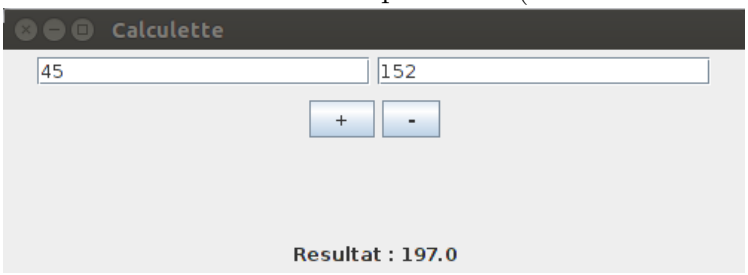
```
    public void handle(ActionEvent actionEvent) {
         Button b= (Button) actionEvent.getSource();
         if (b.getText().endsWith("Bouton 1")) {
             System.out.println("Le champs texte contient " +
                this.appli.getTf().getText());
         }
         else{
             this.appli.getTf().setText("");
             System.out.println("J'efface le champ texte");
         }
     }
```

Le bouton 1 va afficher le contenu du champ texte et le bouton 2 va le supprime.

ATTENTION!!! Les deux exercices suivants sont à rendre

Exercice 3 *Calculette*

Nous allons créer une mini calculette à deux opérations (l'addition et la soustraction).



Dans ce formulaire, l'utilisateur entre des nombres dans les deux champs prévus à cet effet. Lorsqu'il clique sur l'un des boutons, l'opération correspondante est effectuée et le résultat est affiché dans le `Label` du bas.

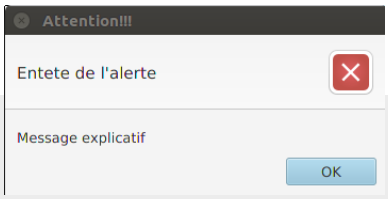
Nous souhaitons avoir le comportement suivant lorsque l'on clique sur un des deux boutons :

- Si les deux champs texte contiennent bien des nombres, on écrit le résultat de l'opération dans le `Label` du bas
- Sinon, on affiche un message indiquant à l'utilisateur le problème et on remet le focus sur le champ en erreur. Le label doit être `Résultat : ERREUR`.

Pour faire cela vous aurez besoin des éléments suivants

1. Pour tester qu'un texte contient bien un nombre il suffit de vérifier que l'appel `Float.parseFloat(texte);` ne lève pas d'exception
2. Pour mettre le focus sur un contrôle particulier il suffit d'utiliser la méthode `requestFocus()` des contrôles.

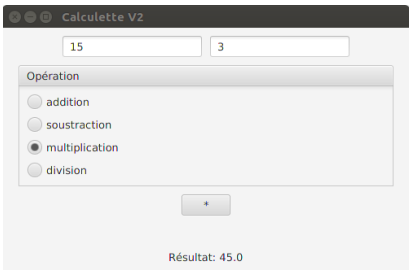
3. Pour afficher un message à l'utilisateur dans une boite de dialogue, vous pouvez utiliser la classe `Alert` comme dans l'exemple ci-dessous




```
Alert al = new Alert(Alert.AlertType.ERROR);
al.setTitle("Attention!!!");
al.setHeaderText("Entete de l'alerte");
al.setContentText("Message explicatif");
al.showAndWait();
```

Exercice 4 *Améliorations*

Pour cette nouvelle version de la calculette on souhaiterait obtenir l'affichage suivant :



On souhaite notamment que la sélection d'un radio bouton entraîne la mise à jour de l'étiquette du bouton de validation avec l'opération correspondante. Pour faire cette nouvelle calculatrice vous pouvez repartir du code précédent, par exemple en écrivant une nouvelle méthode pour construire le graphe de scène, compléter le gestion d'événements Action pour le bouton et créer un gestionnaire d'événements pour les actions sur les radio boutons.