FEUILLE TP N°1 *On dessine des disques colorés*

Objectifs

- Savoir compiler et exécuter une application JavaFX en ligne de commande
- S'initier à la programmation événementielle
- Lire et comprendre du code Java
- Ajouter des fonctionnalités à une application existante

Dans cette feuille, la compilation du code, l'exécution du programme et la génération de la javadoc se fera en lignes de commandes

Exercice 1 *Compiler et exécuter une application JavaFX*

Depuis la version 11 de Java, la librairie JavaFX est dissociée du JDK. Il est donc nécessaire d'indiquer aux commandes `javac`, `java` et `javadoc` où trouver cette librairie et quels modules ajouter. Pour cela, on ajoute donc deux options à ces commandes :
`--module-path /usr/share/openjfx/lib/` et `--add-modules javafx.controls`[1].

Pour **compiler** :

```
javac --module-path /usr/share/openjfx/lib/ --add-modules javafx.controls *.java
```

Pour **executer** :

```
java --module-path /usr/share/openjfx/lib/ --add-modules javafx.controls Executable
```

1.1 Récupère le projet "On dessine des disques" sur Celene. Compile et exécute le code fourni.

Manipule quelques secondes l'application pour répondre aux questions suivantes :

1.2 A quoi sert le curseur à droite ? Est-il fonctionnel ?

1.3 A quoi sert le bouton Quitter ? Est-il fonctionnel ?

1.4 A quoi sert le bouton Annuler ? Est-il fonctionnel ?

1.5 Que se passe-t-il lorsque je clique sur la "boite" grise au centre ?

1.6 Que doit-il se passer si j'appuie sur la touche + ou sur la touche - de mon clavier ? Cette fonctionnalité est-elle correctement implémentée ?

1. Il faudra probablement adapter les chemins sur ton PC perso si tu utilises Windows ou si tu as installé javaFX ailleurs.

Exercice 2 *Générer la javadoc*

Pour **générer la javadoc** d'une application JavaFX, il faut également ajouter les deux options : `--module-path /usr/share/openjfx/lib/` et `--add-modules javafx.controls`.

On spécifie également le dossier dans lequel placer la documentation générée avec l'option `-d` :

```
javadoc -d doc --module-path /usr/share/openjfx/lib/ --add-modules javafx.controls *.java
```

On peut également ajouter d'autres options :

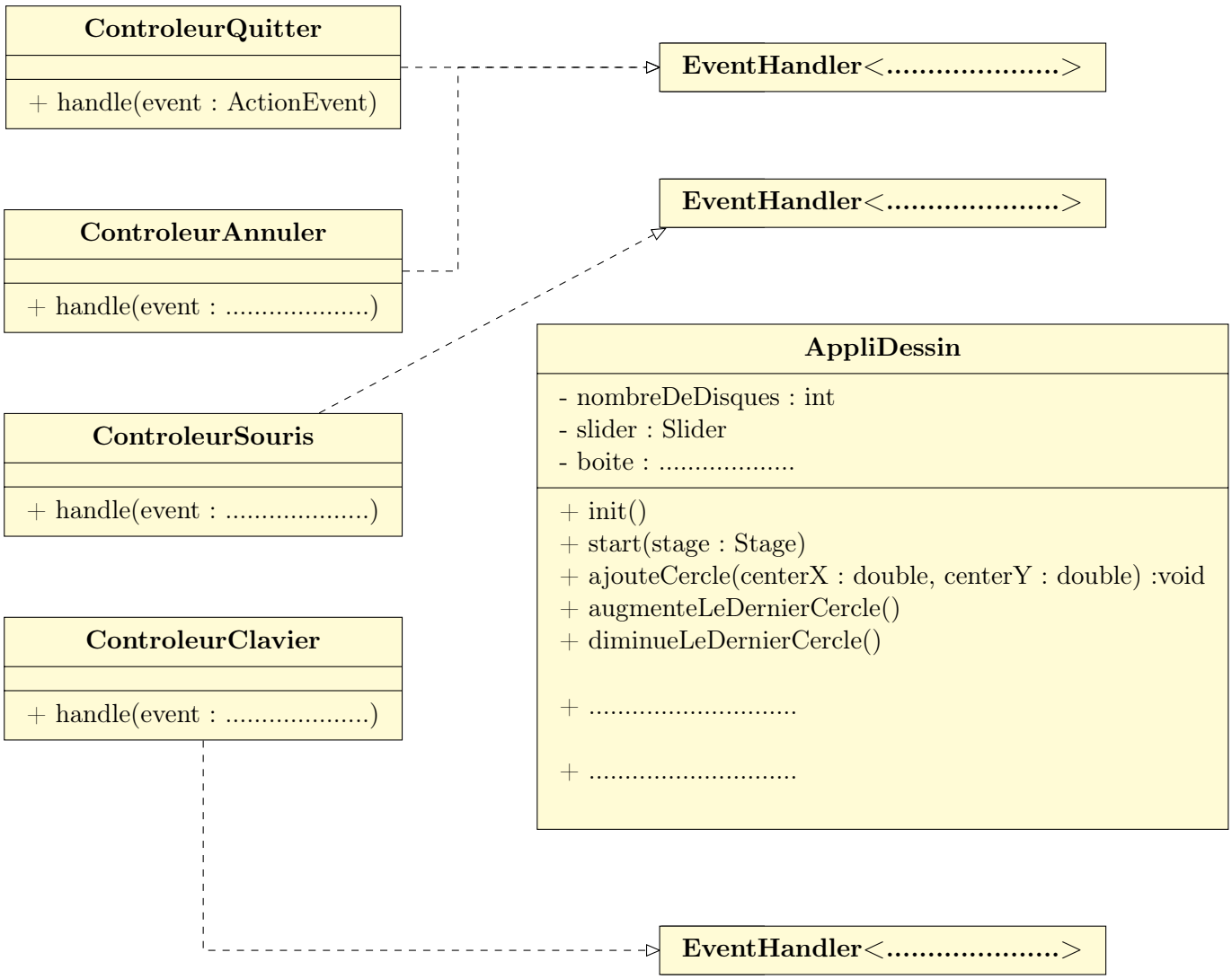
- `-charset name` pour préciser l'encodage, en particulier si on veut des caractères accentués.
- `-private` pour documenter les attributs et méthodes privés (qui sont ignorés par défaut)
- `-noqualifier all` pour ne pas inscrire dans la doc le nom des packages.

```
javadoc -d doc -charset utf8 -private -noqualifier all --module-path /usr/share/openjfx/
lib/ --add-modules javafx.controls *.java
```

2.1 Génère la javadoc et vérifie qu'elle est correctement créée [2]

2.2 Génère la javadoc avec et sans les options proposées et observe les différences.

2.3 Utilise la javadoc pour compléter le diagramme de classes suivant : préciser le profil des interfaces implémentées et ajoute les deux méthodes publiques manquantes dans la classe `AppliDessin`)



2. Le point d'entrée de la javadoc est le fichier `index.html` mais il est intéressant d'aller voir `allclasses.html` par exemple !.

Exercice 3 *Lecture de doc et lecture de code*

Pour répondre aux questions suivantes, tu auras besoin de consulter la documentation de JavaFX [3]

`https://docs.oracle.com/javase/8/javafx/api/toc.htm`

3.1 Dans la méthode `init()` de la classe `AppliDessin` on appelle un constructeur de `Slider` qui comporte trois paramètres. A quoi correspondent ces paramètres ?

3.2 Dans la méthode `ajouteCercle()` de la classe `AppliDessin` on appelle un constructeur de `Circle` qui comporte trois paramètres. A quoi correspondent ces paramètres ?

3.3 Dans la méthode `ajouteCercle()` de la classe `AppliDessin` on appelle un constructeur de `Color` qui comporte quatre paramètres. A quoi correspondent ces paramètres ?

3.4 Dans quelle classe et dans quelle méthode est instancié le `ControleurQuitter` ? A quel widget est-il connecté ? A l'aide de quelle méthode ?

3.5 Dans quelle classe et dans quelle méthode est instancié le `ControleurSouris` ? A quel widget est-il connecté ? A l'aide de quelle méthode ? Quelle autre méthode aurait-on pu utiliser ?

3.6 Dans quelle classe et dans quelle méthode est appelée la méthode `ajouteCercle()` de la classe `AppliDessin` ? A quelle occasion cette méthode est-elle appelée ?

3.7 Quel contrôleur est "réveillé" lorsque je clique sur le bouton Annuler ? Que fait alors ce contrôleur ?

Exercice 4 *Ajout de fonctionnalités*

4.1 Rends le bouton Quitter fonctionnel en complétant le code de la classe `ControleurQuitter`.

4.2 Complète le code de façon à ce que le rayon du dernier disque dessiné augmente lorsque l'utilisateur appuie sur la touche + de son clavier.

4.3 Complète le code de façon à ce que le rayon du dernier disque dessiné diminue lorsque l'utilisateur appuie sur la touche - de son clavier.

4.4 Complète le code de façon à ce que la couleur dernier disque dessiné change aléatoirement lorsque l'utilisateur appuie sur la touche * de son clavier.

4.5 Mets à jour le "mode d'emploi" qui se trouve en haut de la fenêtre.

Exercice 5 *Somme de deux nombres*

5.1 Récupère sur Celene le projet "Somme de deux nombres". Vérifie que le projet compile et s'exécute.

5.2 Complète le code de façon à rendre fonctionnel le bouton Additionner. Pour cela tu dois compléter le code de la méthode `handle()` de la classe `ControleurAdditionner` et le code de la méthode `additionne()` de la classe `AppliSomme` [4]

5.3 Complète le code de façon à rendre fonctionnel le bouton Reset. Pour cela tu dois compléter le code de la méthode `efface()` de la classe `AppliSomme` et tu dois créer une classe `ControleurReset` et l'instancier pour gérer le bouton.

5.4 Que se passe-t-il si l'utilisateur entre `Arthur` dans l'un des champs ? Résous le problème.

Exercice 6 *Pour aller plus loin*

6.1 Reprends le projet *AppliDessin* et rends fonctionnels les boutons de couleur (à gauche) : quand on clique sur l'un de ces boutons, le dernier disque dessiné doit prendre sa couleur.

6.2 Résous les problèmes de robustesse de cette application

3. Ce site est incontournable pour coder en JavaFX. Il est donc indispensable de l'ajouter à tes favoris.

4. Pour transformer un `String` en nombre, tu peux aller jeter un coup d'oeil à la méthode static `Double.valueOf()`