TP2

L'objectif de ce TP est de se familiariser avec la réalisation de gestionnaires d'événements. Il s'agit d'écrire le code réagissant aux divers événements créés par les composants de l'interface afin de rendre l'IHM pleinement fonctionnelle.

Exercice 1 – compteurs

Nous proposons de réaliser un compteur sous la forme d'un bouton qui incrémente une valeur (initialement à 0), comme illustré à la figure 1.

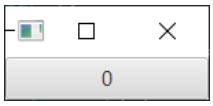

FIGURE 1 – Un compteur

1. Créez une classe `Compteur` constituée d'un unique bouton ayant 0 pour texte initial.
2. Ajoutez au bouton un gestionnaire d'événements sous la forme d'une classe interne et anonyme, chargé d'incrémenter la valeur du bouton lorsque l'utilisateur clique dessus.

Un compteur c'est bien, trois c'est encore mieux, comme illustré à la figure 2.

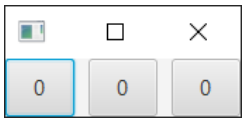

FIGURE 2 – Trois compteurs

3. Créez une classe `CompteursCIA` constituée de trois boutons ayant 0 pour texte initial.
4. Ajoutez à chacun des boutons un gestionnaire d'événements sous la forme d'une classe interne et anonyme, chargé d'incrémenter la valeur du bouton lorsque l'utilisateur clique dessus.
5. Créez une classe `CompteursCI` constituée de trois boutons ayant 0 pour texte initial.
6. Ajoutez à chacun des boutons un gestionnaire d'événements sous la forme d'une classe interne, chargé d'incrémenter la valeur du bouton lorsque l'utilisateur clique dessus.

Exercice 2 – lecteur audio

Nous proposons de réaliser une version simple d'un lecteur audio permettant d'écouter une musique. L'interface du lecteur audio est illustrée à la figure 3. Le bouton *Play*/*Pause* permet de lancer ou mettre en pause la musique, le bouton << permet de redémarrer à zéro la musique, et le curseur permet de gérer le volume.

1. Créez une classe `LecteurAudio` correspondant à la fenêtre du lecteur audio.

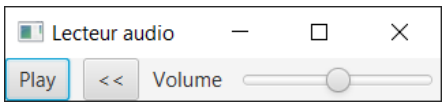

FIGURE 3 – Lecteur audio

2. Créez une instance de la classe `Media`[1] avec un fichier de musique prédéfini, puis, à partir de celle-ci, créez une instance de la classe `MediaPlayer`[2].
3. Ajoutez un gestionnaire d'événements au bouton *Play*, chargé de lancer ou mettre en pause la musique. À chaque clic, le texte du bouton alterne entre *Play* et *Pause*.
4. Ajoutez un gestionnaire d'événements au bouton <<, chargé de redémarrer à zéro la musique.
5. Pour gérer le volume, liez la propriété `volumeProperty` du lecteur à la propriété `valueProperty` du curseur (divisée par 100).
6. Affichez, au démarrage du lecteur, un explorateur de fichiers permettant à l'utilisateur de sélectionner le morceau de musique à jouer dans le lecteur audio.

Exercice 3 – points à relier

Nous proposons de réaliser une application permettant de placer des points à la souris, puis de les relier en traçant un segment du premier point au deuxième, du deuxième au troisième, ..., de l'avant-dernier au dernier.

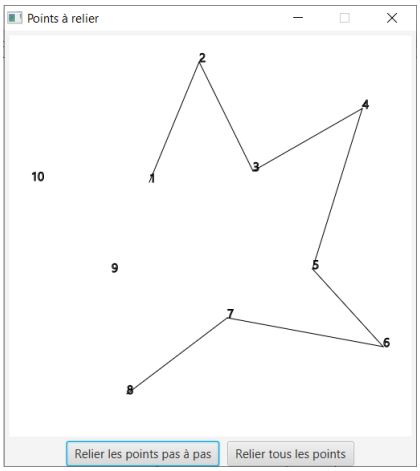

FIGURE 4 – Points à relier

1. Créez une classe `PointsARelier` constituée d'un `Canvas`[3] de dimensions 500×500 sur lequel les points à relier seront affichés, et de deux boutons permettant de relier les points pas à pas ou d'une seule traite, comme illustré à la figure 4.
2. Interdisez le redimensionnement de la fenêtre.
3. Ajoutez une méthode `dessiner(GraphicsContext gc)` qui affiche tous les points (reliés ou non) placés par l'utilisateur sur le `Canvas`. Cette méthode sera à appeler chaque fois que le contenu

1. La classe `Media` représente une ressource multimédia et est instanciée à partir de la forme chaîne de caractères d'une URI.
2. La classe `MediaPlayer` permet de contrôler la lecture des médias. Elle est utilisée en combinaison avec la classe `Media` pour afficher et contrôler la lecture des médias.
3. La classe `Canvas` représente une image qui peut être dessinée à l'aide d'un ensemble de commandes graphiques fournies par un `GraphicsContext`. Toutes les modifications du `Canvas` passent par son contexte graphique (*i.e.*, la matrice de pixels d'un affichage). Pour obtenir ce dernier, utiliser la méthode `getGraphicsContext2D()`.

du `Canvas` doit être redessiné. L'objet passé en paramètre représente le contexte graphique du `Canvas`. La classe `GraphicsContext` contient, entre autres, les méthodes suivantes :

- `setFill(Paint p)` définit la couleur de remplissage,
- `fillRect(int x, int y, int width, int height)` permet de dessiner un rectangle plein en spécifiant les coordonnées de son coin supérieur gauche, sa largeur et sa hauteur,
- `setStroke(Paint p)` définit la couleur des traits,
- `strokeText(String str, double x, double y)` affiche la chaîne de caractères `str` aux coordonnées (`x`, `y`),
- `strokeLine(double` x_1`, double` y_1`, double` x_2`, double` y_2`)` trace un trait de (x_1, y_1) à (x_2, y_2).

(a) Dessinez un rectangle plein blanc aux dimensions du `Canvas`.

(b) Affichez les numéros de chaque point à relier aux coordonnées du point.

(c) Reliez les points à relier.

4. Ajoutez un gestionnaire d'événements au `Canvas` qui ajoute un nouveau point à l'endroit où l'utilisateur à cliquer avec la souris.
5. Ajoutez un gestionnaire d'événements à chaque bouton qui trace un segment du premier point au deuxième, du deuxième au troisième, ..., et de l'avant-dernier au dernier, pas à pas pour l'un, et d'une traite pour l'autre. Les boutons ne doivent être cliquables que s'il reste des points à relier.