Interfaces Graphiques

Exercices - Première série

2015-2016

Exercice 1 (Group, Scene, Stage). Définir une extension de la classe Application. Dans sa méthode `start`, créer un nouveau Group, et une nouvelle Scene de taille 600×400 contenant ce Group. Associer la Scene créée à la Stage argument de la méthode `start`. Ajuster la taille de la Stage à celle de la Scene, puis afficher la Stage à l'écran.

1. Ajouter en tant que descendant du Group un Rectangle centré de taille 400×200, de couleur de bord bleue, de couleur de fond rouge.

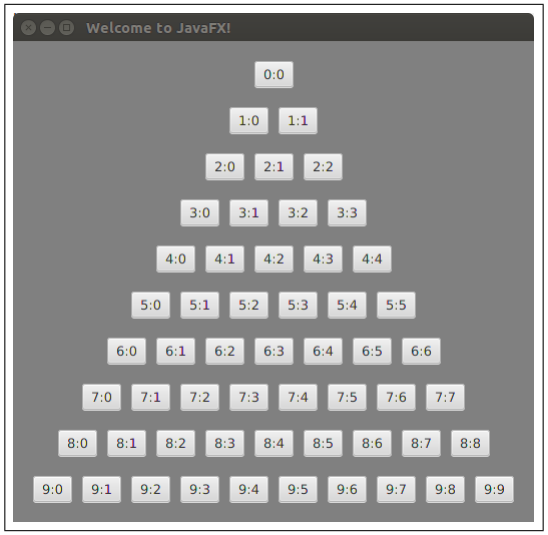
2. Ajouter en descendant supplémentaire du Group un Text centré contenant le message `"Hello World!"`. Les données nécessaires pour ajuster la position du texte sont récupérable via la méthode `boundsInLocalProperty()`.

3. Modifier la fonte du Text en une fonte épaisse de taille 80 pixels. Vérifier que le texte est encore correctement centré quelle que soit la taille de fonte choisie.

4. Appliquer au Rectangle une rotation de 45^o, et observer le résultat. Appliquer également au Group une rotation de 45^o, et observer le résultat.

5. Mettre en commentaires les transformations de la question précédente. Appliquer au Group un effet de type BoxBlur en testant différents réglages.

6. Mettre en commentaires les transformations de la question précédente. Remplir le Rectangle à l'aide d'un LinearGradient balayant le rectangle depuis son coin supérieur gauche jusqu'à son coin supérieur droit, et passant de la couleur bleue à la couleur rouge. Appliquer au Text un effet de type Reflection.



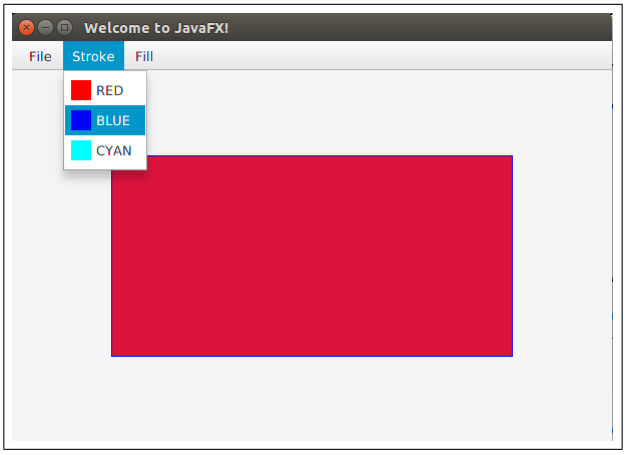
Exercice 2 (VBox, HBox, EventHandler). Définir une extension de la classe Application. Dans sa méthode `start`, créer une VBox et une nouvelle Scene de taille non spécifiée, et contenant cette VBox.

1. Ajuster les propriétés de la VBox de la manière à ce que : le contenu de la VBox soit encadré par une marge de 20 pixels sur chaque bord (*cf.* `setPadding`) ; les éléments ajoutés à cette VBox soient séparés verticalement par des espaces de 20 pixels (*cf.* `setSpacing`) ; le fond de la VBox soit une couleur uniforme différente de sa couleur par défaut, par exemple du gris comme-ci-dessus (*cf.* `setBackground`).

2. Ajouter successivement à cette VBox une dizaine de Hbox(s) contenant respectivement 1, 2, ... 10 Button(s) étiquetés par leur numéros de HBox et leur rang dans celle-ci. Ajuster les propriétés de chaque HBox de manière à ce que les éléments ajoutés soient centrés (`setAlignment`) et séparés horizontalement par des espaces de 10 pixels.

3. Associer à chaque Button un EventHandler<ActionEvent>, affichant l'étiquette d'un Button lorsque celui-ci est pressé.

Enfin, comme précédemment, associer la Scene créée à la Stage argument de la méthode `start`, ajuster la taille de la Stage à celle de la Scene, puis afficher la Stage à l'écran – cette toute dernière étape sera implicite dans les exercices qui suivent.

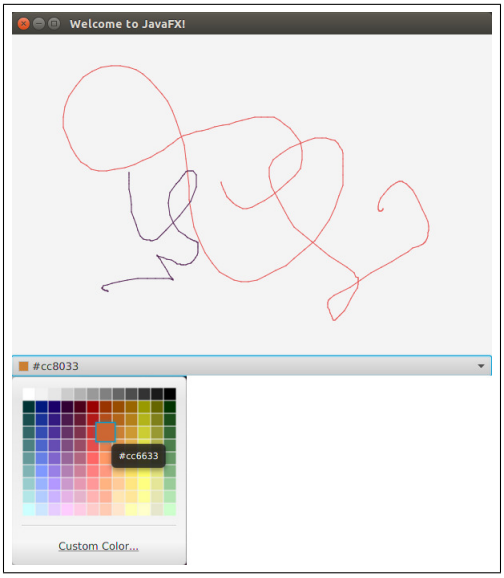


Exercice 3 (Menu, MenuBar, EventHandler). Définir une extension de la classe Application. Dans sa méthode `start`, créer une une BorderPane et une nouvelle Scene de taille 600 × 400 contenant cette BorderPane. Placer dans la région centrale de la BorderPane un Rectangle centré de taille 400 × 200.

1. Dans la région supérieure de la BorderPane, installer une MenuBar contenant trois Menu(s) : “File”, “Stroke” et “Fill”. Le menu “File” contiendra un unique MenuItem “Quit”, associé à un EventHandler<ActionEvent> permettant de quitter l’application.

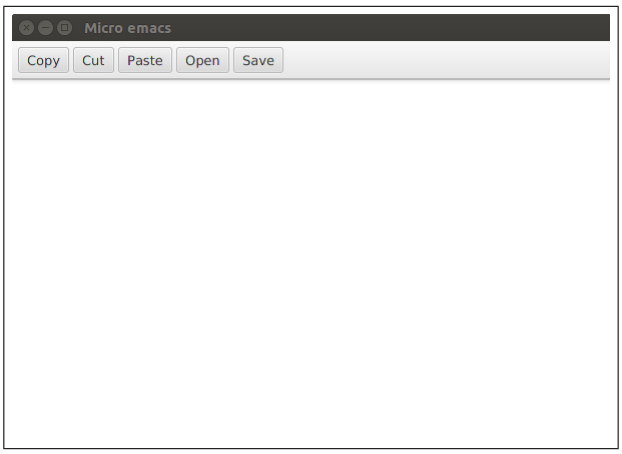
2. Dans le menu “Stroke”, placer trois MenuItem(s) étiquetés par des noms de couleurs. Associer à chaque item une icône formée d’un Rectangle de taille 10×10 rempli de la couleur choisie pour cet item (*cf.* `setGraphic`). Faire de même avec le menu “Fill”, en choisissant des couleurs différentes (variante : faire en sorte que le texte de chaque item soit le nom de sa couleur, affiché dans cette couleur).

3. Ecrire pour les items des EventHandler(s) permettant de choisir : dans le menu “Stroke” la couleur du bord du Rectangle ; dans le menu “Fill” sa couleur de remplissage.



Exercice 4 (**Canvas, EventHandler<MouseEvent>**). Définir une extension de la classe Application. Dans sa méthode `start`, créer une BorderPane et une nouvelle Scene de taille non spécifiée contenant cette BorderPane.

1. Placer dans la région centrale de la BorderPane un Canvas de dimension spécifiée et suffisamment grande.
2. Dans la région inférieure de la BorderPane, placer un ColorPicker étalé sur toute la largeur de cette région (`setMaxWidth`). Recupérer le GraphicsContext du Canvas, et l'ajouter en tant que champ.
3. Ajouter au Canvas des EventHandler<MouseEvent> permettant de dessiner dans le Canvas à l'aide de la souris (Pressed, Dragged... Dragged, Released) - à l'aide, bien sûr, du GraphicsContext extrait.
4. Associer au ColorPicker un EventHandler<ActionEvent> permettant de choisir la couleur du tracé.



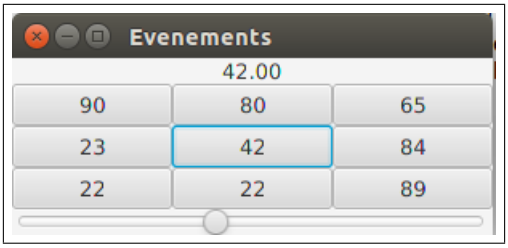
Exercice 5. Définir une extension de la classe Application. Dans sa méthode `start`, créer une BorderPane et une nouvelle Scene de taille spécifiée, contenant cette BorderPane.

1. Dans la région centrale de la BorderPane, ajouter une TextArea. Tester immédiatement son comportement, en particulier les fonctions d'édition accessibles au bouton droit.

2. Dans la région supérieure, ajouter une Toolbar, contenant cinq Button(s) étiquetés par "Copy, Cut, Paste, Open, Save". Associer aux trois premiers des EventHandler<ActionEvent>(s) invoquant simplement les méthodes correspondantes de la TextArea.

3. Associer aux boutons "Open" et "Save" deux EventHandler<ActionEvent>(s) ouvrant un FileChooser (en ouverture ou en écriture). Spécifier pour ce FileChooser un ExtensionFilter limitant la visibilité des fichiers à ceux d'extension ".txt". Les EventHandlers se chargeront respectivement de lire et de sauvegarder dans un fichier le contenu de la TextArea.

Vous pouvez améliorer l'apparence de votre application en remplaçant le texte des boutons par des icônes. Pour la dernière question, il est conseillé de se servir des classes FileReader, BufferedReader, FileWriter et BufferedWriter,

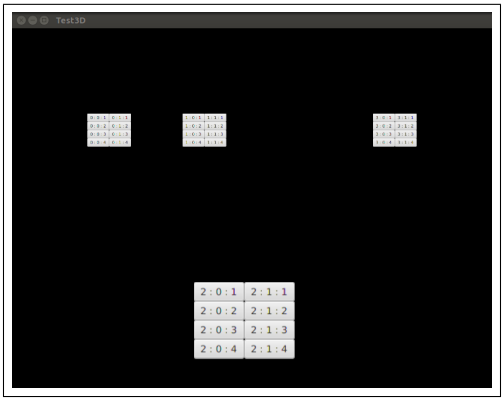


Exercice 6 (ChangeListener). Définir une extension de la classe Application. Dans sa méthode `start`, créer une BorderPane et une nouvelle Scene contenant cette BorderPane. Créer en outre les éléments suivants :

- une GridPane contenant 3×3 instances de Button. Chaque Button sera de largeur 100, étiqueté par une valeur entière aléatoire comprise entre 0 et 100. Le GridPane sera installé dans la région centrale de la BorderPane.
- un Label étiqueté par une valeur `double` affichée avec deux décimales, initialement nulle. Le label sera affiché au centre dans la région supérieure de la BorderPane (*cf.* la méthode statique `setAlignment` de BorderPane).
- un Slider allant de 0 à 100, de valeur initiale nulle, installé dans la région inférieure de la BorderPane.

Associer à chaque Button une implémentation de EventHandler<ActionEvent>; associer à la DoubleProperty du Slider (récupérable via `valueProperty()`) une implémentation de ChangeListener<Number>, de manière à obtenir le comportement suivant :

> Toute modification de la valeur du Slider sera répercutée sur le texte du Label. Toute action sur un bouton modifiera la valeur du Slider, en lui donnant pour nouvelle valeur celle de l'étiquette du Button (ce changement modifiera implicitement le texte du Label).



Exercice 7 (**Interface en 3D).** Un dernier casse-tête, pour les plus rapides. L'image ci-dessus est la fenêtre d'une application dans laquelle quatre panneaux de boutons sont placés dans un espace 3D, visualisé en perspective.

Au lancement, les 4 panneaux sont à l'arrière-plan (valeur de Z égale à 1000). Lorsque l'on clique sur l'un des boutons, et si son panneau n'est pas déjà à l'avant-plan :

- si un autre panneau est à l'avant-plan, ce panneau est d'abord remis à l'arrière-plan à son emplacement d'origine, en une demi-seconde.
- le panneau du bouton cliqué est ensuite déplacé à l'avant-plan (Z nul) en une demi-seconde (un effet supplémentaire: le panneau tourne rapidement sur son axe vertical vers la fin de ce déplacement).

Trouver le moyen d'implémenter cette application – il y a deux problèmes distincts à résoudre : celui de la visualisation, et celui des déplacements.