

Examen

Vendredi 6 juin 2023

Durée : **1h30** - Le barème est donné à titre indicatif et est susceptible d'être modifié.
Documents (un résumé de 2 pages max) autorisés. La rigueur et la propreté seront prises en compte dans l'évaluation. Le langage de programmation de cette évaluation est JavaFX.
Tous les appareils électroniques sont interdits (sauf autorisation spéciale).

Lisez bien l'énoncé – Lisez bien l'énoncé !

Vous rédigez vos réponses sur 5 copies séparées :

Copie 1 : Exercice 1 - Q1.1 et Q1.2 ; **Copie 2** : Exercice 1 - Q1.3 ; **Copie 3** : Exercice 2 ;
Copie 4 : Exercice 3 – Q3.1 ; **Copie 5** : Exercice 3 – Q3.2

Exercice 1 (7 points)

On considère l'interface suivante composée de deux boutons et d'un champ TextField :

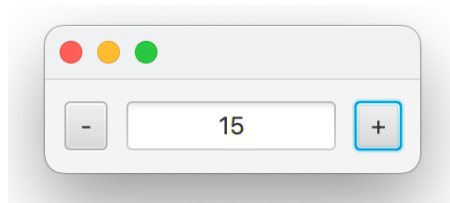


Figure 1 – Interface graphique de l'application

Le programme doit permettre de varier la valeur d'un entier à partir de deux boutons. Le premier permet de décrémenter la valeur affichée, et le second de l'incrémenter. Il est possible de modifier la valeur en saisissant directement une valeur dans le TextField. Dans ce cas, si on appuie sur le bouton « + » ou sur « - », on modifie la valeur affichée. On rappelle que les classes Button et TextField peuvent utiliser les méthodes `getText()` et `setAlignment(Pos)`.

Q1.1 (2pts) Écrire les instructions qui permettent de construire l'interface de la Figure 1 (la plus proche visuellement) – la gestion des événements est demandée dans les Q1.2 et Q1.3

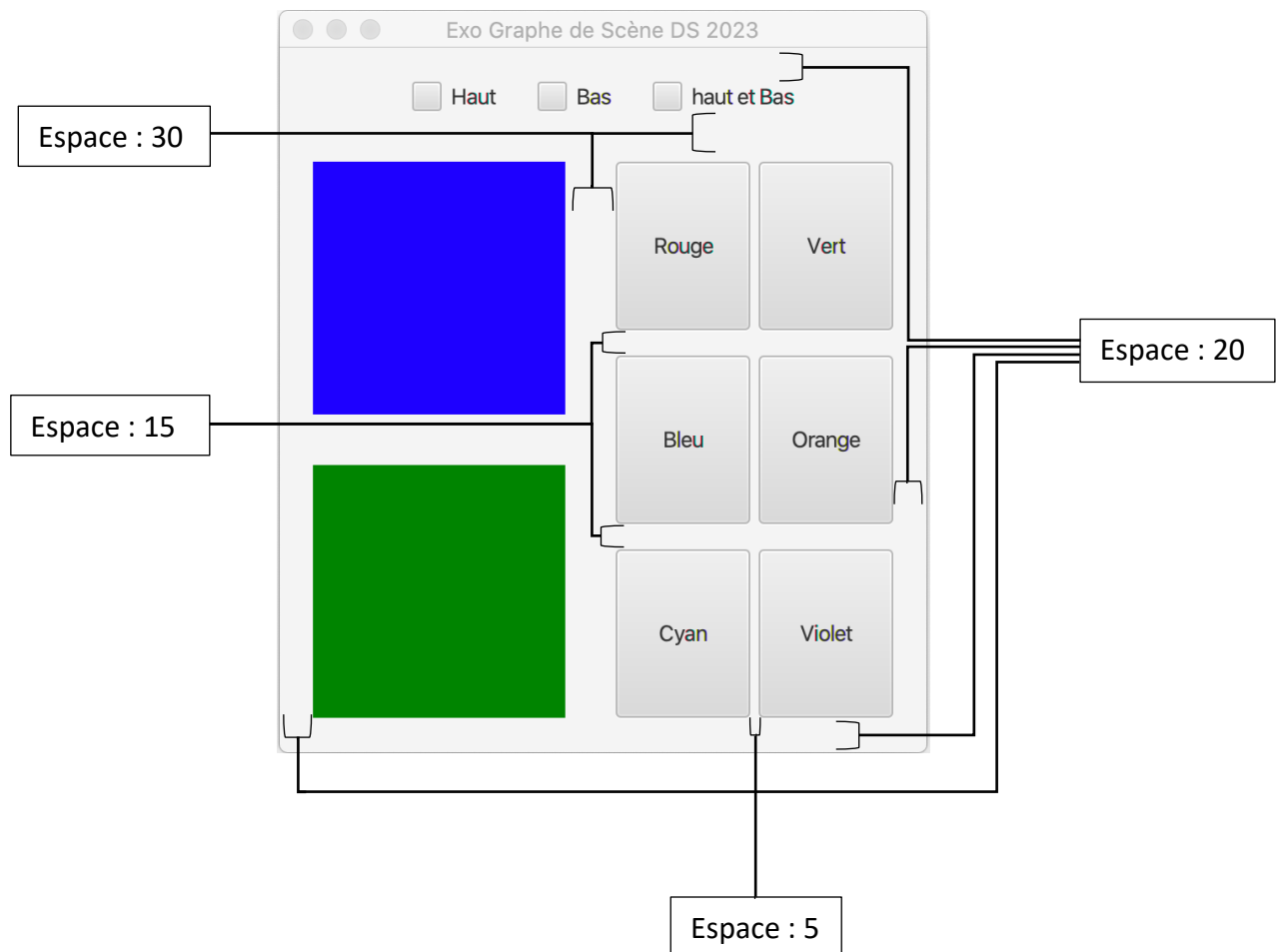
Q1.2 (2pts) Écrire les instructions permettant de contrôler la modification de la valeur avec les deux boutons en implémentant **un auditeur par bouton**, le premier avec une classe anonyme et le second avec une fonction lambda.

Passer à la copie 2

Q1.3 (3pts) Cette fois-ci on veut contrôler les deux boutons avec un seul auditeur que l'on définit par une classe interne. Écrire le code complet de cette classe interne. Écrire également les instructions qui permettent d'utiliser cet auditeur (connecter l'auditeur aux boutons).

Exercice 2 (5 points)

On considère l'interface graphique de la figure ci-dessous. Elle est composée de 3 boîtes à cocher, de 6 boutons de taille 80*100 et de deux rectangles (un bleu et un vert) de taille 150*150.

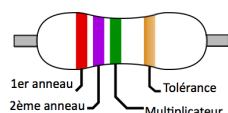


Q2.1 Faire le graphe de scène de cette interface graphique.

Q2.2 Donner les instructions pour **construire itérativement** les 6 boutons et les placer correctement dans l'interface. Le conteneur principal devra être aussi construit. Les espacements précisés sur le graphique doivent être respectés.

Exercice 3 (8 points)

Une résistance est un composant électronique qui se présente sous la forme d'un petit cylindre coloré. Quatre anneaux de couleurs sur ce composant informent sur la valeur de la résistance (exprimée en « Ohms »). En effet, le décodage des couleurs permet de trouver la valeur de la résistance comme le montre la figure suivante :



0	1	2	3	4	5	6	7	8	9
Noir	Marron	Rouge	Orange	Jaune	Vert	Bleu	Violet	Gris	blanc

Figure 2. Décodage des valeurs d'une résistance à partir de dix couleurs.

Afin de simplifier, nous considérons qu'une résistance **ne contient que trois anneaux** :

1. Le premier anneau indique le premier chiffre.
2. Le deuxième anneau indique le deuxième chiffre.
3. Le troisième anneau est un coefficient multiplicateur.

Dans la figure ci-dessus, nous avons un exemple de résistance (rouge, violet et vert). Le tableau indique la correspondance entre les couleurs et les numéros. Pour cet exemple (rouge 2, violet 7 et vert 5), la valeur de la résistance est : 2700000 (27×10^5) Ohms.

On propose de programmer une application qui permet de retrouver la valeur de la résistance à partir des couleurs. La figure suivante montre l'interface graphique initiale, puis son évolution.

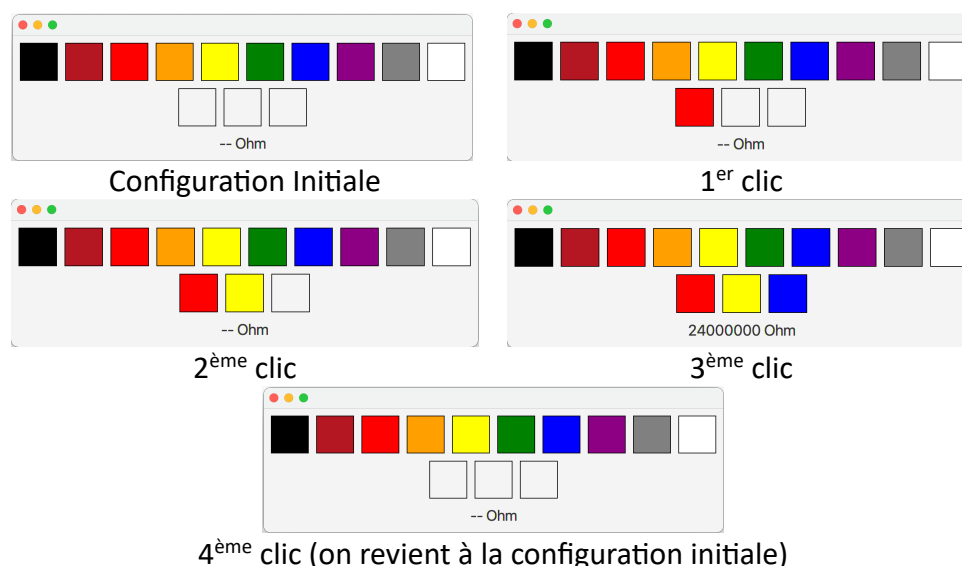


Figure 3. L'interface graphique de l'application de l'exercice 2 et son évolution.

Initialement, l'interface contient une palette des couleurs prédéfinies. La partie inférieure représente la résistance qui contiendra petit à petit les 3 couleurs choisies par l'utilisateur en cliquant sur la couleur de la palette. La valeur de résistance ne s'affiche que lorsque les 3 couleurs ont été sélectionnées. Si l'utilisateur clique sur une couleur une 4ème fois, la partie inférieure se réinitialise et la valeur de la résistance se vide (--Ohm), etc.

Indications :

- On utilisera la classe **Rectangle** pour l'affichage des couleurs (vous pouvez utiliser les méthodes : `void setFill (Color)`, `Color getFill ()` et `setStroke(Color)` de la classe `Rectangle`).
- Pour l'affichage de la valeur de la résistance, vous pouvez utiliser le composant graphique **Label** et la méthode `setText(String s)`
- Pour l'alignement des éléments graphiques d'un conteneur, vous pouvez utiliser la méthode `void setAlignment(Pos p)`
- On suppose que le tableau des couleurs est défini comme attribut que vous utiliserez sans le réécrire dans votre copie :

```
private static final Color[] COLORS =  
{Color.BLACK, Color.BROWN, Color.RED, Color.ORANGE, Color.YELLOW,  
Color.GREEN, Color.BLUE, Color.PURPLE, Color.GRAY, Color.WHITE};
```

- On vous conseille de sauvegarder les choix des utilisateurs dans un tableau.
- La méthode `static double pow(double a, double b)` de la classe `Math` donne **a** à la puissance **b**.

Q3.1 Programmer l'interface graphique (la plus proche visuellement de la figure 3). Pour le moment on ne prend pas en compte la gestion des événements. On ne vous demande pas le squelette principal d'application, mais uniquement les instructions que l'on met habituellement dans la méthode `start()` pour pouvoir répondre à cette question.

Passer à la copie 5

Q3.2 Programmer la gestion des événements où on associe autant d'auditeurs que de couleurs, mais chaque auditeur se base sur la méthode `gestionChoix(int numCol)`, où `numCol` est l'indice de la couleur dans la palette, qu'on vous demande d'écrire. Indiquer ce qui doit être modifié dans Q3.1 pour intégrer efficacement l'auditeur.