

SAE S2.01 – Développement d'une application avec une IHM**TP n°4 *WindowBuilder***

Manipulation de composants graphiques



Dans ce TP, nous allons apprendre comment utiliser certains composants graphiques swing, leur associer des traitements en fonction d'événements.

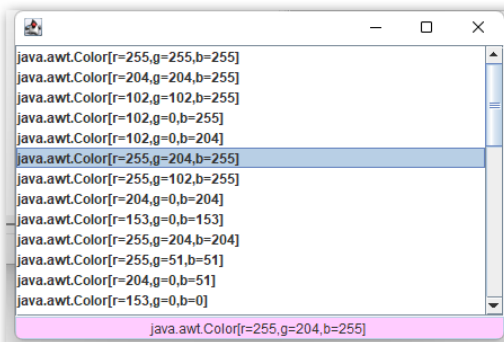
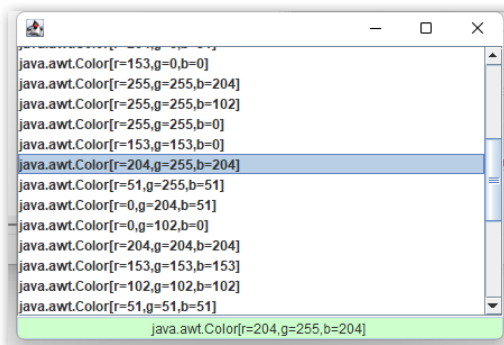
Pour chaque exercice, l'interface graphique sera construite à partir de l'outil JBuilder et le code source devra être modifié pour faire le traitement souhaité.

1. *JList* associée à un *JScrollPane*

Description de l'application

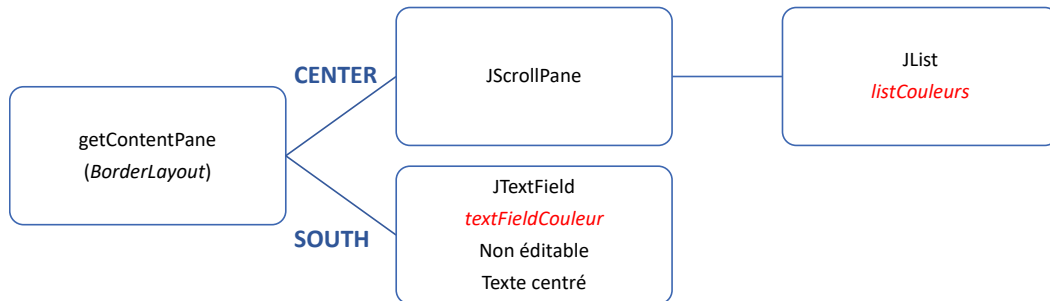
Dans cet exercice, nous souhaitons :

- afficher une liste déroulante contenant des valeurs RVB de couleurs,
- sélectionner une ligne de cette liste pour l'afficher dans un champ textuel utilisant comme couleur de fond la couleur sélectionnée.

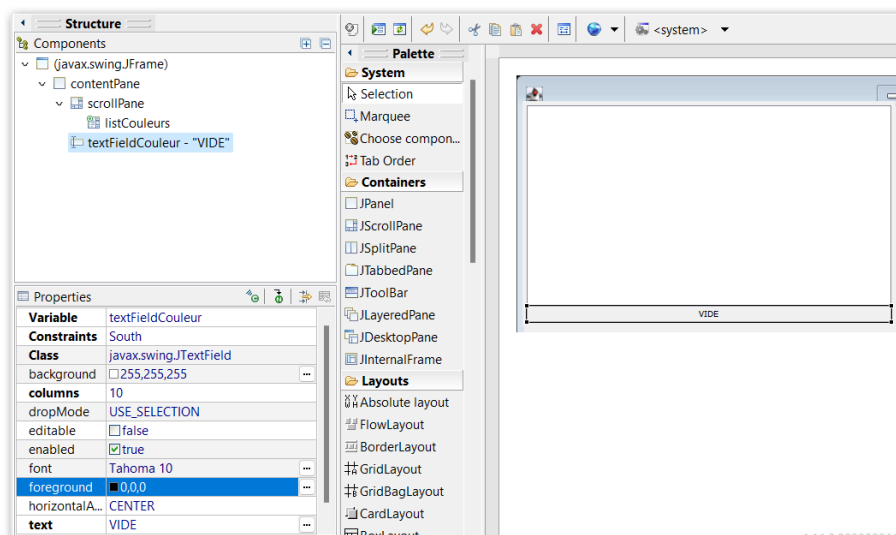


Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante. On prendra soin de renommer la liste et le champ textuel et de modifier les propriétés.



La *JList* a été insérée dans un *JScrollPane* de telle sorte à disposer d'un ascenseur horizontal et vertical pour faire défiler les items d'une liste trop grande pour contenir entièrement dans la fenêtre.



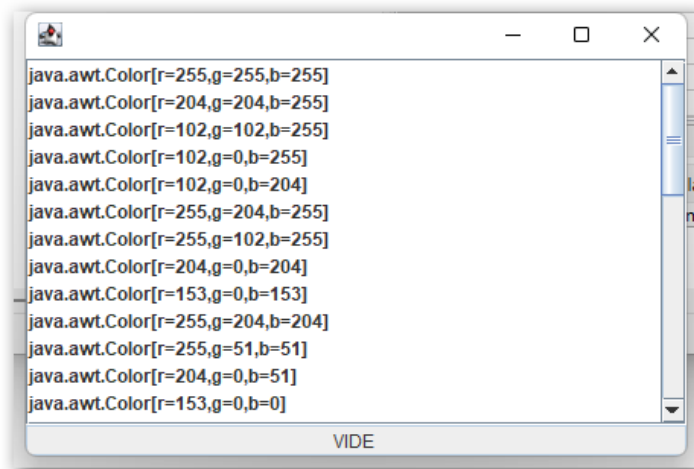
- b) Testons tout de suite. Dans la vue *Source*, rajouter le tableau suivant en attribut.

```

public static final Color[] COULEURS= {

    new Color(255,255,255), new Color(204,204,255), new Color(102,102,255),
    new Color(102,0,255), new Color(102,0,204), new Color(255,204,255),
    new Color(255,102,255), new Color(204,0,204), new Color(153,0,153),
    new Color(255,204,204), new Color(255,51,51), new Color(204,0,51), new Color(153,0,0),
    new Color(255,255,204), new Color(255,255,102), new Color(255,255,0),
    new Color(153,153,0), new Color(204,255,204), new Color(51,255,51),
    new Color(0,204,51), new Color(0,102,0), new Color(204,204,204),
    new Color(153,153,153), new Color(102,102,102), new Color(51,51,51),
    new Color(255,204,102), new Color(255,204,102), new Color(255,153,0),
    new Color(153,102,0), new Color(204,255,255), new Color(102,204,255),
    new Color(0,102,255), new Color(0,0,204), new Color(0,0,0),
    new Color(51,255,255), new Color(0,204,204), new Color(0,153,153)
};
  
```

Puis, mettre en paramètre du constructeur de la liste le tableau *COULEURS*. Aller dans la vue *Design* pour voir la nouvelle interface et vérifier par un test rapide que les items de la liste défilent correctement dans la fenêtre. Chaque item de la liste correspond à l'appel à *toString()* de la couleur.



- c) Dans la vue *Design*, faire un clic droit sur la liste, puis sélectionner :

Add event handler > mouse > mousePressed

Cela signifie qu'un gestionnaire d'événement va être ajouté à la liste. Ce gestionnaire est chargé d'intercepter les événements de la souris (mouse) et d'appeler la méthode *mousePressed()* chaque fois que la souris est enfoncée. Cette action provoque l'ouverture de la vue *Source* pour compléter le traitement à réaliser. Modifier le code jusqu'à ce que l'application fasse le traitement souhaité décrit en début de section.

Quelques méthodes :

Classe	Méthode	Description
JList	<i>Color getSelectedValue()</i>	Récupère la valeur de l'item sélectionné dans la liste (dans notre contexte une couleur)
JList	<i>int getSelectedIndex()</i>	Récupère le rang de l'item sélectionné dans la liste
JTextField	<i>void setText(String texte)</i>	Positionne un texte dans un champ textuel
JTextField	<i>void setBackground(Color couleur)</i>	Positionne une couleur de fond dans un champ textuel

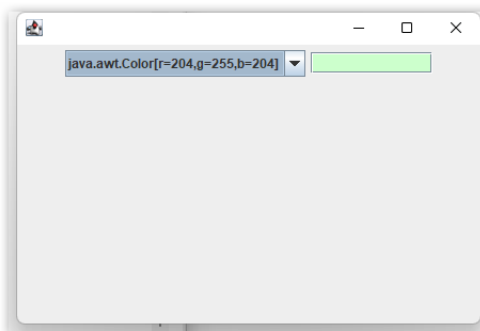
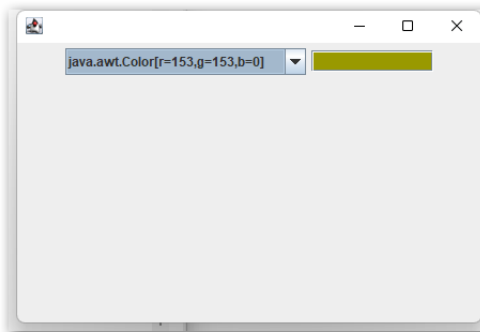
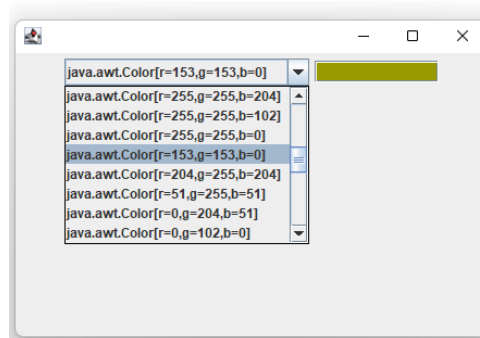
- d) Dans la vue *Source*, changer *mousePressed* par *mouseClicked*. Qu'est-ce qui change à l'exécution ?
- e) Enfin extraire la méthode *construireContrôleurListe()* qui associe l'écouteur de type *MouseListener* à la liste.

2. JComboBox

Description de l'application

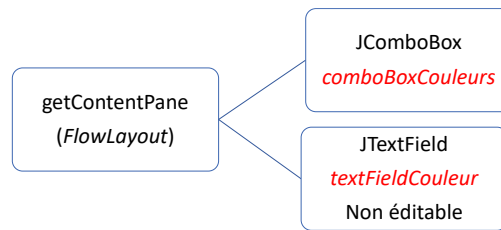
Dans cet exercice, nous souhaitons :

- afficher une combo box contenant des valeurs RVB de couleurs,
- sélectionner un item de la combo box pour mettre à jour la couleur de fond d'un champ textuel à partir de la couleur sélectionnée.



Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante. On prendra soin de renommer la liste et le champ textuel et de modifier les propriétés.



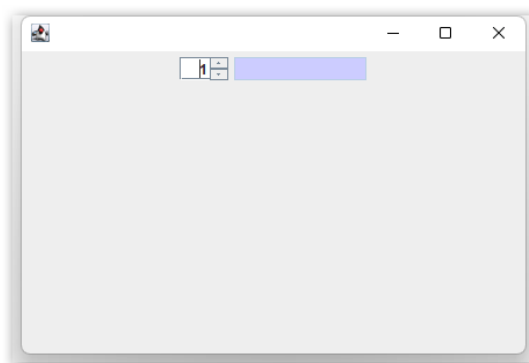
- b) Dans la vue *Source*, rajouter en attribut le tableau *COULEURS* de l'exercice précédent. Aller dans la vue *Design* pour voir la nouvelle interface et vérifier par un test rapide que les items de la combo box ont bien été renseignés. Chaque item correspond à l'appel à *toString()* de la couleur correspondante.
- c) Nous souhaitons à présent intercepter les événements sur la combo box pour qu'à chaque changement d'état soit affichée, dans le *JTextField*, la couleur associée à celle sélectionnée dans la combo box. Pour cela, dans la vue *Design*, ajouter le bon gestionnaire d'événement à la combo box et la méthode qu'il faudra implémenter dans la vue *Source*. Ne pas oublier d'extraire la méthode *construireControleurComboBox()*. Mettre au point le code jusqu'à obtenir le traitement souhaité.

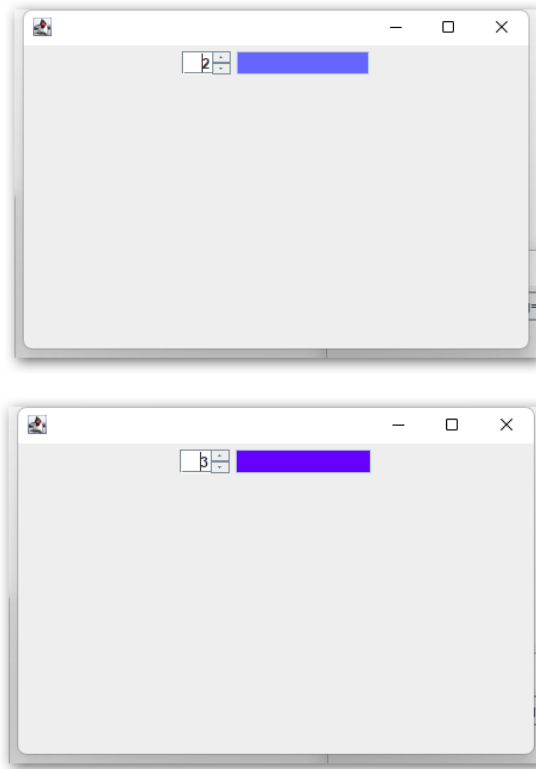
3. Un JSpinner

Description de l'application

Dans cet exercice, nous souhaitons :

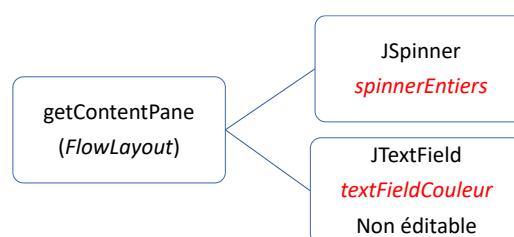
- afficher un spinner (bouton fléché) faisant défiler des valeurs entières (en avant et en arrière), chacune d'elles associée à une couleur,
- afficher dans un champ textuel la couleur associée à la valeur du spinner.



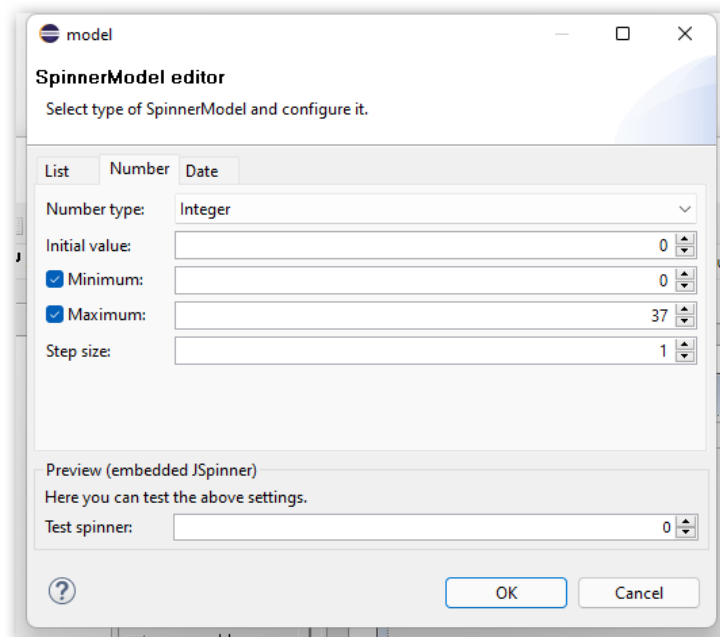


Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante. On prendra soin de renommer le spinner et le champ textuel et de modifier les propriétés si nécessaire.



- b) Dans la vue *Design*, sélectionner le spinner et choisir la propriété « *model* ». Cette propriété définit la valeur initiale du spinner, sa valeur minimum, sa valeur maximum, son pas de variation, son type. Modifier les valeurs comme dans l'image ci-dessous et tester dans la même fenêtre avec *Test Spinner*. S'amuser à mettre des dates à la place des entiers (onglet *Date*) pour voir ce qui se passe. Remettre les valeurs ci-dessous.



- c) Dans la vue *Source*, rajouter en attribut le tableau *COULEURS* des exercices précédents, puis remplacer la valeur 37 du modèle par la taille du tableau *COULEURS* (*COULEURS.length*). Tester le spinner. Il devrait s'arrêter à la valeur 37. C'est normal, il y a 37 valeurs dans le tableau *COULEURS* !
- d) Nous souhaitons à présent intercepter les événements sur le spinner pour qu'à chaque changement de valeur soit affichée dans le *JTextField* la couleur de rang valeur du tableau *COULEURS*. Dans la vue *Design*, ajouter le bon gestionnaire d'événement au spinner et la méthode qu'il faudra implémenter dans la vue *Source*. Ne pas oublier d'extraire la méthode *construireContrôleurSpinner()*.

Méthode utile :

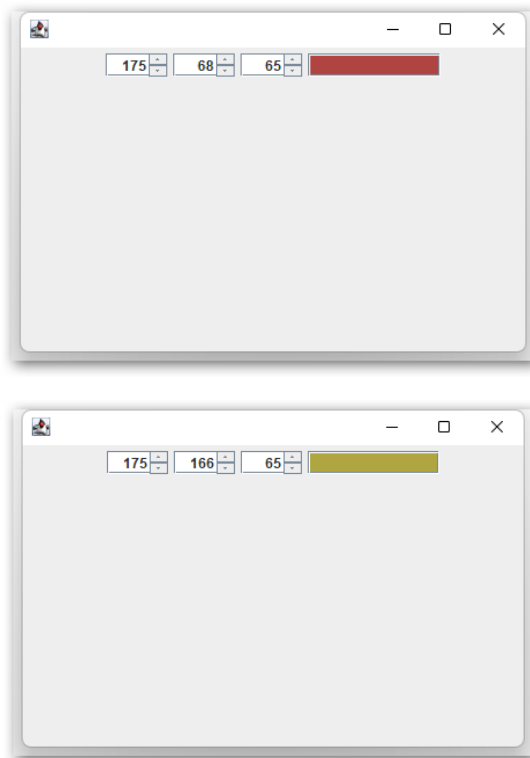
Classe	Méthode	Description
JSpinner	<i>Object getValue()</i>	Renvoie la valeur affichée par le spinner (à transtyper si nécessaire)

4. Trois JSpinners

Description de l'application

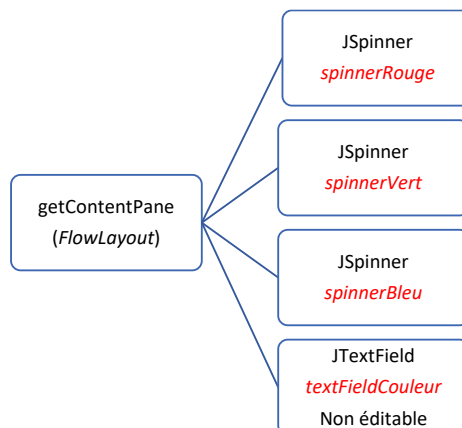
Dans cet exercice, nous souhaitons :

- afficher trois spinners faisant défiler des valeurs entières (en avant et en arrière), chacune d'elles associée à une nuance de couleur RVB,
- afficher dans un champ textuel la couleur correspondant aux 3 nuances RVB contenues dans les spinners.



Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante. On prendra soin de renommer les spinners et le champ textuel et de modifier les propriétés si nécessaire.



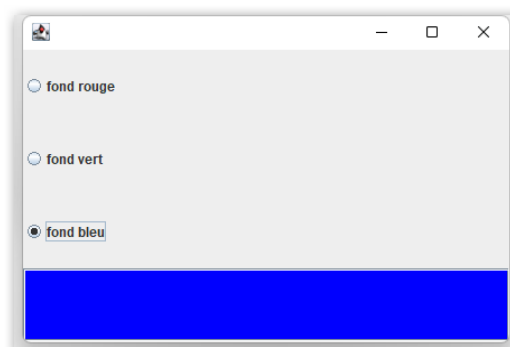
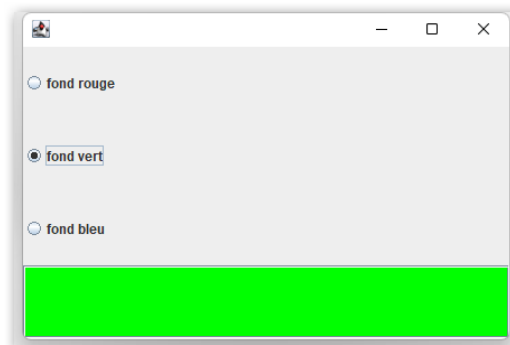
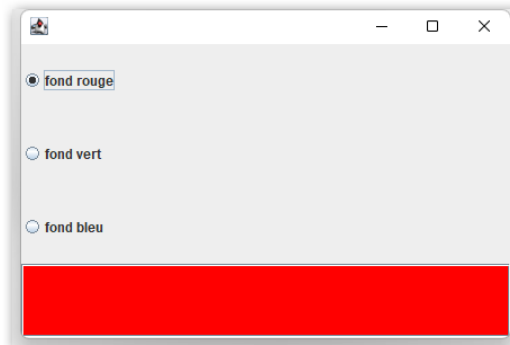
- b) Dans la vue *Design*, modifier la propriété « *model* » de chaque spinner sachant que chaque valeur RVB est un entier compris entre 0 et 255, puis ajouter à chaque spinner un gestionnaire d'événement pour écouter les changements de valeurs. Dans la vue *Source*, programmer chaque écouteur pour mettre à jour la couleur de fond du champ textuel en fonction des valeurs prises par les 3 spinners. On n'oubliera pas d'extraire les méthodes *construireContrôleurSpinnerRouge()*, *construireContrôleurSpinnerVert()* et *construireContrôleurSpinnerBleu()*.

5. RadioButton

Description de l'application

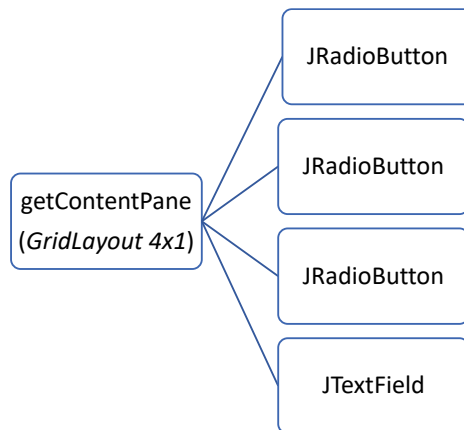
Dans cet exercice, nous souhaitons :

- afficher trois boutons radio,
- afficher dans un champ textuel la couleur correspondant au bouton radio sélectionné.

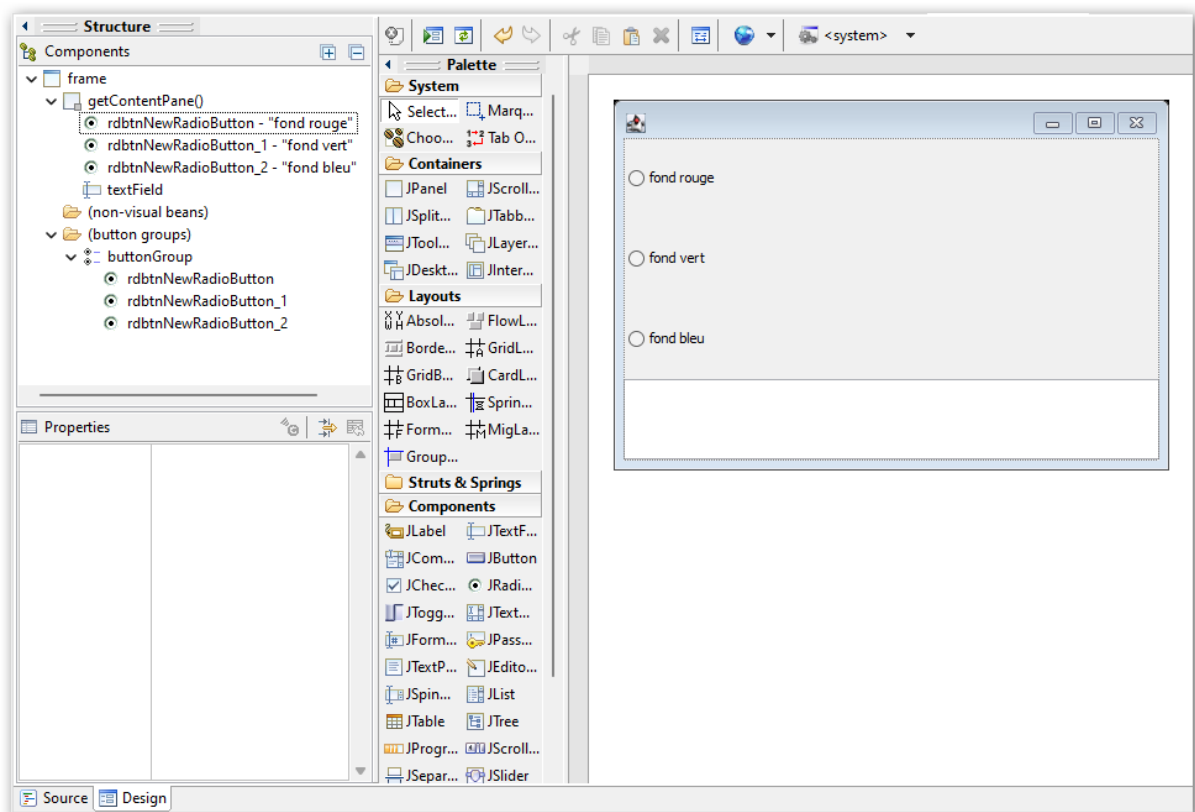


Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante.



- a) Tester l'application en cliquant sur les boutons radio. Ces derniers sont indépendants les uns des autres ; il est donc possible d'en sélectionner plusieurs en même temps. Or, nous souhaiterions que la sélection d'un bouton entraîne automatiquement la désélection des autres. Pour cela, il faut grouper les boutons de la manière suivante : dans la vue *Design* sélectionner les 3 boutons radio simultanément, puis clic droit pour sélectionner *set ButtonGroup > New standard*. Un groupe de boutons a été créé comme la montre la figure ci-dessous. Vérifier maintenant le nouveau comportement.



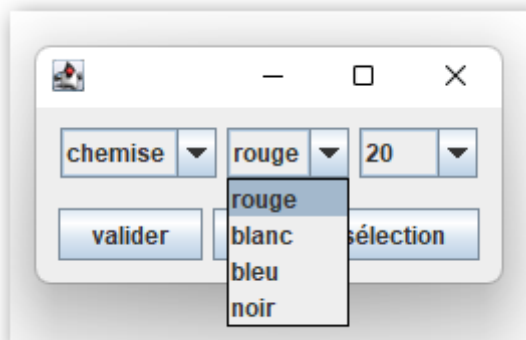
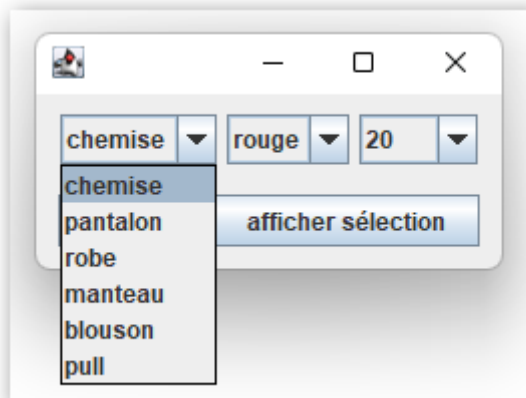
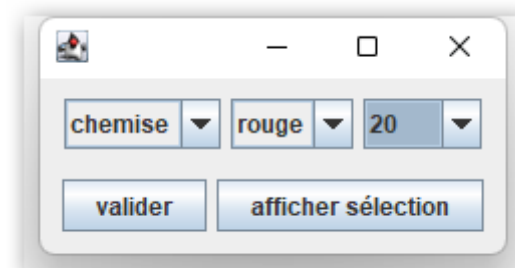
- c) Ajouter à présent un gestionnaire d'événement sur chaque bouton pour que le fond du champ textuel prenne la couleur associée au bouton radio sélectionné. Enfin, extraire les 3 méthodes associées à la construction des contrôleurs.

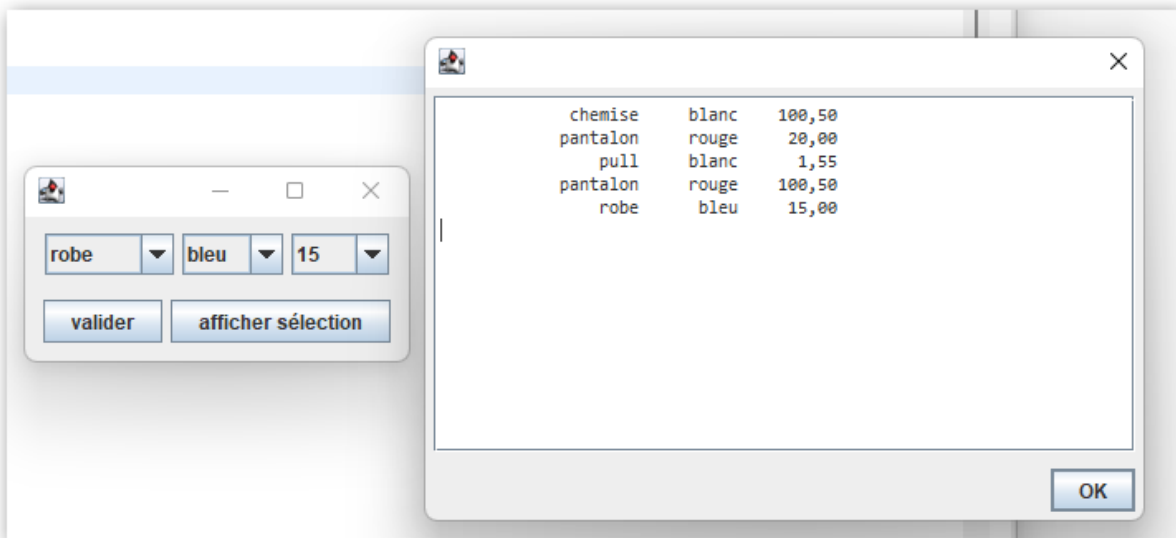
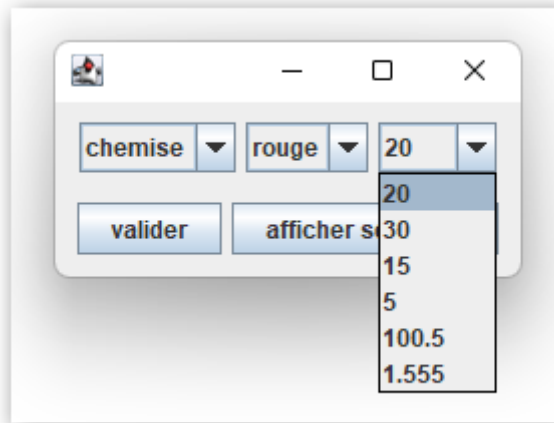
6. JTextPane et JDialog

Description de l'application

Dans cet exercice, nous souhaitons :

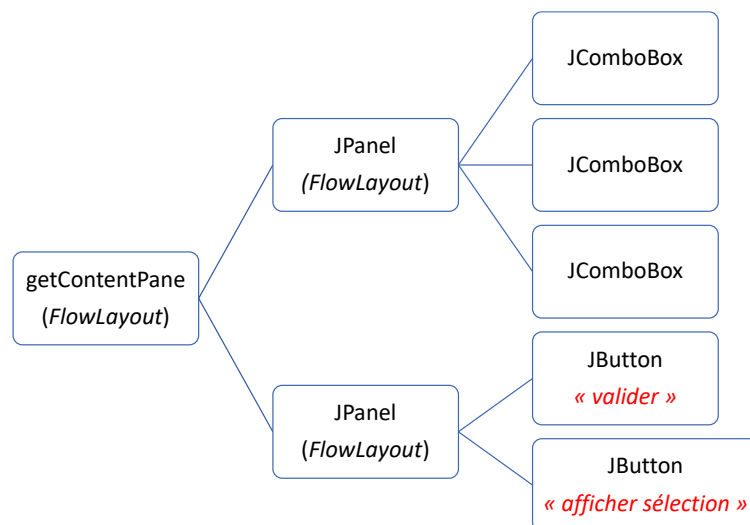
- Afficher 3 comboBox, la première proposant des noms de produits, la deuxième des couleurs et la troisième des prix,
- Par l'appui sur le bouton « *valider* », sauvegarder en historisant dans une chaîne de caractères formatée les 3 valeurs sélectionnées,
- Afficher dans une nouvelle fenêtre de dialogue toutes les sélections effectuées correctement formatées par l'appui sur le bouton « *afficher sélection* ».





Création de l'application

- a) Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante.



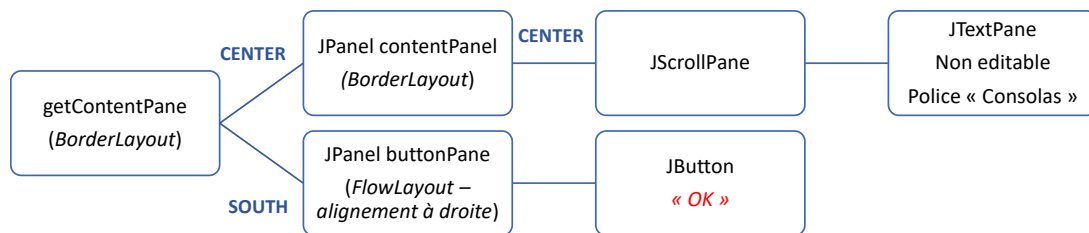
- b) Dans la vue *Design*, à l'aide de la propriété « *model* » valoriser chaque combo box comme dans l'exemple qui a été fourni et réduire la taille de la fenêtre. Tester l'application.
- c) Associer au bouton « *valider* » un gestionnaire d'événement pour qu'à chaque clic, il enregistre dans une chaîne de caractères, définie en attribut, les valeurs des 3 combo box sélectionnées en formatant de la manière suivante : le nom du produit cadré à droite sur 20 positions, la couleur cadrée à droite sur 10 positions et le prix sur 10 positions arrondi à 2 décimales. Tester de telle sorte à afficher dans la console le contenu de la chaîne produite à chaque clic. Cette chaîne doit conserver l'historique de toutes les valeurs sélectionnées en ajoutant « *\n* » à la fin de chaque ligne comme dans la figure ci-dessous :

```

Console
\app13 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (17 mai 2022, 17:08:25)
      robe      bleu      30,00€
      robe      bleu      30,00€
    manteau    bleu      15,00€
      robe      bleu      30,00€
    manteau    bleu      15,00€
    pantalon   blanc     100,50€
      robe      bleu      30,00€
    manteau    bleu      15,00€
    pantalon   blanc     100,50€
    chemise    rouge      30,00€
      robe      bleu      30,00€
    manteau    bleu      15,00€
    pantalon   blanc     100,50€
    chemise    rouge      30,00€
    blouson    rouge       1,55€
  
```

Remarques :

- pour le formatage, on utilisera la méthode *String.format()* déjà utilisée dans le *TPn°17* pour l'affichage des scores d'un match de tennis. Vous trouverez aussi des explications ici : https://koor.fr/Java/Tutorial/java_system_out_printf.wp ou encore ici : <https://fr.tutorialcup.com/java/string-format-java.htm>
 - le prix sélectionné dans la combo box devra être converti en float pour pouvoir faire le formatage de réel. Pour convertir une chaîne en réel, on pourra par exemple utiliser la méthode *Float.parseFloat(String valeur)*
- d) Après la mise au point de l'affichage de la chaîne de caractères, nous allons créer une fenêtre de dialogue. C'est elle qui servira à afficher la chaîne de caractères à la place de la console. Dans le menu d'eclipse, sélectionner *File > New > Other > Swing Design > JDialog*, puis créer dans cette fenêtre de dialogue les composants conformément à l'arborescence ci-dessous. Il faudra notamment supprimer le bouton « *Cancel* » pour ne garder que le bouton « *Ok* » :



- e) Ajouter un gestionnaire d'événement au bouton « OK » pour pouvoir refermer la fenêtre de dialogue. Pour cela, il suffit d'appeler la méthode `dispose()` sur la fenêtre de dialogue. Tester le bon fonctionnement.
- f) A présent, dans la vue *Source*, supprimer la méthode `main()` puisque cette fenêtre de dialogue doit être ouverte depuis la fenêtre précédente contenant les combo box. Ajouter un constructeur admettant une chaîne de caractères en paramètre. Cette chaîne définit le texte à afficher dans le *JTextPane*. Pour associer un texte à un *JTextPane*, utiliser la méthode `void setTexte(String texte)`.
- g) Il ne reste plus qu'à programmer le bouton « Afficher Sélection » de la fenêtre précédente pour qu'il ouvre la fenêtre de dialogue afin d'afficher la chaîne de caractères formatée par le code suivant, et tester. Dans le code fourni ci-dessous la classe associée à la fenêtre de dialogue s'appelle *FenêtreDialogue* et la chaîne formatée s'appelle *chaîne* :

```

public void actionPerformed(ActionEvent e) {
    FenêtreDialogue f = new FenêtreDialogue(chaîne);
    f.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    f.setVisible(true);
}

```

- h) Enfin pour finir, ne pas oublier de faire les *extract methods* pour tous les gestionnaires d'événement des 2 classes créées.

Remarque :

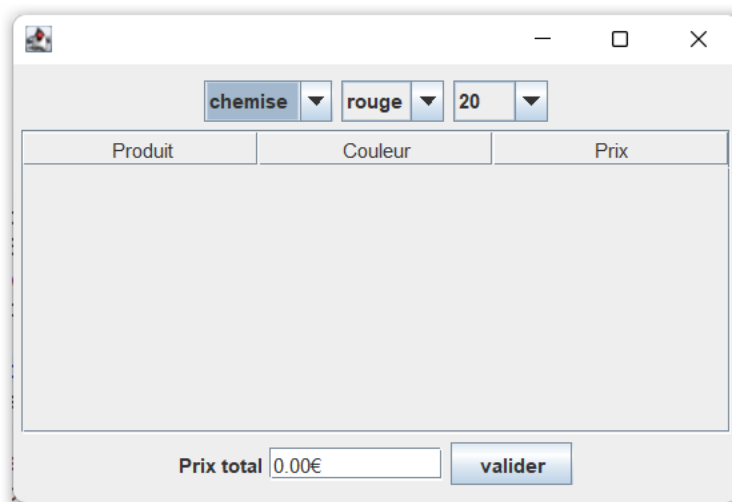
Lorsqu'on conçoit une application multi-fenêtre, il est souvent nécessaire que la fenêtre courante transmette des informations à la nouvelle fenêtre. Dans ce cas ces informations sont transmises via le constructeur de la nouvelle fenêtre.

7. JTable

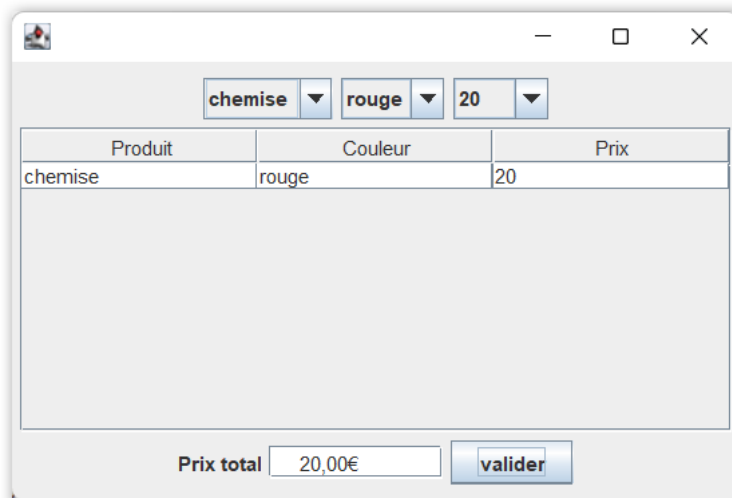
Description de l'application

Dans cet exercice, nous souhaitons :

- Afficher 3 comboBox, la première proposant des noms de produits, la deuxième des couleurs et la troisième des prix, une table de 3 colonnes vide et un textfield contenant la valeur « 0,0€ ».
- Par l'appui sur le bouton « *valider* », ajouter une ligne dans la table associée au 3 valeurs courantes des comboBox, calculer le prix total des articles contenus dans la table.



The screenshot shows a Java Swing window with a title bar. Inside, there are three JComboBox components at the top, each with a dropdown arrow. The first dropdown shows 'chemise', the second shows 'rouge', and the third shows '20'. Below these is a table with three columns: 'Produit', 'Couleur', and 'Prix'. The table is currently empty. At the bottom, there is a text field labeled 'Prix total' containing '0.00€' and a button labeled 'valider'.



The screenshot shows the same Java Swing window after the 'valider' button has been clicked. The three JComboBox components remain the same. The table now has one row of data: 'chemise' under 'Produit', 'rouge' under 'Couleur', and '20' under 'Prix'. The 'Prix total' text field now displays '20,00€'.

Produit	Couleur	Prix
chemise	rouge	20
robe	blanc	15

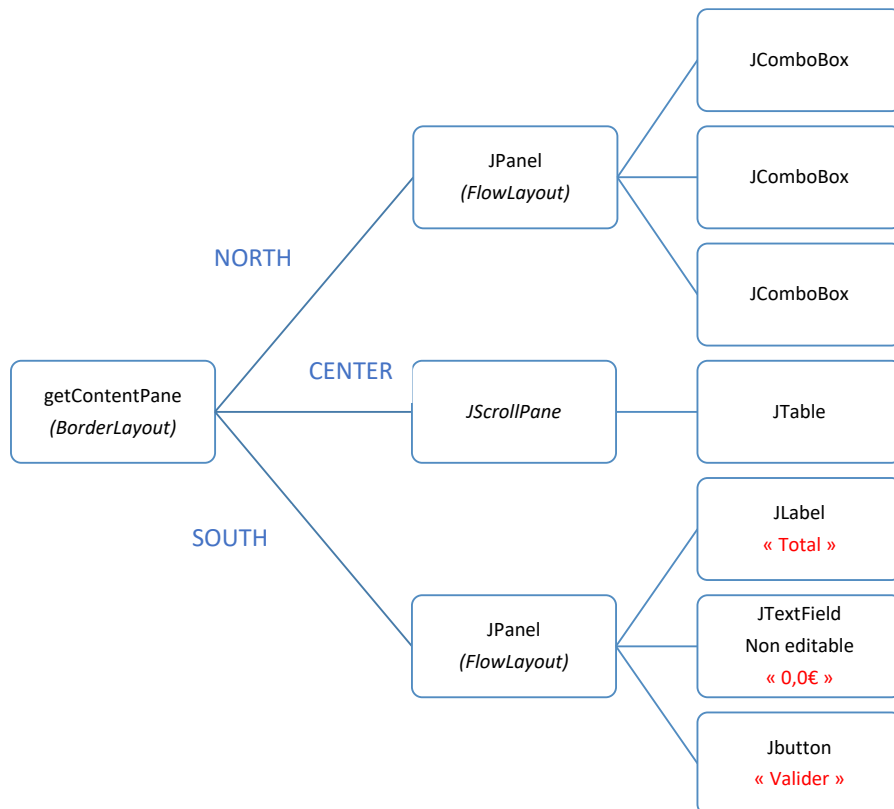
Prix total: 35,00€ valider

Produit	Couleur	Prix
chemise	rouge	20
robe	blanc	15
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30
robe	bleu	30

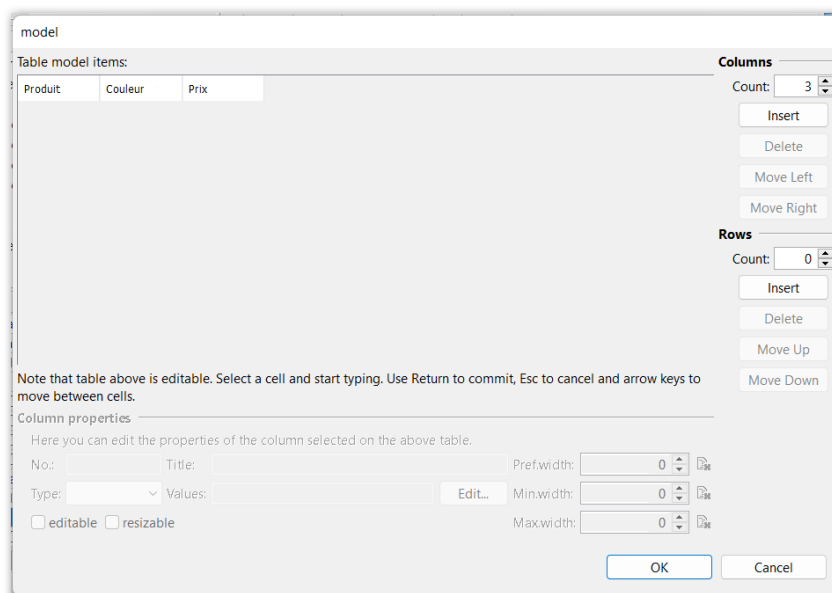
Prix total: 395,00€ valider

Création de l'application

- Créer une nouvelle application *WindowBuilder* conformément à l'arborescence suivante.



- b) Dans la vue *Design*, à l'aide de la propriété « *model* » valoriser chaque combo box avec les mêmes valeurs que dans l'exercice précédent.
- c) Dans la vue *Design*, à l'aide de la propriété « *model* », créer les 3 colonnes de la *JTable* :
- « insert » pour ajouter 1 colonne puis dans les propriétés de la colonne créée, saisir le titre de la colonne,
 - Laisser le nombre de colonnes à 0.



- d) Associer au bouton « *valider* » un gestionnaire d'événement pour qu'à chaque clic, soit ajouté une ligne à la *JTable* contenant les items sélectionnés dans les 3 *ComboBox*. Actualiser le *JTextField* pour mettre à jour le prix total des articles.

Remarques :

- Tous les accès aux données d'une *JTable* se font à partir de son modèle. Pour le récupérer :

```
DefaultTableModel model = (DefaultTableModel) table.getModel();
```

- Les données d'une *JTable* sont stockées dans un tableau à 2 dimensions d'objets :

```
Object[][] data
```

- Pour insérer une ligne dans une *JTable* :

```
model.addRow(new Object[] { valeur1, valeur2, ..., valeurn});
```

- Pour connaître le nombre de ligne d'une *JTable* :

```
model.getRowCount()
```

- Pour accéder à une donnée de la *JTable* (i désigne l'indice de ligne, j l'indice de colonne) :

```
model.getValueAt(i, j)
```