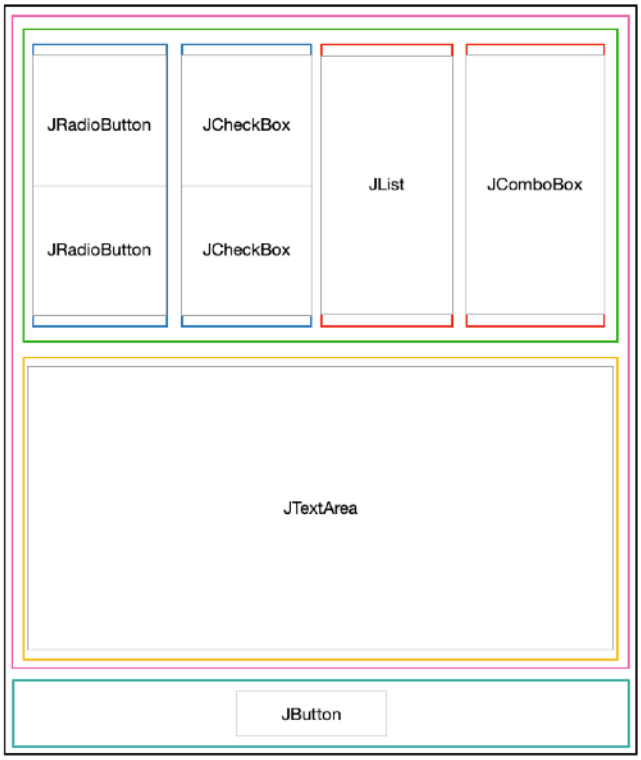
PROST Quentin
Groupe IE2-022

Rapport technique n°1

Partie 1 : Création de l'application « ApprendLesComposants »

Première étape, création d'un schéma pour l'organisation des composants.

Avant de commencer le travail sur NetBeans, il a été nécéssaire de dessiner un schéma afin d'organiser au mieux les différents composants nécessaires, puis de faire une arborescence.



JFrame ApprendLesComposants BorderLayout
- **JPanel jPanelC GridLayout(2, 1) CENTER**
  - **JPanel jPanelC1 GridLayout(1, 4)**
    - **JPanel jPanelC11 GridLayout(2, 1)**
      - JRadioButton jRadHo "Hoefler Text, 14"
      - JRadioButton jRadHe "Helvetica, 16"
    - **JPanel jPanelC12 GridLayout(2, 1)**
      - JCheckBox jChkB "Bleu"
      - JCheckBox jChkM "Magenta"
    - **JPanel jPanelC13 AbsoluteLayout**
      - JList jListPlantes
    - **JPanel jPanelC14 AbsoluteLayout**
      - JComboBox jCBoxAnimaux
  - **JPanel jPanelC2 AbsoluteLayout**
    - JTextArea Edition
- **JPanel jPanelS FlowLayout SOUTH**
  - JButton Effacer "Effacer"

J'ai décidé d'opter pour une disposition en BorderLayout afin d'avoir le bouton « effacer » tout en bas de la page (jPanelS, SOUTH) et le reste des composants au milieu de la fenêtre (jPanelC, CENTER).

Ensuite, je sépare les différents composants de « contrôle » de la zone d'édition. Pour cela j'utilise un GridLayout(2, 1) pour avoir deux lignes. En haut, nous aurons les RadiosButton, les CheckBox et les listes (dans le jPanelC1), puis en bas la zone d'édition (jPanelC2).

J'utilise ensuite un JPanel divisé en 4 colonnes (jPanelC1) pour avoir les JRadioButton (jPanelC11), les JCheckBox (jPanelC12), la JList (jPanelC13) et enfin la JComboBox (jPanelC14).

Les JCheckBox et JRadioButton utilisent la même disposition, c'est à dire en GridLayout(2, 1) pour avoir deux lignes sur une seule colonne.

La JList et la JComboBox utilisent aussi la même disposition, c'est à dire en GridLayout(1, 1), afin qu'elles occupent tout l'espace possible dans leur panneau.

Pour la JTextArea, elle se trouve dans le jPanelC2, elle aussi en GridLayout(1, 1).

Je JButton Effacer est seul dans le jPanelS en FlowLayout.

J'ai utilisé NetBeans pour construire l'interface graphiquement en plaçant les éléments tout en suivant l'arborescence.

Deuxième étape : JCheckBox, JRadioButton et JButton

Après avoir terminé l'interface, je m'occupe des JRadioButton, JCheckBox et du JButton. Tout d'abord, je modifie les noms de variable des composants (jRadHo, jRadHe, jChkB, jChkM et Effacer).
J'associe à chaque composant, un événement en faisant un clic gauche sur celui-ci, puis Events, Action, ActionPerformed. Et j'écris le code correspondant à chaque composants (voir commentaire sur le programme).

Le JButton Effacer, vide la zone d'édition lorsqu'il est cliqué.

Troisième étape : JList et JComboBox.

Pour commencer, je m'occupe de la JComboBox. Je créer une méthode « initjCBoxAnimaux » qui sera appelé dans le constructeur. Dans cette méthode, je remplis le liste grâce à « addItem » avec les différents noms d'animaux.
Ensuite, pour récupérer l'item contenu dans la liste lorsque l'utilisateur clique dessus, on associe le composant à un événement (même technique que vu précédemment), puis dans cette méthode, on modifie la zone d'édition en y ajoutant l'item sélectionné et on ajoute à la fin une virgule avec un espace pour plus de clarté.

Pour la JList, je créer une méthode « initjLPlantes » qui va créer le modèle de la liste, définir le model en sélection simple puis associer ce même modèle à la liste. Ensuite, on remplit la liste avec les différentes plantes.
Afin d'afficher le choix réaliser par l'utilisateur, on associe le composant à un événement (MoussePressed cette fois-ci), puis on modifie la zone d'édition en y ajoutant l'élément sélectionné dans la liste tôt en ajoutant encore une fois une virgule et un espace.

Dernière étape : le constructeur.

Le constructeur contient uniquement 3 lignes*. Il y a **initComponents();** qui va construire l'interface lorsque l'on va lancer le programme. Ensuite, il y a **initjCBoxAnimaux();** qui va initialiser la JComboBox. Puis il y a **initjPlantes();** qui va initialiser la JList.

*Après avoir fini le programme, j'observe que la zone d'édition se remplit dès la construction de l'interface avec le choix d'index 0 de la JComboBox. Pour éviter cela, je vide la zone d'édition après avoir appelé les différents méthodes **init**.

Partie 2 : Application « LesCouleurs »

1. Fonctionnement d'une JComboBox

Les listes déroulantes JComboBox peuvent être utilisées de deux manières différentes. On peut, soit utiliser le modèle prédéfini, soit définir nous-même un modèle. En TP et TD, nous utilisons en général le modèle prédéfini.
Le modèle prédéfini s'utilise très simplement, puisque l'on va uniquement manipuler différentes méthodes que nous propose la JComboBox. Pour ce faire il y a 5 méthodes principales que nous avons vu en cours, pour :
- créer la liste, nous utiliserons : **JComboBox Liste = new JComboBox();**
- ajouter un élément dans cette liste : **Liste.addItem(Item);**
- récupérer l'indice de l'élément sélectionné : **int i = Liste.getSelectedIndex();**

(*l'indice commence à 0)*
- récupérer la valeur de l'indice sélectionné, on y ajoute un « toString(); » :

  String s = Liste.getSelectedItem().toString();
- récupérer le nombre d'élément total de la liste : **int n = Liste.getItemCount();**

D'autre méthode non vu en cours existe, tel que :
- **removeItem(Item);** qui permet de retirer un élément en particulier de la liste.
- **removeAllItem();** qui permet de supprimer tous les éléments de la liste.

2. Fonctionnement d'une JList

Contrairement à la JComboBox, la JList à besoin qu'on lui créer un modèle pour pouvoir l'utiliser, et ensuite de travailler uniquement avec ce modèle. Cependant, il existe un modèle prédéfini qu'on utilisera et qu'on pourra modifier selon nos besoins. Le modèle par défaut s'appelle : « **DefaultListModel** ». Voici les différentes méthodes pour pouvoir travailler avec les JList :

- Création de la JList : **JList Liste = new JList();**
- Création du modèle : **DefaultListModel mod = new DefaultListModel();**
- Association du modèle à la liste : **Liste.setModel(mod);**
- Ajout d'un élément à la liste via le modèle : **mod.addElement(String);**
- Suppression de tous les éléments de la liste : **mod.removeAllElements();**

Les JList peuvent permettre à l'utilisateur de faire des choix multiples. Et, elle nous permet aussi de restreindre ces choix à une mono sélection ou une multi sélection. Ce paramètre s'applique dans les propriétés de la JList au nom de **selectionMode**, où nous avons le choix entre MULTIPLE_INTERVAL, SINGLE ou SINGLE_INTERVAL.

Dans le cas d'une sélection simple, on récupère l'indice de l'élément sélectionné :
Int indice = Liste.getSelectedIndex();

Dans le cas d'une sélection multiple :
Int tab[ ] = List.getSelectedIndices();
Le tableau contiendra les indices de tous les éléments sélectionnés.

3. Voir programme « LesCouleurs »

Le programme est pratiquement complet, à l'exception de quelques lignes où des erreurs d'inattention dans la recopie du tableau on fait qu'il manque des éléments nécessaires à son fonctionnement. J'ai essayé de combler ces erreurs en suivant la logique du programme.
Cependant, je ne suis pas en mesure d'essayer le programme puisque je rencontre une erreur que je ne saurais résoudre. Et par conséquent, sans pouvoir le tester, je ne sais pas si les erreurs que j'ai comblées sont résolues ou non.