TP Les bases de la programmation graphique

Exercice 1 :

Créer une classe permettant de gérer une fenêtre nommée TP SWING dont la taille est 400 ;200.

Afficher cette fenêtre.

Sans créer de classe supplémentaire, afficher un message en console lorsque l'utilisateur clique sur la fenêtre. Ce message indique le message clic ainsi que les coordonnées de l'endroit où l'utilisateur a cliqué.

Refaire la même chose en ajoutant une classe EcouteurSouris.

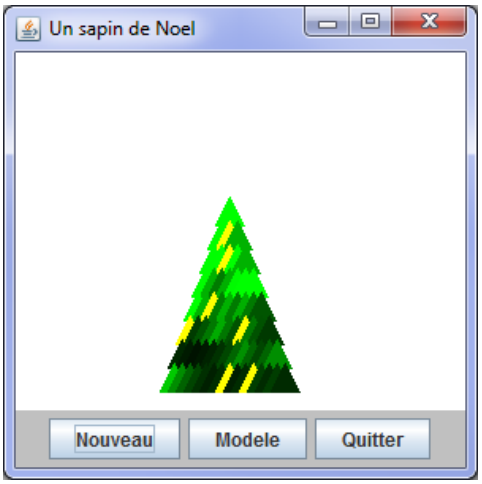
Ajoutez un bouton « Bouton1 » à votre fenêtre et une classe EcouteurBouton. Lorsque l'utilisateur clique sur le bouton un message affichant « clic sur bouton 1 » apparaît.

Ajoutez un bouton « Bouton2 ». Lorsque l'utilisateur clique sur ce bouton un message affichant « clic sur bouton 2 » apparaît. Ne pas ajouter une classe supplémentaire pour gérer cela.

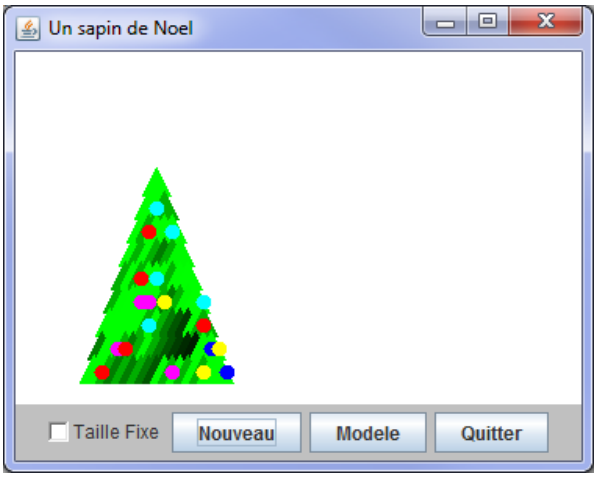
Exercice 2 :

L'objectif du TP est d'améliorer le programme fourni :

- *Si la case Taille fixe est cochée, les nouveaux sapins de taille constante sont dessinés avec une guirlande formée de cercles de différentes couleurs.*
- *Si la case n'est pas cochée, les nouveaux sapins sont également dessinés mais avec une taille variable.*



Avant modifications



Après modifications

Première étape : Comprendre les techniques d'affichage graphique

Pour afficher un sapin de taille différente chaque fois que l'utilisateur clique sur le bouton `Nouveau` :

a. Recherchez dans l'ensemble des classes de l'application `Fenetre`, la méthode associée au clic sur le bouton `Nouveau`.

b. Modifiez cette méthode de façon à ce que l'arbre se construise avec une taille aléatoire, variant entre `3` et `10`.

c. Une fois ces modifications réalisées, compilez l'application, et vérifiez le bon fonctionnement du bouton `Nouveau`.

Pour dessiner une guirlande de cercles de couleurs différentes :

a. Avant d'afficher une guirlande, modifiez les classes `Triangle` et `Arbre` de sorte que le design ne soit affiché qu'à l'aide de triangles verts. Vérifiez l'exécution du programme.

b. Pour afficher une guirlande de couleur rouge, créez une classe `Boule` en vous inspirant de la classe triangle.

c. Modifiez ensuite la méthode `dessine()` de la classe `Arbre`, de façon à construire et afficher par-dessus le sapin des objets `Boule` lorsque le tableau `sapin` vaut `1`.
Vérifiez l'affichage des boules.

d. Pour afficher une guirlande de couleurs différentes, définissez dans la classe Boule une tableau de plusieurs couleurs, comme suit :

```
Color[] couleur= {Color.red, Color.blue, Color.yellow, Color.cyan, Color.magenta} ;
```

Le choix de la couleur est ensuite effectué dans le constructeur de la classe boule en tirant au hasard une valeur comprise entre `0` et `4`. Cette valeur est utilisée comme indice de tableau de couleurs pour initialiser la couleur d'affichage (`setColor()`) à la couleur du tableau correspondant à l'indice tiré au hasard.

Remarque :
La méthode `fillOval(x,y,l,h)` permet l'affichage de cercles remplis. Elle s'applique à un objet `Graphics`, comme la méthode `fillPolygon()`. Les paramètres `x` et `y` représentent la position à l'écran du coin supérieur gauche du rectangle englobant le cercle, `l` et `h` représentant la largeur et la hauteur de ce même rectangle.

Seconde étape : Apprendre à gérer les événements

Placer une case à cocher dans la boîte à boutons :

a. Sachant que la classe décrivant les cases à cocher a pour nom Checkbox, ajoutez un objet de ce type dans la boîte à boutons de l'application Fenetre. Le texte(« taille fixe ») suivant la case à cocher est placé en paramètre du constructeur de la classe.

b. L'écouteur d'événement associé aux objets de type `Checkbox` s'appelant `ItemListener`, ajoutez cet écouteur à la case à cocher.

Associer l'événement à l'action.

Lorsque la case est cochée, les nouveaux sapins affichés par le bouton `Nouveau` sont de taille fixe. Inversement, lorsque la case n'est pas cochée, les sapins sont de taille aléatoire. L'état de la case à cocher a donc une influence sur l'affichage du sapin géré par le bouton `Nouveau`. C'est pourquoi il est logique de gérer l'écouteur `ItemListener` dans la même classe `qu'ActionListener`.

a. Sachant que l'interface `ItemListener` ne définit qu'un seul comportement `itemStateChanged()`, modifez l'en-tête de la classe `GestionAction` de façon à ce qu'elle implémente les deux interfaces `ActionListener` et `ItemListener` en séparant les deux termes par une virgule.

b. Analysez la méthode `itemStateChanged()` décrite ci-dessous, et déterminez comment déclarer la variable Ok pour qu'elle puisse être également visible de l'objet BNouveau.

```
public void itemStateChanged(ItemEvent e) {
   if(e.getStateChange() == ItemEvent.SELECTED)  OK = false;
   else OK = true;
 }
```

c. Sachant que les sapins de taille aléatoire sont affichés par l'intermédiaire de la méthode `nouveau()` (classe `Dessin`), modifies la méthode de façon que la taille du sapin soit fixe ou aléatoire, en fonction de la valeur de la variable OK.