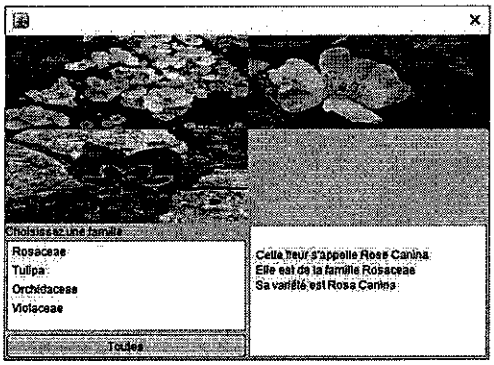
Licence Informatique 1ère année - Module Info2b – Interfaces visuelles

Examen– Durée 2h – Lundi 13 mai 2019
Documents autorisés (polycopiés de CM, TD et TP)

Un fleuriste souhaite proposer à ses clients une application leur permettant de créer leur bouquet en sélectionnant ses fleurs.
L'utilisateur de l'application peut choisir une fleur dans une liste déroulante, sa description textuelle est alors affichée dans la zone d'édition, ainsi que sa photo. Si la fleur lui convient, il peut l'ajouter au bouquet en cliquant sur le bouton « Ajouter ». Le nombre de fleurs du bouquet est affiché en bas de la fenêtre.

- La sous-option « Bouquet » de l'option « Visualiser » du menu, permet l'ouverture d'une boîte de dialogue qui permet de visualiser toutes les fleurs du bouquet sous forme d'une liste des fleurs.
- La sous-option « Galerie » de l'option « Visualiser » du menu permet l'ouverture d'une boîte de dialogue qui permet de visualiser toutes les fleurs du bouquet sous forme d'une galerie de photos ou bien seulement les fleurs du bouquet appartenant à une même famille en sélectionnant le nom de la famille dans une liste.

L'interface de l'application décrite ci-contre comporte un menu, avec une option nommée « Visualiser » et deux sous-options nommées « Bouquet » et « Galerie ».
Pour gérer les données de cette application, deux classes ont été créées.
• La classe « Fleur » modélise une fleur qui est décrite par son nom, sa famille, sa variété et sa photo. La photo d'une fleur est gérée à l'aide d'une image de type « ImageIcon ».
• La classe « LesFleurs » permet de gérer un ensemble de fleurs en utilisant une liste de type « ArrayList<Fleur> ».



La sélection de la sous-option « Galerie » ouvre une boîte de dialogue dont l'interface est décrite ci-contre. Elle comporte une liste de type « JList », nommée « ListeFamilles » contenant tous les noms de famille des fleurs. Une zone d'Edition nommée « Edition » dans laquelle s'affiche les informations d'une fleur lorsque l'on clique sur sa photo.
Un panneau de type « JPanel » nommé « PGalerie » qui comporte les photos de fleurs. Ses photos sont affichées à l'aide de composants de type « PanneauImage ».
Initialement toutes les fleurs du bouquet sont affichées dans la galerie, ce qui est aussi le cas lors du clic sur le bouton « Toutes ».
Lors de la sélection d'un nom de famille de fleurs dans la liste, seules les fleurs du bouquet appartenant à cette famille sont affichées dans la galerie. Il est toujours possible de cliquer sur une photo pour avoir le détail textuel d'une fleur.

L'annexe 1 détaille la classe « Fleur » qui décrit une fleur et l'annexe 2 décrit la classe « LesFleurs » qui permet la gestion de l'ensemble des fleurs.

Exercice 1 (5 pts) - *Classes « Fleur » et « LesFleurs »*
En utilisant le code de la classe « Fleur » fourni en annexe 1,

a. (1 pt) Expliquer comment est initialisée la photo d'une fleur.

b. (1 pt) Indiquer comment il est possible de changer la valeur de la photo d'une fleur, illustrer les explications avec les instructions nécessaires (en supposant que la fleur est nommée « f » et que le fichier image de cette fleur est « fleur.jpg » et qu'il se trouve dans le répertoire « src » du projet de cette application.

En utilisant le code de la classe « LesFleurs » fourni en annexe 2,

c. (1 pt) Donner le code java de la méthode toString() de la classe « LesFleurs » qui permet d'obtenir l'affichage des fleurs, comme décrit sur la fenêtre principale ci-dessus (avec le numéro de la fleur et sa description).

```
public String toString(){ // à compléter }
```

d. (1 pt) Donner le code java de la méthode qui permet de rechercher la fleur dont le nom est donné en paramètre. La valeur null est retournée si la fleur n'existe pas.

```
public Fleur rechFleurNom(String n){ // à compléter }
```

e. (1 pt) Donner le code de la méthode « initLesFleurs() » qui permet l'ajout dans la liste des fleurs, d'une fleur nommée "Tulipe Agenensis", de la famille "Tulipa", de la variété "Agenensis" et avec l'image « tulipaage.jpg » qui se trouve dans le répertoire src du projet (sous Netbeans).

```
public void initLesFleurs( ){ // à compléter }
```

Exercice 2 : (4 pts) *Classe « PanneauImage »*
En utilisant le code de la classe « PanneauImage » fourni en annexe 3 et qui a été étudiée en cours.

a. (0,5 pt) Indiquer en une phrase le rôle de cette classe.

b. (1 pt) Expliquer précisément le rôle de la méthode nommée « paint » et commenter ligne à ligne son code
c. (1 pt) Expliquer pourquoi il est nécessaire de surcharger cette méthode « paint »
d. (1 pt) Commenter le code de l'accesseur « setImage » en expliquant précisément ce que fait la méthode « repaint() » qui y est appelée.
e. (0,5 pt) Expliquer pourquoi il est intéressant de construire la galerie d'images de l'application pour le fleuriste avec des composants de type « PanneauImage » plutôt qu'avec des composants de type « JPanel ».

Exercice 3 : ***(11 pts) – Boîte de dialogue (« GalerieDlg »)***
L'annexe 5 décrit partiellement la classe « GalerieDlg ».
La boîte de dialogue « GalerieDlg » permet de visualiser, initialement ou lors du clic sur le bouton « Toutes », toutes les fleurs du bouquet sous forme d'une galerie de leurs photos ou uniquement les fleurs du bouquet appartenant à une famille, lors de la sélection d'un nom de famille dans la liste déroulante nommée « ListeFamille » et de type « JList ».

a. (1,5 pts) **En utilisant l'annexe 4 contenant la méthode « initComponents() »** qui construit l'interface de cette boîte de dialogue, représenter la, sous forme d'une arborescence, avec les types de composant, leur nom et si besoin leur valeur.

b. (1,5 pts) Expliquer les attributs de cette classe et son constructeur (en commentant ligne par ligne son code et le rôle de ses paramètres).

c. (1,5 pts) Donner en le commentant le code de la méthode « initListeFamilles() » qui remplit la liste nommée « ListeFamilles » de type « JList » avec les différents noms de famille des fleurs contenus dans le tableau « tabFam ».

La galerie des photos des fleurs est construite dynamiquement par la méthode « remplirGalerie() » qui est appelée dans le constructeur de la classe et lors de la sélection d'un nom de famille dans la liste. Cette méthode crée une galerie de photos pour la liste de fleurs contenue dans l'attribut « lesF », de type « LesFleurs. La valeur de cet attribut est fixée à la valeur souhaitée avant l'appel de la méthode remplirGalerie() ».

Elle réalise le traitement suivant :

- Supprime tous les composants du panneau « PGalerie » (en utilisant la méthode removeAll())
- Fixe la disposition du panneau « PGalerie » en GridLayout de taille n lignes et n+1 colonnes où n est la racine carrée du nombre de fleurs du bouquet (transtypée en entier).
- Réalise pour chaque fleur de la liste les actions suivantes
  - Récupère cette fleur d'indice i,
  - Crée un objet (instance) de type « PanneauImage » nommé « pani »
  - Fixe son image à la photo de la fleur
  - Fixe sa propriété Name, à la valeur de l'indice i de la fleur
  - Fixe sa taille préférée (setPreferredSize), à une dimension 120 x120 (new Dimension(120,120))
  - Abonne le panneau à un écouteur de type « MouseListener » avec le code ci-dessous

```
pani.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
          afficheInfosFleur(evt);
        }
    });
```

  - Ajoute le panneau, dans le panneau de la galerie.

d. (2,5 pts) Donner le code de cette méthode

```
private void remplirGalerie()  { // à compléter }
```

Le clic sur une photo de la galerie, déclenche l'exécution du gestionnaire d'événement « afficheInfosFleur » qui :
- Récupère le composant de type « PanneauImage » qui a déclenché l'action (en utilisant la méthode « getSource() », appliquée sur l'événement passé en paramètre),
- Récupère la propriété Name du bouton, qui correspond à l'indice du bouton, donc aussi à celui de la fleur dans le bouquet,
- Récupère la fleur de cet indice dans la liste contenue dans l'objet « this.lesF » de type « LesFleurs »
- Affiche dans la zone nommée « Edition » de type « JTextArea » le nom de la fleur, sa famille, et sa variété (comme indiqué sur l'image écran donnée ci-dessus).

e. (2,5 pts) Donner le code de ce gestionnaire.

```
private void afficheInfosFleur(java.awt.event.MouseEvent evt)  { // à compléter }
```

La sélection du nom d'une famille de fleur dans la liste nommée « ListeFamilles » de type JList, effectue les traitements suivants
- récupère le nom de la famille sélectionnée
- recherche dans l'attribut « this.lesF », les fleurs du bouquet appartenant à cette famille
- appelle la méthode pour remplir la galerie
- vide la zone d'édition.

f. (1,5 pts) Donner le code du gestionnaire de sélection dans la liste.

```
private void ListeFleursValueChanged(javax.swing.event.ListSelectionEvent evt) { // à compléter }
```

ANNEXE 1 : Extrait de la classe « Fleur »

```
public class Fleur {
   private String nom;
   private String famille;
   private String variete;
   private ImageIcon photo;

   public String getNom() { return this.nom; }
   public String getFamille() { return this.famille; }
   public String getVariete() { return this.variete; }
   public ImageIcon getPhoto() { return this.photo;}

   public void setNom(String n) { this.nom=n; }
   public void setFamille(String f) { this.famille = f; }
   public void setPhoto(ImageIcon p) { this.photo = p; }
   public void setVariete(String v) { this.variete = v; }

  public Fleur() {
      this.nom = ""; this.famille=""; this.variete = "";
      this.photo = new ImageIcon(getClass().getResource("/fleurDefaut.png")); }

  public Fleur(String n, String f, String v) {
      this.nom= n; this.famille=f;this.variete = v;
      this.photo = new ImageIcon(getClass().getResource("/fleurDefaut.png")); }

   public String toString()
         { String res= "\nNom " + this.nom;
              res+= "\nFamille " + this.famille;
              res+= "\nVariete " + this.variete;
              return res; }
}
```

ANNEXE 2 : Extrait de la classe « LesFleurs »

```
public class LesFleurs {
   private ArrayList <Fleur> lst;

   public LesFleurs()
    { lst = new ArrayList<Fleur>(); }

   public int getTaille() { return lst.size(); }
   public Fleur getFleur(int i) { return lst.get(i);}

   public void ajouteFleur(Fleur f)
    { lst.add(f); }

   public LesFleurs rechFleursFamille(String f)
   {
      LesFleurs ll= new LesFleurs();
      for (int i=0; i<lst.size(); i++)
         if (lst.get(i).getFamille().equals(f))
            ll.ajouteFleur(lst.get(i));
      return ll;
   }
    public Fleur rechFleurNom(String n)
    { // à compléter }

   public void initLesFleurs()
    { // à compléter }

    public String toString() { // à compléter }
}
```

ANNEXE 3 : Classe « PanneauImage »

```
public class PanneauImage extends JPanel
{ private Image img;

   public PanneauImage()
   { super();
     this.img=null;
   }

   public PanneauImage(Image im)
   {  super();
     this.img=im;
   }

   public Image getImage () { return this.img;}
   public void setImage (Image im) { this.img=im;this.repaint();
   }

   @Override
   public void paint(Graphics g) {
     super.paint(g);
     if (img != null)
     { g.drawImage(img,0,0, this.getWidth(), this.getHeight(),this); }
   }
}
```

ANNEXE 4 : Méthode « initComponents() » de la classe « GalerieDlg » - simplifiée

```
private void initComponents() {
PGalerie = new javax.swing.JPanel();
      PanBas = new javax.swing.JPanel();
      jPanel1 = new javax.swing.JPanel();
      LMessage = new javax.swing.JLabel();
      jScrollPane1 = new javax.swing.JScrollPane();
      ListeFamilles = new javax.swing.JList<>();
      Btoutes = new javax.swing.JButton();
      jScrollPane2 = new javax.swing.JScrollPane();
      Edition = new javax.swing.JTextArea();

      PGalerie.setLayout(new java.awt.GridLayout(2, 2));
      getContentPane().add(PGalerie, java.awt.BorderLayout.CENTER);

      PanBas.setLayout(new java.awt.GridLayout(1, 2));

      jPanel1.setLayout(new java.awt.BorderLayout());

      LMessage.setText("Choisissez une famille ");
      jPanel1.add(LMessage, java.awt.BorderLayout.NORTH);
      . . .
      jScrollPane1.setViewportView(ListeFamilles);
      jPanel1.add(jScrollPane1, java.awt.BorderLayout.CENTER);
      Btoutes.setText("Toutes");
      ...
      jPanel1.add(Btoutes, java.awt.BorderLayout.SOUTH);
      PanBas.add(jPanel1);
      jScrollPane2.setViewportView(Edition);
      PanBas.add(jScrollPane2);
      getContentPane().add(PanBas, java.awt.BorderLayout.SOUTH);
      pack();
}
```

ANNEXE 5 : Extrait de la classe « GalerieDlg»

```
public class GalerieDlg extends JDialog {

    private LesFleurs lbq;
    private String tabFam[ ];


   private LesFleurs lesF; // les fleurs affichées dans la galerie

   public GalerieDlg(java.awt.Frame parent, boolean modal, LesFleurs bq, String  [ ] tf) {
      super(parent, modal);
      initComponents();
      DefaultListModel mod = new DefaultListModel();
      ListeFamilles.setModel(mod);
      this.lbq=bq;
      this.tabFam=tf;
      this.lesF=bq;
      remplirGalerie();
      initListeFamilles()
   }

    private void initComponents() { ...}

    private void initListeFamilles()
   { // à compléter }

   private void remplirGalerie()
    { // à compléter }

   private void afficheInfosFleur(java.awt.event.MouseEvent evt)
     { // à compléter }


   private void ListeFamillesValueChanged(javax.swing.event.ListSelectionEvent evt)
   { // à compléter  }

    private void BtoutesActionPerformed(java.awt.event.ActionEvent evt)
    { // à compléter  }

...
   }
```