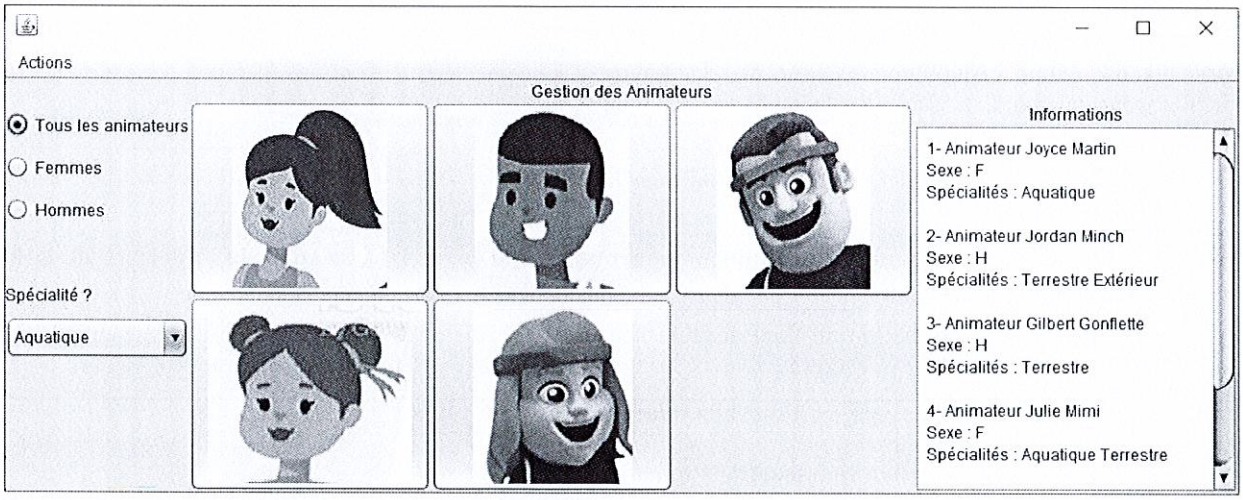
Licence Informatique 1ère année
Module Info2b – Interfaces visuelles

Examen – 2ème session - Durée 2h – Mardi 13 juin 2023
Documents autorisés (polycopiés de CM, TD et TP)

Un étudiant stagiaire a été chargé de développer une application permettant de gérer les animateurs d'une structure associative sportive. Il a proposé un 1er prototype dont l'interface est la suivante :



Elle comporte :
- Un menu avec une option « Actions » et une sous-option « Animateur » pour permettre la saisie d'un nouvel animateur dans une boîte de dialogue
- Une galerie des photos des animateurs (au centre) gérée dynamiquement selon le nombre d'animateurs
- Des boutons radio pour filtrer l'affichage des informations (Tous, homme ou femme)
- Une zone d'édition pour afficher des informations

A l'ouverture de l'application, la zone d'édition est vide, le clic sur un des boutons radio, affiche les informations souhaitées.

Pour gérer les données de cette application, **une classe nommée « Animateur » a été créée**. La classe « Animateur » modélise un animateur avec son nom, son prénom, son sexe (sous forme d'un caractère 'H' ou 'F'), la liste de ses spécialités (par exemple, Aquatique, Terrestre, etc.) et sa photo (sous forme d'un objet de type « ImageIcon »). Un animateur peut avoir plusieurs spécialités. Pour gérer l'ensemble des animateurs, **une classe nommée « LesAnimateurs »** a également été créée.

L'annexe 1 détaille partiellement la classe « Animateur » et l'annexe 2 décrit partiellement la classe « LesAnimateurs ».
En utilisant le code de la classe « Animateur » fourni en annexe 1 :

1. (1,5 pts) Rédiger le code de la méthode « toString() » qui permet de renvoyer sous forme d'une chaîne de caractères les informations concernant un animateur. Attention un animateur peut avoir plusieurs spécialités et les informations concernant sa photo ne sont pas traitées dans cette méthode.

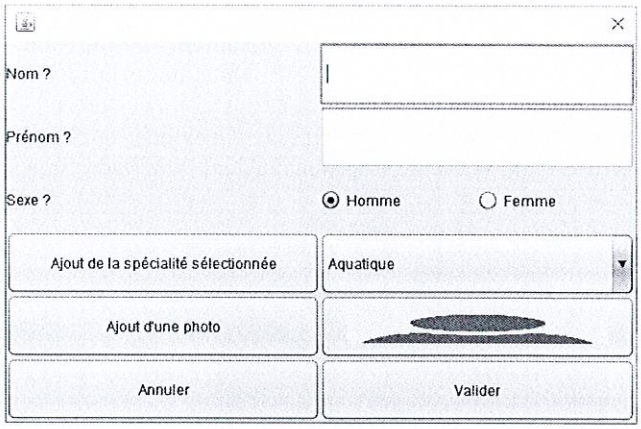
En utilisant le code de la classe « LesAnimateurs » fourni en annexe 2 :

2. (1,5 pts) Rédiger le code de la méthode « initAnimateurs() » pour créer un animateur nommé « Jean MARTIN », de sexe masculin, ayant comme spécialité « Aquatique » et dont la photo est stockée dans le répertoire src du projet (en ressources) dans le fichier nommé « jeanMartin.jpg » et l'ajouter dans la liste des animateurs.

3. (1 pt) Expliquer avec des phrases la méthode « getListeSpecialites() », en détaillant notamment les 3 boucles for de cette méthode.

4. (1 pt) Rédiger le code de la méthode « toString() » qui permet de renvoyer sous forme d'une chaîne de caractères les informations de tous les animateurs sous un format de votre choix.

La sélection de la sous-option « Animateur » de l'option « Actions » du menu, permet l'ouverture **d'une boite dialogue nommée « SaisieAnimateurDlg » de type « JDialog »** dont l'interface est la suivante.



La sélection du sexe est faite à l'aide de boutons radio, l'ajout d'une spécialité se fait en sélectionnant dans la liste déroulante nommée « LstSpe » de type « JComboBox », puis en cliquant sur le bouton nommé « AjoutSpecialite ». Ce processus peut être répété pour ajouter plusieurs spécialités.

La liste déroulante « LstSpe » est remplie avec toutes les spécialités des animateurs existants, à l'ouverture de la boîte de dialogue.

Le clic sur le bouton nommé « AjoutPhoto » permet de sélectionner un fichier pour la photo, à l'aide de la boîte de dialogue « JFileChooser ».

Le clic sur le bouton nommé « BValider » permet de confirmer la création de l'animateur.

L'annexe 3 détaille partiellement la méthode « initComponents() « qui permet la création de l'interface de la boite de dialogue. En utilisant cette annexe 3 :

5. (2.5 pts) Proposer une description de l'interface de cette boite de dialogue nommée « SaisieAnimateurDlg » de type « JDialog » sous la forme d'une arborescence avec les types de composant, leur nom et si besoin leur valeur.

L'annexe 4 détaille partiellement la classe « SaisieAnimateurDlg ».

En utilisant le code cette classe « SaisieAnimateurDlg » fourni en annexe 4 :

6. (1 pt) Expliquer les informations qui doivent être reçues par la boîte de dialogue. Et si c'est le cas, indiquer comment elles sont récupérées par la boîte de dialogue.
7. (1,5 pts) Expliquer les informations qui doivent être renvoyées par la boîte de dialogue. Et si c'est le cas, indiquer comment elles peuvent être récupérées par l'application principale
8. (1 pt) Expliquer les paramètres du constructeur de la classe et les lignes 3,4 et 5 (seulement).
9. (1,5 pts) La méthode « *remplirListe(ArrayList<String> l)*» permet de remplir la liste déroulante de type « JComboBox » nommée « LstSpe » avec les spécialités possibles des animateurs. Décrire le code de cette méthode.
10. (2 pts) Décrire le code du gestionnaire « *private void AjoutSpecialiteActionPerformed(java.awt.event.ActionEvent evt)* » du clic sur le bouton nommé « AjoutSpecialite ». Il permet de récupérer la spécialité sélectionnée dans la liste déroulante « LstSpe » et de l'ajouter aux spécialités de l'animateur en cours de création.
11. (2 pts) Décrire le code du gestionnaire « *private void BValiderActionPerformed(java.awt.event.ActionEvent evt)*» du clic sur le bouton nommé « BValider ». Il permet de récupérer et d'affecter à l'animateur en cours de création, son nom et son prénom ainsi que son sexe selon le bouton radio coché. Il permet également de gérer la valeur du booléen ok, et la fermeture de la boîte de dialogue.
12. (1 pt) Le gestionnaire « *private void AjoutPhotoActionPerformed(java.awt.event.ActionEvent evt)* » du clic sur le bouton « AjoutPhoto » permet la sélection d'une image et son traitement. Expliquer les actions faites dans cette méthode pour gérer et traiter la photo de l'animateur.

L'annexe 5 détaille partiellement la classe « Examen2S2023 » qui dérive de la classe « JFrame ».

13. (2.5 pts) Décrire le code java du gestionnaire « private void AnimateurActionPerformed(java.awt.event.ActionEvent evt) » du clic sur la sous-option « Animateur » de l'option « Actions » du menu, en le commentant précisément. Ce gestionnaire doit permettre :
    - La création de la boîte de dialogue. Expliquer précisément les valeurs des paramètres.
    - L'ouverture de la boîte. Expliquer le rôle de la méthode utilisée.
    - Si la boîte de dialogue a été fermée par « Valider ». Expliquer comment ce test est fait.
        - Ajout du nouvel dans la liste. Expliquer les attributs, variables ou méthodes utilisées.
        - Réinitialisation du panneau de photo nommé « PanPhotos de type « JPanel ». Indiquer brièvement le rôle de la méthode utilisée.
        - Mise à jour de l'affichage dans la zone de texte nommée « Edition » de type « JTextArea » avec les informations de tous les animateurs y compris le nouveau.

ANNEXE 1 : Classe « Animateur »

```
public class Animateur
{  private String nom;
   private String prenom;
   private char sexe;
   private ArrayList<String> specialites;
   private ImageIcon photo;

   public Animateur() {
      this.nom = ""; this.prenom =""; this.sexe = 'H';
      this.specialites = new ArrayList<String>();
      this.photo = new ImageIcon(getClass().getResource("/animateurDefaut.png"));
   }
   public Animateur( String n, String p, char s) {
      this.nom = n; this.prenom =p; this.sexe = s;
      this.specialites = new ArrayList<String>();
      this.photo = new ImageIcon(getClass().getResource("/animateurDefaut.png"));
   }

   public ArrayList<String> getSpecialites() {return specialites;}
   public void setNom(String nom) {this.nom = nom;}
   public void setPrenom(String prenom) { this.prenom = prenom;}
   public void setSexe(char sexe) {this.sexe = sexe;}
   public ImageIcon getPhoto() {return this.photo;}
   public void setPhoto(ImageIcon p) {this.photo = p; }


   public void ajouteSpecialite(String spe)
   {this.specialites.add(spe);}

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

ANNEXE 2 : Extrait de la classe « LesAnimateurs »

```
public class LesAnimateurs {
   private ArrayList<Animateur> lstA;

   public LesAnimateurs(){ lstA=new ArrayList<Animateur>(); }

   public int getNbAnimateurs() { return this.lstA.size();}
   public Animateur getAnimateur(int i) {return this.lstA.get(i);}
   public void ajouteAnimateur(Animateur a) {this.lstA.add(a);}

   public void initAnimateurs() { // A compléter }
   public String toString()  { // A compléter }

   public ArrayList<String> getListeSpecialites()
   { // A expliquer
   ArrayList<String> ll= new ArrayList<String> ();
   for (int i=0; i< this.lstA.size(); i++)
   { ArrayList<String> ls = this.lstA.get(i).getSpecialites();
     for ( int j=0; j< ls.size(); j++){
       String cat = ls.get(j);
       boolean trouve=false;
       for( int k=0; k< ll.size(); k++)
          if (ll.get(k).equals(cat)) trouve=true;
       if (! trouve) ll.add(cat);  }
   }
   return ll;
   }
}
```

ANNEXE 3 : Extrait de « initComponents() »

```
LNom = new javax.swing.JLabel();
Nom = new javax.swing.JTextField();
LPrenom = new javax.swing.JLabel();
Prenom = new javax.swing.JTextField();
LSexe = new javax.swing.JLabel();
jPanel2 = new javax.swing.JPanel();
BHomme = new javax.swing.JRadioButton();
BFemme = new javax.swing.JRadioButton();
AjoutSpecialite = new javax.swing.JButton();
LstSpe = new javax.swing.JComboBox<>();
AjoutPhoto = new javax.swing.JButton();
BPhoto = new javax.swing.JButton();
BAnnuler = new javax.swing.JButton();
BValider = new javax.swing.JButton();
```

```
private void initComponents() {
    getContentPane().setLayout(new java.awt.GridLayout(6, 2));

    LNom.setText("Nom ?");
    getContentPane().add(LNom);
    getContentPane().add(Nom);

    LPrenom.setText("Prénom ?");
    getContentPane().add(LPrenom);
    getContentPane().add(Prenom);

    LSexe.setText("Sexe ?");
    getContentPane().add(LSexe);

    jPanel2.setLayout(new java.awt.GridLayout(1, 2));

    BHomme.setSelected(true);
    BHomme.setText("Homme");
    jPanel2.add(BHomme);

    BFemme.setText("Femme");
    jPanel2.add(BFemme);

    getContentPane().add(jPanel2);

    AjoutSpecialite.setText("Ajout de la spécialité sélectionnée");
    AjoutSpecialite.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            AjoutSpecialiteActionPerformed(evt);
        }
    });
    getContentPane().add(AjoutSpecialite);

    getContentPane().add(LstSpe);

    AjoutPhoto.setText("Ajout d'une photo");
    AjoutPhoto.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            AjoutPhotoActionPerformed(evt);
        }
    });
    getContentPane().add(AjoutPhoto);
    getContentPane().add(BPhoto);

    BAnnuler.setText("Annuler");
    getContentPane().add(BAnnuler);

    BValider.setText("Valider");
    BValider.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BValiderActionPerformed(evt);
        }
    });
    getContentPane().add(BValider);
...
}
```

ANNEXE 4 : Extrait de la classe « SaisieAnimateurDlg »

```
public class SaisieAnimateurDlg extends javax.swing.JDialog {
   private Animateur anim;
   private boolean ok;

   public SaisieAnimateurDlg(java.awt.Frame parent, boolean modal, ArrayList<String> ls) {
      super(parent, modal);
      initComponents();
      remplirListe(ls);                    // ligne 3
      this.ok=false;                       // ligne 4
      this.anim= new Animateur();          // ligne 5
      //redimensionnement de la photo de l'animateur à la taille du bouton BPhoto
      ImageIcon ScaleIc = new ImageIcon(this.anim.getPhoto().getImage().getScaledInstance(BPhoto.getWidth(),
BPhoto.getHeight(), Image.SCALE_DEFAULT));
      BPhoto.setIcon(ScaleIc);
   }

   public boolean getOk() {return ok;}
   public Animateur getAnimateur(){ return this.anim;}

   private void remplirListe(ArrayList<String> l) { // A compléter  }
   private void initComponents() { ...}

private void AjoutPhotoActionPerformed(java.awt.event.ActionEvent evt) { // A expliquer
   JFileChooser jf= new JFileChooser();
   if (jf.showOpenDialog(this)== JFileChooser.APPROVE_OPTION)
   { String path = jf.getSelectedFile().getPath(); // chemin complet;
     ImageIcon ic= new ImageIcon(path);
     ImageIcon ScaleIc = new ImageIcon(ic.getImage().getScaledInstance(BPhoto.getWidth(), BPhoto.getHeight(),
Image.SCALE_DEFAULT));
      this.anim.setPhoto(ic);
      BPhoto.setIcon(ScaleIc);
     }
    }

   private void BValiderActionPerformed(java.awt.event.ActionEvent evt) { // A compléter}

   private void AjoutSpecialiteActionPerformed(java.awt.event.ActionEvent evt) { // A completer}
}
```

ANNEXE 5 : Classe « Examen2S2023 »

```
public class Examen2S2023 extends javax.swing.JFrame {

   LesAnimateurs lesAnim;

   public Examen2S2023() {
       initComponents();
      lesAnim = new LesAnimateurs();
      lesAnim.initAnimateurs();
      initPanPhotos();
      remplirListeS();
   }

   private void remplirListeS() { ... } // remplit la JComboBox avec les spécialités

   private void initPanPhotos()
   {  PanPhotos.removeAll();
      int nba = lesAnim.getNbAnimateurs();
      int nbLig;
      if (nba%3 == 0)
            nbLig= nba/3;
       else nbLig=nba/3+1;
      PanPhotos.setLayout(new GridLayout(nbLig, 3));
      for(int i=0; i<lesAnim.getNbAnimateurs(); i++)
      {
         JToggleButton jb = new JToggleButton();
         jb.setIcon(lesAnim.getAnimateur(i).getPhoto());
         jb.setName(""+i);
         jb.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
               jbActionPerformed(evt);     }
            });
         PanPhotos.add(jb);
      }
      this.pack();

   }

private void jbActionPerformed(java.awt.event.ActionEvent evt) { ... }

private void AnimateurActionPerformed(java.awt.event.ActionEvent evt) {// A compléter}

private void initComponents() { ... }

}
```