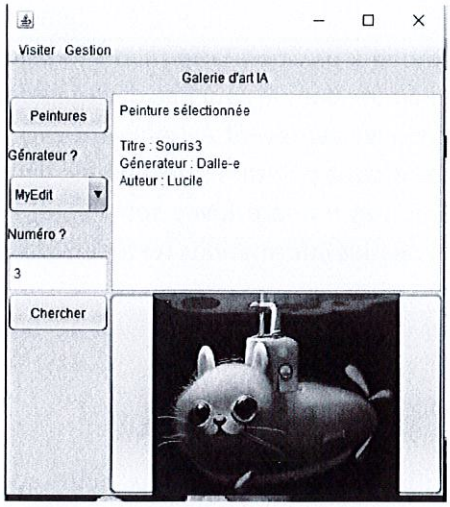
Licence Informatique 1ère année
Module Info2b – Interfaces visuelles

Examen 2025
Vendredi 16 mai 2025 – Durée 1h30
Documents de CM/TD et TP autorisés

Un directeur de galerie d'art a décidé de demander le développement d'une application exposant des peintures artistiques créées par des logiciels d'IA. Un premier prototype avec quelques fonctionnalités a été implémenté et de nouvelles fonctionnalités sont à tester.

L'application principale nommée « GalerieArt » comporte deux classes : une classe « PeintureIA » pour modéliser une peinture générée par un auteur avec une IA et la classe « GalerieArt » qui est l'application principale.

L'interface de l'application principale est la suivante :



Cette interface comporte une barre de menu avec une option « Visiter » et une sous-option « Galerie » qui permet l'ouverture d'une boîte de dialogue nommée « GalerieDlg » qui affiche les peintures de la galerie, des informations sur ces peintures et des statistiques sous forme d'un histogramme.

L'annexe 1 décrit la classe « PeintureIA ».

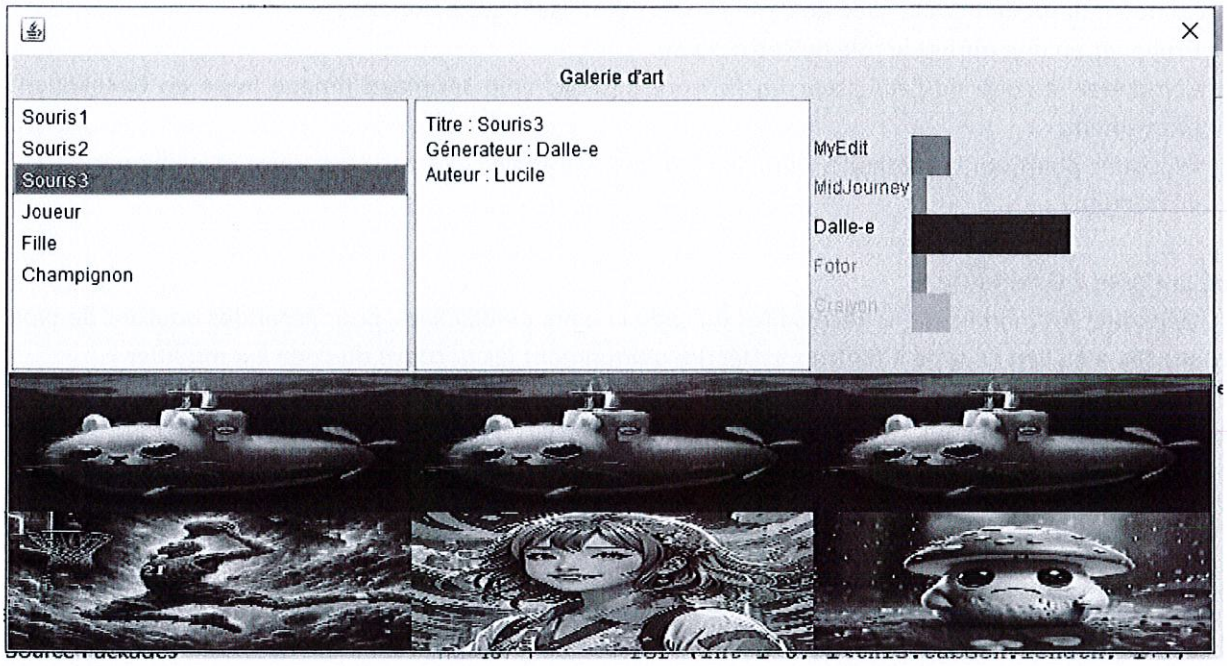
L'annexe 2 décrit partiellement la classe « GalerieArt » qui est l'application principale qui dérive de la classe « JFrame ».

L'annexe 3 décrit partiellement la boîte de dialogue « GalerieDlg » qui dérive de la classe JDialog.

L'annexe 4 décrit la classe « BoutonImage »

Pour gérer l'ensemble des peintures, un attribut « private ArrayList <PeintureIA> lstIA » a été déclaré dans la classe « GalerieArt », ainsi qu'un attribut de type tableau « private final String tabGen[] » pour gérer les noms des IA qui ont permis de générer les peintures.

On souhaite développer une boîte de dialogue permettant de visualiser les peintures et des informations statistiques en utilisant une boîte de dialogue. L'interface de cette boîte de dialogue est la suivante :



Elle comporte une liste de type « JList » permettant de sélectionner une peinture selon son titre. Les informations textuelles de la peinture sont alors affichées dans une zone d'édition de type « JTextArea » au centre en haut. Un histogramme en haut à droite montre la répartition des peintures selon leur générateur IA.

En bas, les peintures sont affichées dans une galerie. Il est possible de cliquer sur la photo d'une peinture pour avoir ses informations textuelles.

Classe « GalerieDlg »

1. (3 pts) En utilisant le code de la méthode « initComponents() » de la classe « GalerieDlg » donné en **annexe 5**, proposer une description de l'interface de cette boîte de dialogue sous la forme d'une arborescence avec les types de composant, leur nom et si besoin leur valeur, en respectant les noms proposés décrits dans la méthode.
2. (1,5 pts) Expliquer les paramètres du constructeur et leur usage.
3. (1,5 pt) Donner le code de la méthode « private void initListePeintures() » qui remplit la « JList » nommée « ListePeinture » avec les titres des peintures.
4. (2,5 pts) Compléter le code de la méthode « private void afficheGalerie() » qui crée dynamiquement pour chaque peinture un bouton de type « JButton », lui affecte un numéro à l'aide de sa propriété « Name », l'abonne à un écouteur de type « ActionListener » pour exécuter la méthode « btActionPerformed » lorsque l'on clique sur ce bouton, affiche la photo de la peinture sur le bouton, et l'ajoute dans le panneau de la galerie.
5. (2 pts) Compléter le code de la méthode « public void btActionPerformed(java.awt.event.ActionEvent evt) » exécutée lors du clic sur un bouton de la galerie qui affiche les informations de la peinture sélectionnée dans la zone d'édition. On pourra 1) récupérer le bouton cliqué 2) récupérer son numéro (avec son Name) 3) récupérer la peinture ayant cet indice dans la liste des peintures et 4) afficher les informations textuelles dans la zone d'édition.
6. (1,5 pts) La méthode « private void dessineHisto() » permet de dessiner un histogramme. Expliquer à l'aide phrases :
    a. À quoi correspond la longueur des barres de l'histogramme (pour chaque générateur).
    b. À quoi correspondent les paramètres des méthodes « fillRect ».
7. (2,5 pts) Expliquer quand et comment la méthode « dessineHisto() » peut être appelée pour que l'histogramme soit affiché à l'ouverture de la boîte de dialogue et reste affiché même si la boîte est redessinée. Le cas échéant, donner le code à ajouter, en l'expliquant.

« Classe BoutonImage »

8. (1,5 pts) On souhaite modifier le code de la classe « GalerieDlg » pour améliorer l'affichage des peintures sur les boutons, en utilisant une classe dédiée nommée « BoutonImage » qui dérive de la classe JButton ». L'annexe 4 décrit partiellement le code de cette classe. En utilisant cette annexe :
    a. Expliquer en une phrase le rôle de cette classe.
    b. Compléter le code de l'accesseur en écriture « public void setImage (Image im) » en l'expliquant brièvement.
    c. Expliquer pourquoi la méthode « public void paint(Graphics g) » est nécessaire et expliquer sa 1ère instruction.

Modification de la classe « GalerieDlg »

9. (1,5 pts) Expliquer les modifications nécessaires du code la boîte de dialogue, pour gérer des boutons de type « BoutonImage » au lieu du type « JButton ». Décrire uniquement les portions de code à « modifier ».

Classe principale « GalerieArt »

10. (1 pt) Indiquer les informations que doit recevoir la boîte de dialogue de la part de l'application principale et sous quelle forme, et le cas échéant les informations renvoyées par la boîte et sous quelle forme.
11. (1,5 pts) Donner le code du gestionnaire d'événement « private void MGalerieActionPerformed(java.awt.event.ActionEvent evt) » qui permet d'afficher la boîte de dialogue lors du clic sur la sous-option « Galerie » de l'option « Visiter » du menu de l'application principale.

ANNEXE 1

Classe « PeintureIA »

```
public class PeintureIA {
   private String titre;
   private String generateur;
   private String auteur;
   private ImageIcon peinture;

   public PeintureIA(String titre, String generateur) {
      this.titre = titre; this.generateur = generateur; this.auteur="Inconnu";
      this.peinture=new ImageIcon(getClass().getResource("/Img/ImageDefaut.jpg"));
   }

   public String getTitre() {return titre;}
   public void setTitre(String titre) {this.titre = titre;}
   public String getGenerateur() { return this.generateur; }
   public void setGenerateur(String generateur) { this.generateur = generateur; }
   public String getAuteur() {return auteur;}
   public void setAuteur(String auteur) {this.auteur = auteur;}
   public ImageIcon getPeinture() {return peinture;}
   public void setPeinture(ImageIcon peinture) {this.peinture = peinture;}

   public String toString() {
      String s=""; s+="Titre : "+this.titre+"\n";
      s+="Générateur : "+this.generateur+"\n";
      s+="Auteur : "+this.auteur+"\n";
      return s;
   }
}
```

ANNEXE 2

Classe « GalerieArt »

```
public class GalerieArt extends javax.swing.JFrame {
   private ArrayList<PeintureIA> lstIA;
   private final String tabGen[]={"MyEdit","MidJourney","Dalle-e","Fotor","Craiyon"};

   public GalerieArt() {
      initComponents();
      this.lstIA=new ArrayList<>();
      creationPeinturesTest();
      initListeGenerateurs();
      afficheInfos(this.lstIA,"Peintures de la galerie");
   }
...
    private void MGalerieActionPerformed(java.awt.event.ActionEvent evt) {
      // A COMPLETER
   }
}
```

ANNEXE 3

Classe «GalerieDlg »

```
public class GalerieDlg extends javax.swing.JDialog {
   private ArrayList<PeintureIA> lstP;
   private String [] tabGen;

   public GalerieDlg(java.awt.Frame parent, boolean modal, ArrayList<PeintureIA> lst, String tab []) {
      super(parent, modal);
      initComponents();
      this.lstP=lst;
      this.tabGen=tab;
      initListePeintures();
      afficheGalerie();
   }

   public int NbPeinturesGen(String gen) {
      int compt=0;
      for (int i=0; i<this.lstP.size(); i++)
        if (gen.equals(this.lstP.get(i).getGenerateur())) compt++;
      return compt;  }

   private void dessineHisto() {
    int largeurPan= this.PDessin.getWidth();
    int hautPan = this.PDessin.getHeight();
    int nbGen = this.tabGen.length;
    int hautBarre= hautPan/(nbGen+2);
    int largBarreMax=largeurPan/2;
    Graphics g=PDessin.getGraphics();
    for (int i=0; i<this.tabGen.length; i++) {
      int nbG=NbPeinturesGen(this.tabGen[i]);
      int largB;
      if (nbG==0) largB=10;
      else largB=(largBarreMax*nbG)/nbGen;
      int r=(int) (Math.random()*256);
      int v=(int) (Math.random()*256);
      int b=(int)(Math.random()*256);
      g.setColor(new Color(r,v,b));
      g.fillRect(largBarreMax/2,(i+1)*hautBarre, largB, hautBarre);
      g.drawString(this.tabGen[i], 0, (i+1)*hautBarre+hautBarre/2);
    }}

   private void initListePeintures()   { // A COMPLETER  }

   private void afficheGalerie() {
     int nbp= this.lstP.size();
     int n= (int) (Math.sqrt(nbp));
     this.PGalerie.setLayout(new GridLayout( n, n+1));
    // A COMPLETER
   }

  public void btActionPerformed(java.awt.event.ActionEvent evt) {//A COMPLETER }
}
```

ANNEXE 4

Classe « BoutonImage »

```
public class BoutonImage extends JButton {
   private Image img;
   public BoutonImage() { super(); this.img=null;}
   public BoutonImage(Image im) {   super();  this.img=im;}
   public Image getImage () { return this.img;}
   public void setImage (Image im) { // A COMPLETER  }
   public  void paint(Graphics g) {
      super.paint(g);
      if (this.img != null)
      { Image imgB = this.img.getScaledInstance(this.getWidth(),this.getHeight(), Image.SCALE_DEFAULT);
       this.setIcon(new ImageIcon(imgB)); }
   }
}
```

ANNEXE 5

Méthode « initComponents() » de la boîte de dialogue « galerieDlg »

```
private void initComponents() {
      PHaut = new javax.swing.JPanel();
      LTitre = new javax.swing.JLabel();
      PCentre = new javax.swing.JPanel();
      PCentreHaut = new javax.swing.JPanel();
      jScrollPane1 = new javax.swing.JScrollPane();
      ListePeintures = new javax.swing.JList<>();
      jScrollPane2 = new javax.swing.JScrollPane();
      Edition = new javax.swing.JTextArea();
      PDessin = new javax.swing.JPanel();
      PGalerie = new javax.swing.JPanel();
      LTitre.setText("Galerie d'art");
      PHaut.add(LTitre);
      getContentPane().add(PHaut, java.awt.BorderLayout.NORTH);
      PCentre.setLayout(new java.awt.GridLayout(2, 1));
      PCentreHaut.setLayout(new java.awt.GridLayout(1, 3));
      ListePeintures.addMouseListener(new java.awt.event.MouseAdapter() {
         public void mouseClicked(java.awt.event.MouseEvent evt) {
            ListePeinturesMouseClicked(evt);
         } });
      jScrollPane1.setViewportView(ListePeintures);
      PCentreHaut.add(jScrollPane1);
      Edition.setColumns(20);
      Edition.setLineWrap(true);
      Edition.setRows(5);
      jScrollPane2.setViewportView(Edition);
      PCentreHaut.add(jScrollPane2);
      PDessin.setLayout(new java.awt.GridLayout(1, 1));
      PCentreHaut.add(PDessin);
      PCentre.add(PCentreHaut);
      PGalerie.setLayout(new java.awt.GridLayout(1, 4));
      PCentre.add(PGalerie);
      getContentPane().add(PCentre, java.awt.BorderLayout.CENTER);
}
```