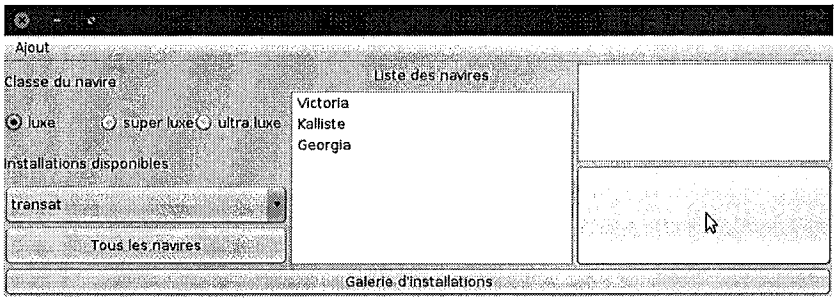
Licence Informatique 1ère année
Module Info2b – Interfaces visuelles

Examen – Durée 2h - Mardi 11 mai 2021
Documents autorisés (polycopiés de CM, TD et TP)

Une compagnie maritime possède différents types de navires.
Un **navire** (décrit par la classe « Navire ») est caractérisé par un nom, une classe (luxe, super luxe, ultra luxe), une photo (de type « ImageIcon »), et un ensemble d'installations proposées à bord sous forme d'une liste de type « ArrayList<Installation> ».
Une **installation** (décrite par la classe « Installation ») comporte une catégorie (transat, jaccuzzi, fauteuil, etc.), un prix et une photo (de type « Image »).
L'ensemble des navires est géré dans une classe nommée « LesNavires » qui comporte une liste de navires (de type « ArrayList<Navire> »).

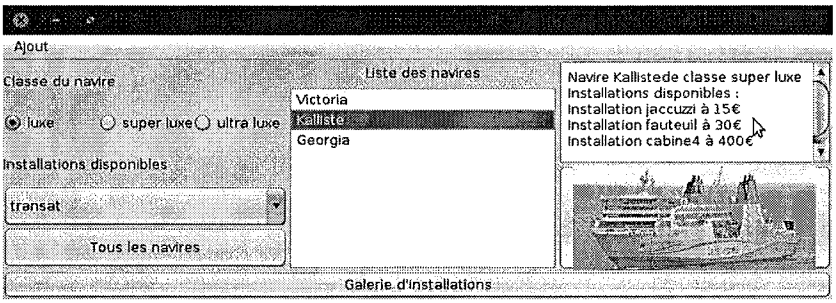
La compagnie a décidé de présenter au public l'ensemble de ses navires à l'aide d'une application, dont un 1er prototype a été développé. L'interface de l'application principale « Examen2021 » est de la forme suivante :



Cette interface comporte

- une liste (nommée « Navires » de type « JList ») affichant les noms des différents navires de la compagnie (au centre). La liste les navires (de type « JList ») est en mode mono-sélection.
- une liste déroulante (nommée « Installations » de type « JComboBox ») permettant de sélectionner une installation présente sur au moins un des navires.
- trois boutons radio permettant de sélectionner une classe de navire (luxe, super luxe ou ultra luxe)
- un bouton permettant d'afficher tous les navires dans la liste centrale (affichage par défaut au démarrage)
- un bouton de type « JButton » nommé « Photo » en bas à droite
- une zone d'édition de type « JTextArea » nommée « Infos » en haut à droite
- un bouton "Galerie d'installations" qui permet d'ouvrir une boite de dialogue pour visualiser une galerie des photos des installations du navire sélectionné.
- un menu avec une option « Ajout » et une sous-option « Installation » qui permet d'ajouter une installation au bateau sélectionné via une boite de dialogue.

La sélection d'une installation dans liste déroulante permet d'afficher dans la liste centrale uniquement le nom des navires comportant cette installation. De même, la sélection d'une classe à l'aide des boutons radio permet d'afficher dans la liste centrale uniquement le nom des navires de cette classe.
La sélection d'un navire dans la liste centrale affiche sa photo sur le bouton de droite et les informations le concernant dans la zone d'édition à droite comme le montre l'interface ci-dessous.



L'annexe 1 présente un extrait des classes « Navire », « Installation » et « LesNavires ».

Exercice 1 (4 pts)
En utilisant les classes fournies en annexe 1,

1. (1 pt) Rédigez le code java de la méthode « **public String toString()** » de la classe « Navire » qui permet d'obtenir l'affichage des informations du navire visibles sur la capture d'écran, c'est-à-dire les informations sur le navire et sur ses installations.
2. (1 pt) Expliquez ligne par ligne le constructeur de la classe « Installation » (en gras).
3. (1,5 pts) Expliquez avec quelques phrases synthétiques l'attribut de la classe « LesNavires » et l'extrait de la méthode « public initNavires() ».
4. (0,5 pt) Expliquez précisément en une phrase le rôle de la méthode « **public LesNavires getNaviresInst(String cat)** » de la classe « LesNavires».

Exercice 2 (5 pts) - *Application principale « **Examen2021** »*
L'application principale (nommée « Examen2021 ») qui hérite de la classe « JFrame » comporte l'attribut et le constructeur suivants :

```
private LesNavires lesN;

public Examen2021() {
    initComponents();
    this.lesN = new LesNavires();
    this.lesN.initNavires();
    DefaultListModel mod = new DefaultListModel();
    Navires.setModel(mod);
    initListeInst();  // remplit la liste des catégories d'installations (JComboBox)
    afficheTous();  // remplit la liste des navires (JList)
  }
```

1. (1,5 pts) Rédigez le code de la méthode « **private void initListeInst()** » qui remplit la liste des catégories d'installations (nommée « Installations » de type « JComboBox ») avec les installations disponibles sur les navires de la compagnie (sans doublon).

Lors de la sélection dans la liste déroulante d'une catégorie d'installation, tous les noms des navires qui possèdent cette catégorie d'installation sont affichés dans la liste « Navires ».

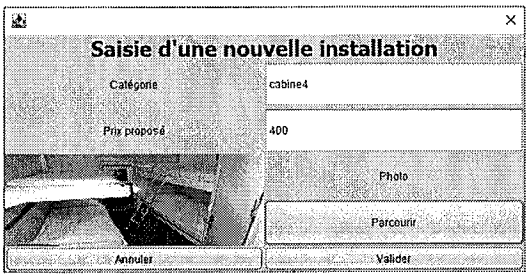
2. (2 pts) Rédigez le code du gestionnaire de la sélection dans la liste déroulante « Installations ».

   « **private void InstallationsActionPerformed(java.awt.event.ActionEvent evt)** ».

   Ce gestionnaire doit récupérer la catégorie sélectionnée, récupérer tous les navires qui ont une installation de cette catégorie, vider la liste, puis y ajouter les noms des navires qui ont l'installation qui a été sélectionnée. On rappelle que la méthode « RemoveAllElements() » de la classe « DefaultListModel » permet de supprimer tous les éléments d'une JList.

3. (1,5 pts) La sélection d'un navire dans la liste centrale affiche sa photo sur un bouton (nommé « Photo » de type « JButton ») et les informations le concernant dans une zone d'édition (nommée « Infos » de type « JTextArea »).
   Rédigez le code du gestionnaire « **private void NaviresValueChanged(javax.swing.event.ListSelectionEvent evt)** ».

Exercice 3 (5,5 pts) - *Boîte de dialogue « **AjoutInstallationDlg** »*
Le clic sur la sous-option « Installation » de l'option « Ajout » du menu de l'application principale ouvre une boîte de dialogue « AjoutInstallationDlg ». Cette dernière permet à l'utilisateur de réaliser la saisie d'une nouvelle installation. L'interface de cette boîte de dialogue est visible ci-dessous.



L'annexe 3 décrit la méthode « initComponents() » de la classe « AjoutInstallationDlg » qui dérive de la classe « JDialog ».

1. (2 pts) A l'aide de l'annexe 3, représentez cette interface sous forme d'une arborescence avec les types des composants, leur nom et si besoin leur valeur.
2. (1,5 pts) Cette classe comporte un attribut « **private Image img** » qui contient la valeur de l'image affichée lorsqu'elle a été sélectionnée avec le bouton « Parcourir ».

   Expliquez brièvement le code du gestionnaire du clic sur le bouton « Parcourir » et de la méthode « paint() donnés ci-dessous, en expliquant particulièrement comment est réalisé l'affichage de l'image et pourquoi la méthode « paint » est nécessaire.

```
private void ParcourirActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser jf=new JFileChooser();
    if (jf.showOpenDialog(jf)==JFileChooser.APPROVE_OPTION)
    img=Toolkit.getDefaultToolkit().getImage(jf.getSelectedFile().getPath());
    repaint();
}
public void paint(Graphics g)
{
   super.paint(g);
   if (img!=null)
   { Graphics gg= Photo.getGraphics();
    gg.drawImage(img, 0, 0, Photo.getWidth(), Photo.getHeight(), this);
   }
}
```

3. (2 pts) La classe « AjoutIntallationDlg » comporte des accesseurs : **« public boolean isOk() { return this.ok; } »** et **« public Installation getInst() { return this.inst; } »** pour savoir si la boite de dialogue a été fermée par le bouton « Valider » (avec l'attribut « ok » valant « true ») et obtenir l'installation qui a été saisie.
   Donner le code du gestionnaire d'évènement de la sous-option « Installation » de l'option « Ajout » du menu de l'interface principale
   « private void AjoutInstallationActionPerformed(java.awt.event.ActionEvent evt) » qui :
   - vérifie qu'un navire a été sélectionné dans la liste « Navires » (en vérifiant que l'index sélectionné est différent de -1)
   - récupère le nom du navire sélectionné et le navire correspondant
   - crée une instance de la boîte de dialogue « AjoutInstallationDlg » et l'affiche
   - ajoute l'installation créée au navire sélectionné si la boîte de dialogue a été fermée avec le bouton « Valider ».

Exercice 4 (5,5 pts) - *Boîte de dialogue* ***« GalerieDlg »***

Lorsqu'un nom de navire est sélectionné dans la liste (JList au centre) de la fenêtre principale, le clic sur le bouton « Galerie des installations » de la fenêtre principale ouvre une boîte de dialogue « GalerieDlg » qui permet de visualiser les photos des installations disponibles sur ce navire.

L'interface de cette boîte de dialogue est visible ci-contre.

Elle est composée d'un label nommé « Titre », d'un panneau nommé « Centre » rempli dynamiquement avec les photos des installations disponibles sur le navire, et d'un bouton « Fermer » permettant de fermer la fenêtre et de revenir à l'application principale.

1. (1 pt) Décrire le ou les attributs nécessaires pour cette classe « GalerieDlg » en expliquant votre choix.

2. (2 pts) Donner le code du constructeur de la classe. Expliquer en particulier comment il permet de récupérer le navire dont les installations doivent être affichées dans la galerie. Il est à noter que le nom du navire apparaît dans le label du titre (comme sur l'exemple présenté, où le navire s'appelle « Kalliste ».

   Ce constructeur devra appeler une méthode pour créer et remplir la galerie (nommée « remplirGalerie() »).

3. (2,5 pts) Donner le code java de cette méthode **« private void remplirGalerie() »**.

   Elle réalise les traitements suivants :

   - elle récupère la liste des installations du navire
   - elle fixe la disposition du panneau « Centre » en GridLayout de taille n lignes et n+1 colonnes où n est la racine carrée du nombre d'installations à afficher (arrondie à l'entier supérieur). On rappelle que la fonction donnant la racine carrée de n est Math.sqrt(n).
   - elle réalise pour chaque installation de la liste les actions suivantes :
     a) récupération de l'installation d'indice i,
     b) création d'une instance d'un composant de type « BoutonImage »
     c) affectation de la photo de cette installation pour ce « BoutonImage »
     d) ajout du « BoutonImage » dans le panneau « Centre ».

L'annexe 2 fournit le contenu de la classe « BoutonImage ».

ANNEXE 1: extrait des classes «Navire», «Installation» et «LesNavires»

```
public class Navire {
   private String nom;
   private String classe;
   private ArrayList<Installation> installations;
   private ImageIcon photo;

   public String getNom() { return nom;}
   public ImageIcon getPhoto() {return photo; }
   public ArrayList<Installation> getInstallations() {return installations;}

   public Navire(String nom, String classe, ArrayList<Installation> installations, String photo) {
      this.nom = nom;
      this.classe = classe;
      this.installations = installations;
      this.photo = new ImageIcon(getClass().getResource(photo));
   }

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

```
public class Installation {
   private String categorie;
   private int prix;
   private Image photo;

   public Installation(String cat, int c, String img) {  //à expliquer
      this.categorie = cat;
      this.prix = c;
      this.photo = Toolkit.getDefaultToolkit().getImage(getClass().getResource(img));
   }

   public Installation(String cat, int c) {
      this.categorie = cat;
      this.prix = c;
   }

   public String getCategorie() {return this.categorie;}
   public void setCategorie(String categorie) {this.categorie = categorie;}
   public void setPrix(int prix) {this.prix = prix; }
   public void setPhoto(Image photo) {this.photo = photo;}
   public int getPrix() {return this.prix;}
   public Image getPhoto() {return this.photo;}

   public String toString() { return "Installation " + this.categorie + " à " + prix + "€";}
}
```

```
public class LesNavires {
    private ArrayList<Navire> ln;
    public LesNavires()  { this.ln = new ArrayList<Navire> (); }

   public void initNavires() { //à expliquer
      Installation i1 = new Installation("jaccuzzi", 15, "jaccuzzi.jpg");
      Installation i2 = new Installation("fauteuil", 30, "fauteuil.jpg");
      Installation i3 = new Installation("cabine2", 100, "cabine2-lits-int.jpg");
      ArrayList<Installation> t1 = new ArrayList<Installation>();
      t1.add(i1); t1.add(i2);  t1.add(i3);
      Navire n1 = new Navire("Victoria", "luxe", t1, "victoria.jpg");
      ln.add(n1);
       ...
   }
```

```
public void ajoutNavire(Navire t) { this.ln.add(t); }
  public int getNbNavires() {return this.ln.size();}

  // méthode qui retourne le navire dont l'indice est passé en paramètre (ou null si l'indice est incorrect)
  public Navire getNavire(int ind) { . . . }

  // méthode qui retourne le navire dont le nom est passé en paramètre (ou null si le nom est incorrect)
  public Navire getNavireNom(String n) { . . . }

  // méthode qui retourne  la liste des catégories d'installation sans doublon
  public ArrayList<String> getListeCatInst() { . . . }

  public LesNavires getNaviresInst(String cat) { //à expliquer
    LesNavires lp = new LesNavires();
    for(int i=0; i<this.ln.size(); i++){
      ArrayList<Installation> li = ln.get(i).getInstallations();
      int j = 0;
      boolean trouve = false;
      do{
        if(li.get(j).getCategorie().equals(cat)){
          lp.ajoutNavire(this.ln.get(i));
          trouve = true;
        }
        else
          j++;
      } while (!trouve && j < li.size());
    }
    return lp;
  }

  public String toString() {
    String s="";
    for (int i=0; i<ln.size(); i++)
    {
      s+="\n\nNavire N°"+(i+1);
      s+=ln.get(i).toString();
    }
    return s;
  }
}
```

ANNEXE 2 : Contenu de la classe « BoutonImage » (vue en CM)

```
public class BoutonImage extends JButton{
  private Image img;

  public BoutonImage(Image i)  {  super();  this.img=i;  }
  public BoutonImage() {  this(null);  }

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

  public void paint(Graphics g) {
    super.paint(g);
    if (img != null)
    {      Image imgB = img.getScaledInstance(this.getWidth(),this.getHeight(), Image.SCALE_DEFAULT);
           this.setIcon(new ImageIcon(imgB));
    }
  }
}
```

ANNEXE 3

Extrait de la méthode « initComponents » de la classe « AjoutInstallationDlg »

```
private void initComponents() {
    jLabel1 = new javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jPanel6 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    PrixProp = new javax.swing.JLabel();
    jPanel5 = new javax.swing.JPanel();
    Categorie = new javax.swing.JTextField();
    Prix = new javax.swing.JTextField();
    Photo = new javax.swing.JPanel();
    jPanel4 = new javax.swing.JPanel();
    jLabel3 = new javax.swing.JLabel();
    Parcourir = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    Annuler = new javax.swing.JButton();
    Valider = new javax.swing.JButton();

    jLabel1.setText("Saisie d'une nouvelle installation");
    getContentPane().add(jLabel1, java.awt.BorderLayout.NORTH);
    jPanel1.setLayout(new java.awt.GridLayout(2, 2));
    jPanel6.setLayout(new java.awt.GridLayout(2, 1));
    jLabel2.setText("Catégorie");
    jPanel6.add(jLabel2);
    PrixProp.setText("Prix proposé");
    jPanel6.add(PrixProp);
    jPanel1.add(jPanel6);
    jPanel5.setLayout(new java.awt.GridLayout(2, 0));
    jPanel5.add(Categorie);
    jPanel5.add(Prix);
    jPanel1.add(jPanel5);
    Photo.setLayout(new java.awt.GridLayout(1, 1));
    jPanel1.add(Photo);
    jPanel4.setLayout(new java.awt.GridLayout(2, 1));
    jLabel3.setText("Photo");
    jPanel4.add(jLabel3);
    Parcourir.setText("Parcourir");
     . . .
    jPanel4.add(Parcourir);
    jPanel1.add(jPanel4);
    getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
    jPanel2.setLayout(new java.awt.GridLayout(1, 2));
    Annuler.setText("Annuler");
    . . .
    jPanel2.add(Annuler);
    Valider.setText("Valider");
    . . .
    jPanel2.add(Valider);
    getContentPane().add(jPanel2, java.awt.BorderLayout.SOUTH);
    . . .
}
```