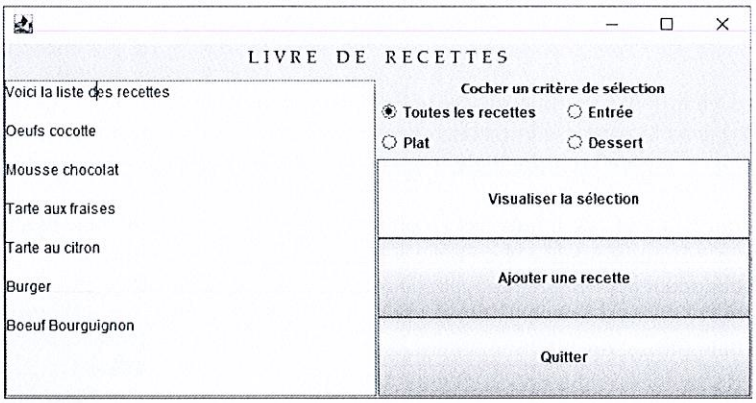
Licence Informatique 1ère année
Module Info2b – Interfaces visuelles

Examen – Durée 2h – Vendredi 12 mai 2023
Documents autorisés (polycopiés de CM, TD et TP)

Un étudiant stagiaire a été chargé de développer une application permettant de gérer un livre de recettes. Il a commencé avec une première version qui a été enrichie avec des fonctionnalités supplémentaires. L'interface du nouveau prototype est la suivante :



Elle comporte :
- En haut un titre
- A droite, se trouvent des outils, pour gérer le livre de recettes, dont :
  - Une série de boutons radios exclusifs afin de préciser le critère de recherche des recettes que l'on souhaite obtenir (Toutes les recettes, les entrées, les plats, les desserts)
  - Un bouton pour visualiser, dans la partie gauche de l'interface, les recettes répondant au critère de sélection
  - Un bouton qui permet d'ajouter une recette au livre de recettes
  - Un bouton pour quitter l'application
- A gauche une zone d'édition, nommée « Edition », où s'affichent le nom des recettes correspondant au critère de sélection défini avec les boutons radios.

Pour gérer les données de cette application, une classe nommée « Recette » a été créée. La classe « Recette » modélise une recette avec son nom, sa description, sa catégorie qui peut être « Entrée », « Plat » ou « Dessert », la liste de ses ingrédients et sa photo. La photo d'une recette est gérée à l'aide d'une image de type « ImageIcon ». Toutes les images sont dans un dossier « images » situé dans le répertoire « src » de l'application.

L'ensemble des recettes est géré directement à l'aide d'une liste de type ArrayList<Recette>.

L'annexe 1 détaille partiellement la classe « Recette » et l'annexe 2 décrit partiellement la classe « LivreRecettes».

En utilisant le code de la classe « Recette » fourni en annexe 1:

1. (1 pt) Rédiger le code de l'accesseur en lecture « getIngredients() » qui permet de renvoyer la liste des ingrédients d'une recette.

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

La classe « LivreRecettes » comporte deux attributs.

- L'attribut « livre » qui est une liste qui contient toutes les recettes disponibles.
- L'attribut « laSelection » qui comporte soit toutes les recettes si le bouton radio « Toutes les recettes » est coché, soit uniquement les recettes de la catégorie sélectionnée par les boutons radio « Entrée », « Plat » ou « Dessert ».

2. (1,5 pts) Ecrire le code java de la méthode « *private String NomsRecettes()* » de la classe « LivreRecettes » qui retourne sous forme de chaîne de caractères les noms des recettes contenues dans la liste de recettes de l'attribut « laSelection ».
3. (1 pt) Expliquer, en une phrase, le rôle de la méthode « *public ArrayList<Recette> recherche(String c)* ».
4. (1,5 pts) Expliquer, ligne par ligne avec des phrases, le code du gestionnaire d'événement « *private void RBToutesActionPerformed (java.awt.event.ActionEvent evt)* » qui est appelé lorsque le bouton radio « Toutes les recettes » est cliqué.

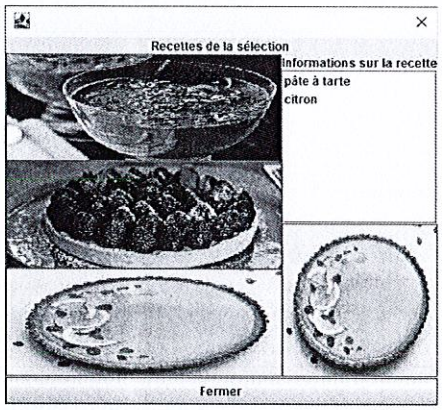
5. (1,5 pts) Lors du clic sur le bouton radio « Entrée », nommé « BREntree », seules les recettes concernant des entrées sont affectées dans l'attribut « laSelection » et leurs noms apparaissent dans la zone d'édition nommée « Edition » précédés du message « Voici la liste des entrées ». Rédiger le code java de la méthode « *private void BREntreeActionPerformed (java.awt.event.ActionEvent evt)* » en se basant sur la méthode précédente.

En utilisant l'extrait du code fourni en annexe 3 :

6. (3 pts) Proposer une description de l'interface de la fenêtre nommée « LivreRecettes » de type JFrame sous la forme d'une arborescence avec les types de composant, leur nom et si besoin leur valeur.

Dans l'application, la sélection d'un critère par l'intermédiaire des boutons radio affiche la liste des noms des recettes souhaitées (toutes, entrées, plats ou desserts) dans la zone d'édition. En cliquant sur le bouton « Visualiser la sélection », cela ouvre une boîte de dialogue nommée « ConsultDlg ». La fenêtre principale transmet à la boîte de dialogue la liste des recettes qu'elle doit afficher.

A l'ouverture de la boîte de dialogue, toutes les photos des recettes sélectionnées sont affichées dynamiquement dans le panneau de gauche sur des boutons de type « BoutonImage ». La classe « BoutonImage », qui figure dans l'annexe 5, est celle vue en CM.



Voici l'interface de la boîte de dialogue « ConsultDlg » dont un extrait du code est fourni dans l'annexe 5. Elle comporte les composants graphiques suivants :

- à gauche, un panneau de type JPanel, nommé « PanRecettes ». Il est construit dynamiquement à l'ouverture de la JDialog par la méthode « initPanRecettes() »
- à droite, un panneau de type JPanel nommé « PanInfos », contient déjà un Jlabel pour le titre, et une liste de type JList, nommée « ListeIngredients ». Le bouton situé sous la JList, de type « BoutonImage », est nommé « bti » et ajouté dynamiquement à l'ouverture de la boîte de dialogue.
- en bas, un bouton de type JButton, nommé « Fermer », pour fermer la fenêtre.

Un clic sur la photo d'une recette, provoque aussitôt l'affichage de la liste de ses ingrédients dans la JList nommée « ListeIngredients » ainsi que l'affichage de la photo de la recette sur le bouton « bti »

En utilisant le code de la classe « ConsultDlg » fourni en annexe 4 :

7. (1 pt) Expliquer le rôle des attributs de la classe « ConsulDlg »
8. (1 pt) Expliquer le cas échéant, les informations reçues et renvoyées par cette boîte de dialogue.
9. (2,5 pts) La méthode « *private void initListeIngredients()* » permet de vider la liste déroulante de type « JList » nommée « ListeIngredients », puis de la remplir avec les ingrédients de la recette choisie quand elle existe. Donner le code java de cette méthode.
10. (3 pts) Compléter le code de la méthode « *private void initPanRecettes()* » qui initialise l'interface en créant, dans le panneau « PanRecettes », autant de boutons de type « BoutonImage » qu'il y a de recettes à afficher. Chacun des boutons devra avoir comme valeur de l'attribut « name » la position de la recette dans la collection, devra être abonné à un événement de type « ActionPerformed » qui exécutera le code contenu dans la méthode « btActionPerformed », et devra avoir comme icône la photo de la recette.
11. (1,5 pts) Expliquer le rôle de la méthode « *private void btActionPerformed(java.awt.event.ActionEvent evt)* ».

Gestionnaire du clic sur le bouton « Visualiser » de la classe principale « LivreRecettes »

12. (1,5 pts) Rédiger le code java de la méthode « *private void BVisualiserActionPerformed (java.awt.event.ActionEvent evt)* » qui, lorsque l'utilisateur clique sur le bouton « BVisualiser » de l'application principale, ouvre la boîte de dialogue « ConsultDlg » en lui fournissant la liste des recettes qui correspondent au critère de sélection. Au retour, l'interface devra être remise dans son état initial, à savoir, avec toutes les recettes sélectionnées.

ANNEXE 1 : Classe « Recette »

```
public class Recette {
   private String nom;                          // intitulé de la recette
   private String description;                  // explications
   private String categorie;                    // catégorie de la recette : Entrée, Plat ou Dessert.
   private ArrayList<String> lstIngredients;    // liste des ingrédients
   private ImageIcon photo;                     // photo de la recette


   public Recette()
   {  this.nom="Recette";
      this.lstIngredients= new ArrayList<String>();
      this.photo=new ImageIcon (getClass().getResource("/images/receteDefaut.png"));
      this.description="Description";
      this.categorie="";
   }

    public Recette(String n, String d, String cat)
   {  this.nom=n;
      this.lstIngredients= new ArrayList<String>();
      this.photo=new ImageIcon (getClass().getResource("/images/receteDefaut.png"));
      this.description=d;
      this.categorie=cat;
   }

   public String getNom() { return nom;}
   public void setNom(String nom) {this.nom = nom;}
   public String getCategorie() {return categorie;}
   public void setCategorie(String categorie) {this.categorie = categorie;}
   public ImageIcon getPhoto() {return photo;}
   public void setPhoto(ImageIcon photo) {this.photo = photo;}
   public String getDescription() {return description;}
   public void setDescription(String description) {this.description = description;}

   public void ajoutIngredient(String ig)
   { this.lstIngredients.add(ig);}


   public ArrayList<String> getIngredients()
   {// à compléter}

   public String toString() {
      String res= "";
      res+= "Recette : "+this.nom+"\n";
      res+= "Description : "+this.description+"\n";
      res+="Ingrédients :\n";
      for (int i= 0; i < this.lstIngredients.size(); i++)
         res+= "- "+this.lstIngredients.get(i)+"\n";
      return res;
   }

}
```

ANNEXE 2 : Extrait de la classe « LivreRecettes »

```
public class LivreRecettes extends java.awt.Frame {

  private ArrayList<Recette> livre;
  private ArrayList<Recette> laSelection;            // collection des recettes sélectionnées

  public LivreRecettes() {
     initComponents();
     this.livre = new ArrayList<Recette>();
     this.initLivreRecettes();
     RBToutes.setSelected(true);            // RBToutes = bouton radio correspondant à Toutes les recettes
     this.laSelection = this.livre;
     Edition.setText("Voici la liste des recettes \n");
     Edition.append(this.NomsRecettes());
  }

  public void initLivreRecettes()
  { Recette r;
    r=new Recette("Oeufs cocotte", "Oeufs cocote au four", "Entrée");
    r.ajoutIngredient("oeufs");
    r.ajoutIngredient("crème");
    r.ajoutIngredient("gruyère");
    r.setPhoto(new ImageIcon(getClass().getResource("/images/oeufsCocotte.jpg")));
    this.livre.add(r);
    ...
  }

  public ArrayList<Recette> recherche(String c)   // à expliquer
  {
     ArrayList<Recette> ll= new ArrayList<Recette>();
     for (int i=0; i<livre.size(); i++)
        if (livre.get(i).getCategorie().equals(c))
           ll.add(livre.get(i));
     return ll;
  }

 private String NomsRecettes() { // restitue la liste des noms des recettes contenues dans laSelection
          // à compléter
}

  private void RBToutesActionPerformed(java.awt.event.ActionEvent evt) {          // à commenter
         this.laSelection = this.livre;
        Edition.setText("Voici la liste des recettes \n");
        Edition.append(this.NomsRecettes());    }

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

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


   ...


}
```

ANNEXE 3 : Extrait de « initComponents() »

```
private void initComponents() {

    ...

    jPanel1.setLayout(new java.awt.GridLayout(1, 2));
    jScrollPane2.setViewportView(Edition);
    jPanel1.add(jScrollPane2);

    jPanel4.setLayout(new java.awt.GridLayout(4, 1));

    jPanel5.setLayout(new java.awt.BorderLayout());

    jPanel6.setLayout(new java.awt.GridLayout(2, 2));
    RBToutes.setText("Toutes les recettes");
    jPanel6.add(RBToutes);

    RBEntree.setText("Entrée");
    jPanel6.add(RBEntree);

    RBPlat.setText("Plat");
    jPanel6.add(RBPlat);

    RBDessert.setText("Dessert");
    jPanel6.add(RBDessert);
    jPanel5.add(jPanel6, java.awt.BorderLayout.CENTER);

    jLabel1.setText("Cocher un critère de sélection");
    jPanel5.add(jLabel1, java.awt.BorderLayout.NORTH);
    jPanel4.add(jPanel5);

    BVisualiser.setText("Visualiser la sélection");
    BVisualiser.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {     BVisualiserActionPerformed(evt);    }     });
    jPanel4.add(BVisualiser);

    BAjouter.setText("Ajouter une recette");
    BAjouter.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {         BAjouterActionPerformed(evt);     }     });
    jPanel4.add(BAjouter);

    BQuitter.setText("Quitter");
    BQuitter.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {     BQuitterActionPerformed(evt);     }     });
    jPanel4.add(BQuitter);

    jPanel1.add(jPanel4);

    getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

    jPanel2.setLayout( new java.awt.FlowLayout());
    LTitre.setText("L I V R E   D E   R E C E T T E S");
    jPanel2.add(LTitre);
    getContentPane().add(jPanel2, java.awt.BorderLayout.NORTH);

}
```

```
jPanel1 = new javax.swing.JPanel();
jScrollPane2 = new javax.swing.JScrollPane();
Edition = new javax.swing.JTextArea();
jPanel4 = new javax.swing.JPanel();
jPanel5 = new javax.swing.JPanel();
jPanel6 = new javax.swing.JPanel();
RBToutes = new javax.swing.JRadioButton();
RBEntree = new javax.swing.JRadioButton();
RBPlat = new javax.swing.JRadioButton();
RBDessert = new javax.swing.JRadioButton();
jLabel1 = new javax.swing.JLabel();
BVisualiser = new javax.swing.JButton();
BAjouter = new javax.swing.JButton();
BQuitter = new javax.swing.JButton();
jPanel2 = new javax.swing.JPanel();
LTitre = new javax.swing.JLabel();
```

ANNEXE 4 : Extrait de la classe « ConsultDlg »

```
public class ConsultDlg extends javax.swing.JDialog {

  private ArrayList<Recette> lesR;
  private Recette rc;
  private BoutonImage bti;

  public ConsultDlg(java.awt.Frame parent, boolean modal, ArrayList<Recette> lr) {
     super(parent, modal);
     initComponents();
     this.lesR = lr;
     this.rc = null;
     DefaultListModel mod= new DefaultListModel();
     this.ListeIngredients.setModel(mod);
     this.bti = new BoutonImage();
     this.bti.setImage(new ImageIcon(getClass().getResource("/images/receteDefaut.png")).getImage());
     PanInfos.add(bti);
     initPanRecettes();
  }

  private void initListeIngredients()
  {
        // à compléter
  }

   private void initPanRecettes()
   {   int nbr= this.lesR.size();
      int nblig= 3;
      int nbcol = nbr/3;
      PanRecettes.setLayout(new GridLayout(nblig, nbcol+1));
      // à compléter
   }

   private void btActionPerformed(java.awt.event.ActionEvent evt)    // à expliquer
   {
      BoutonImage bt=(BoutonImage) evt.getSource();
      int ind= Integer.parseInt(bt.getName());
      this rc = this.lesR.get(ind);
      initListeIngredients();
      this.bti.setImage(this.rc.getPhoto().getImage());
   }

  private void BFermerActionPerformed(java.awt.event.ActionEvent evt) {
     this.setVisible(false);
    dispose() ;
  }
  ...
}
```

ANNEXE 5 : Classe « BoutonImage »

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

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

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

    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));
       }
    }
}
```