Université Hassan 1er

École Nationale des Sciences Appliquées – Berrechid

POLYCOPIE DES TRAVAUX PRATIQUES

INTERFACE HOMME-MACHINE

PROFESSEUR : LAHCEN MOUMOUN

Sommaire

Introduction

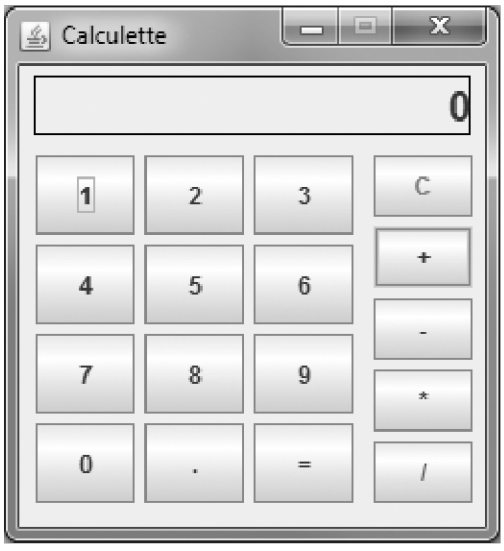
Les Interfaces Homme-Machine (IHM) constituent une composante essentielle du développement logiciel, permettant l'interaction entre l'utilisateur et une application. Dans le monde Java, il existe plusieurs bibliothèques graphiques (Swing JavaFx,…) qui offrent des outils puissants et flexibles pour concevoir des interfaces ergonomiques et dynamiques.

Ce polycopié a pour objectif d'accompagner les étudiants dans l'apprentissage et la maîtrise de la création d'IHM. Il aborde les principes fondamentaux du développement d'interfaces graphique en Java pour une maîtriser des différents composants et contrôles de ces interfaces (JFrame, JPanel, JButton, JLabel, etc.) et une bonne gestion des différents événements utilisateur qui leur sont associés(gestion des clics, saisies, etc.).

TP N°1 : Concept de base des IHMs (calculatrice)

L'objectif de ce travail est de réaliser une petite calculatrice basique (il faut générer un .jar exécutable) permettant de :

- Effectuer un calcul simple : 12 + 3.
- Faire des calculs à la chaîne, par exemple : 1 + 2 + : : : ; lorsqu'on clique à nouveau sur un opérateur, il faut afficher le résultat du calcul précédent.
- Donner la possibilité de tout recommencer à zéro.
- Gérer l'exception d'une division par 0 !
- L'interface graphique à respecter est la suivante :



Il faut prévoir deux booléens : un pour savoir si un opérateur a été sélectionné et un autre pour savoir si nous devons effacer ce qui figure à l'écran et écrire un nouveau nombre.

Elément de correction

```
import java.awt.*;   import java.awt.event.*;   import javax.swing.*;
public class Calculatrice extends JFrame {
private JPanel container = new JPanel();
    //Tableau stockant les éléments à afficher dans la calculatrice
       String[] tab_string = {"1", "2", "3", "4", "5", "6", "7", "8",  "9", "0",
             ".", "=", "C", "+", "-", "*", "/"};
    JButton[] tab_button =new JButton[tab_string.length];//bouton par élément
    private JLabel ecran = new JLabel();
    private Dimension dim = new Dimension(50, 40);
    private Dimension dim2 = new Dimension(50, 31);
    private double chiffre1;
    private boolean clicOperateur = false, update = false;
    private String operateur = "";
    public Calculatrice(){
      this.setSize(240, 260); this.setTitle("Calculette");
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      this.setLocationRelativeTo(null); this.setResizable(false);
      initComposant();  //On initialise le conteneur avec tous les composants
      this.setContentPane(container); //On ajoute le conteneur
      this.setVisible(true);  }
    private void initComposant(){
      Font police = new Font("Arial", Font.BOLD, 20); // police d'écriture
      ecran = new JLabel("0"); ecran.setFont(police);
      ecran.setHorizontalAlignment(JLabel.RIGHT);//alignement à droite
      ecran.setPreferredSize(new Dimension(220, 20));
      JPanel operateur = new JPanel();
      operateur.setPreferredSize(new Dimension(55, 225));
      JPanel chiffre = new JPanel();
      chiffre.setPreferredSize(new  Dimension(165, 225));
      JPanel panEcran = new JPanel();
      panEcran.setPreferredSize(new Dimension(220, 30));
```

```
for(int i=0;i<tab_string.length;i++){ //création des boutons
   tab_button[i] = new JButton(tab_string[i]);
   tab_button[i].setPreferredSize(dim);
   switch(i){
      //Pour chaque élément qui n'est pas un chiffre on définit
      // le comportement à avoir grâce à un listener
      case 11 : tab_button[i].addActionListener(new EgalListener());
                chiffre.add(tab_button[i]); break;
      case 12 : tab_button[i].setForeground(Color.red);
            tab_button[i].addActionListener(new ResetListener());
            operateur.add(tab_button[i]); break;
      case 13 : tab_button[i].addActionListener(new PlusListener());
            tab_button[i].setPreferredSize(dim2);
            operateur.add(tab_button[i]); break;
      case 14 : tab_button[i].addActionListener(new MoinsListener());
            tab_button[i].setPreferredSize(dim2);
            operateur.add(tab_button[i]); break;
      case 15 : tab_button[i].addActionListener(new MultiListener());
            tab_button[i].setPreferredSize(dim2);
            operateur.add(tab_button[i]); break;
      case 16 : tab_button[i].addActionListener(new DivListener());
            tab_button[i].setPreferredSize(dim2);
            operateur.add(tab_button[i]); break;
      default :  //Par défaut, ce sont les premiers éléments du tableau
                 //donc des chiffres, on affecte alors le bon listener
            chiffre.add(tab_button[i]);
            tab_button[i].addActionListener(new ChiffreListener());
             break;
   }
}
panEcran.add(ecran);
panEcran.setBorder(BorderFactory.createLineBorder(Color.black));
container.add(panEcran, BorderLayout.NORTH);
```

```
    container.add(chiffre, BorderLayout.CENTER);
    container.add(operateur, BorderLayout.EAST);
  }
  private void calcul(){
    //Méthode permettant d'effectuer un calcul selon l'opérateur sélectionné
    if(operateur.equals("+")){
      chiffre1 = chiffre1 + Double.valueOf(ecran.getText()).doubleValue();
      ecran.setText(String.valueOf(chiffre1));
    }
    if(operateur.equals("-")){
      chiffre1 = chiffre1 - Double.valueOf(ecran.getText()).doubleValue();
      ecran.setText(String.valueOf(chiffre1));
    }
    if(operateur.equals("*")){
      chiffre1 = chiffre1 * Double.valueOf(ecran.getText()).doubleValue();
      ecran.setText(String.valueOf(chiffre1));
    }
    if(operateur.equals("/")){
      try{
          chiffre1 = chiffre1 /
                 Double.valueOf(ecran.getText()).doubleValue();
          ecran.setText(String.valueOf(chiffre1));
      } catch(ArithmeticException e) {        ecran.setText("0"); }
    }
  }
  //Listener utilisé pour les chiffres Permet de stocker
  //les chiffres et de les afficher
  class ChiffreListener implements ActionListener {
    public void actionPerformed(ActionEvent e){
      //On affiche le chiffre additionnel dans le label
      String str = ((JButton)e.getSource()).getText();
      if(update){ update = false; }
          else{
```

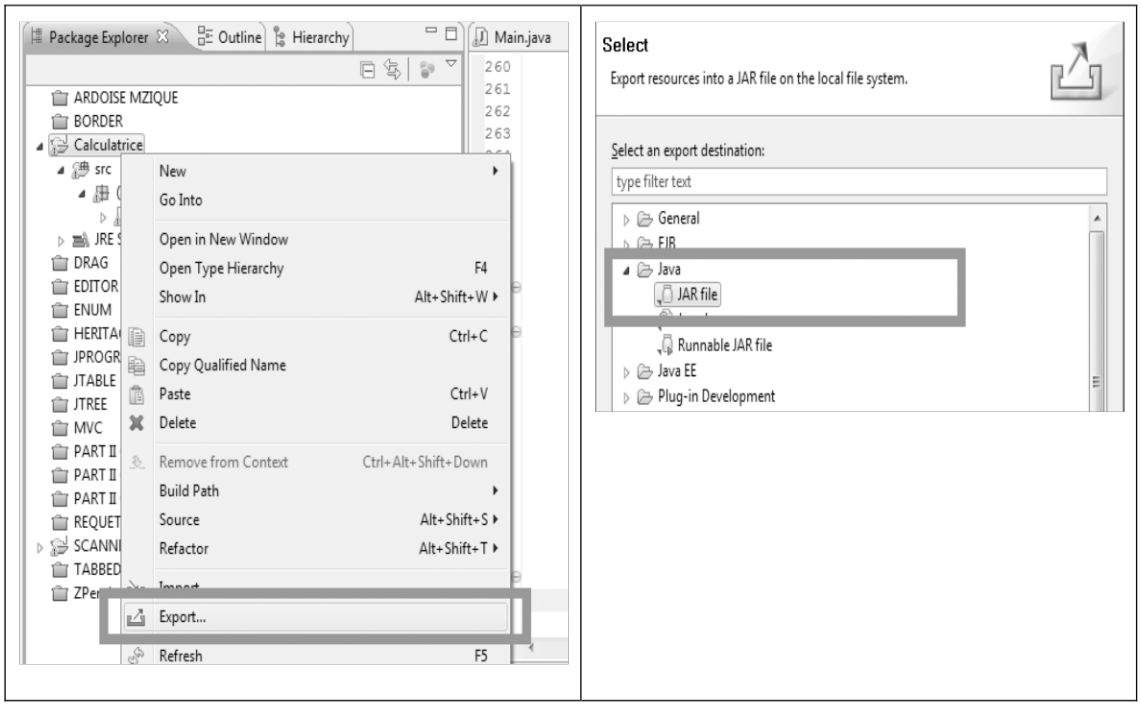
```
            if(!ecran.getText().equals("0")) str = ecran.getText() + str; }
        ecran.setText(str);
    }
}
//Listener affecté au bouton =
class EgalListener implements ActionListener {
    public void actionPerformed(ActionEvent arg0){
        calcul(); update = true;  clicOperateur = false;
     }
}
//Listener affecté au bouton +
class PlusListener implements ActionListener {
    public void actionPerformed(ActionEvent arg0){
        if(clicOperateur){ calcul();ecran.setText(String.valueOf(chiffre1));}
            else{chiffre1 = Double.valueOf(ecran.getText()).doubleValue();
                   clicOperateur = true; }
        operateur = "+";  update = true;
    }
}
//Listener affecté au bouton -
class MoinsListener implements ActionListener {
    public void actionPerformed(ActionEvent arg0){
        if(clicOperateur){ calcul(); ecran.setText(String.valueOf(chiffre1));
    }
            else{ chiffre1 = Double.valueOf(ecran.getText()).doubleValue();
                  clicOperateur = true; }
        operateur = "-"; update = true;
    }
}
//Listener affecté au bouton *
class MultiListener implements ActionListener {
    public void actionPerformed(ActionEvent arg0){
        if(clicOperateur){
```

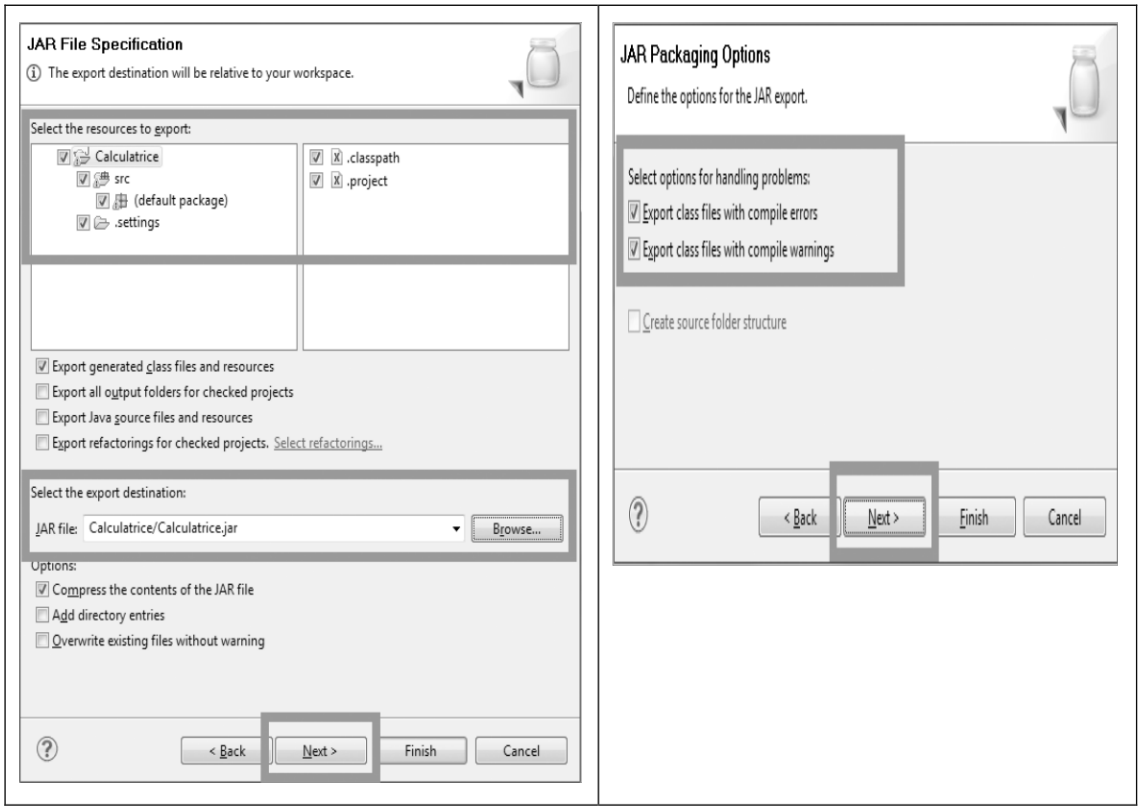
```
                calcul();  ecran.setText(String.valueOf(chiffre1)); }
            else{  chiffre1  =  Double.valueOf(ecran.getText()).doubleValue();
      clicOperateur = true;
         }
         operateur = "*"; update = true;
      }
    }

    //Listener affecté au bouton /
    class DivListener implements ActionListener {
      public void actionPerformed(ActionEvent arg0){
         if(clicOperateur){
            calcul(); ecran.setText(String.valueOf(chiffre1)); }
            else{ chiffre1 = Double.valueOf(ecran.getText()).doubleValue();
                  clicOperateur = true; }
         operateur = "/"; update = true;
      }
    }
    //Listener affecté au bouton de remise à zéro
    class ResetListener implements ActionListener {
      public void actionPerformed(ActionEvent arg0){
         clicOperateur = false; update = true; chiffre1 = 0; operateur = "";
         ecran.setText("");
      }
    }
}
public class Main {
    public static void main(String[] args) {
      Calculatrice calculette = new Calculatrice();
    }
}
```

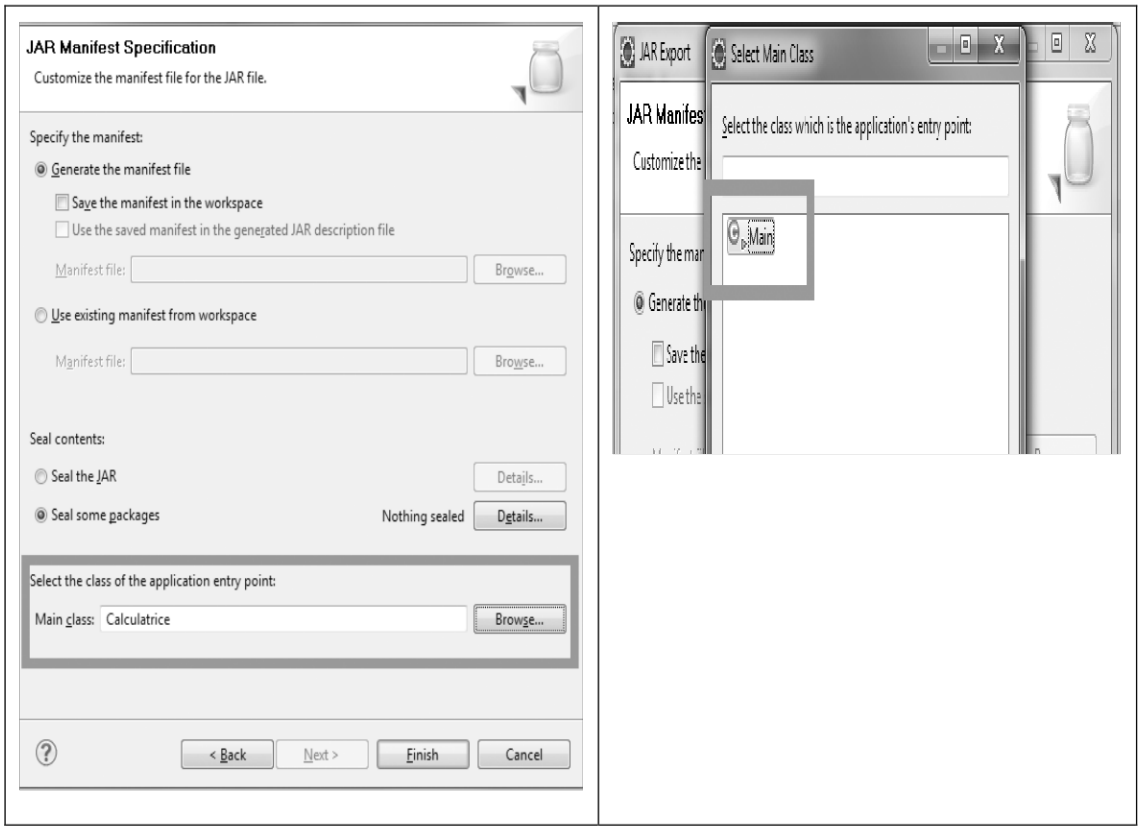
Élaboration du fichier jar

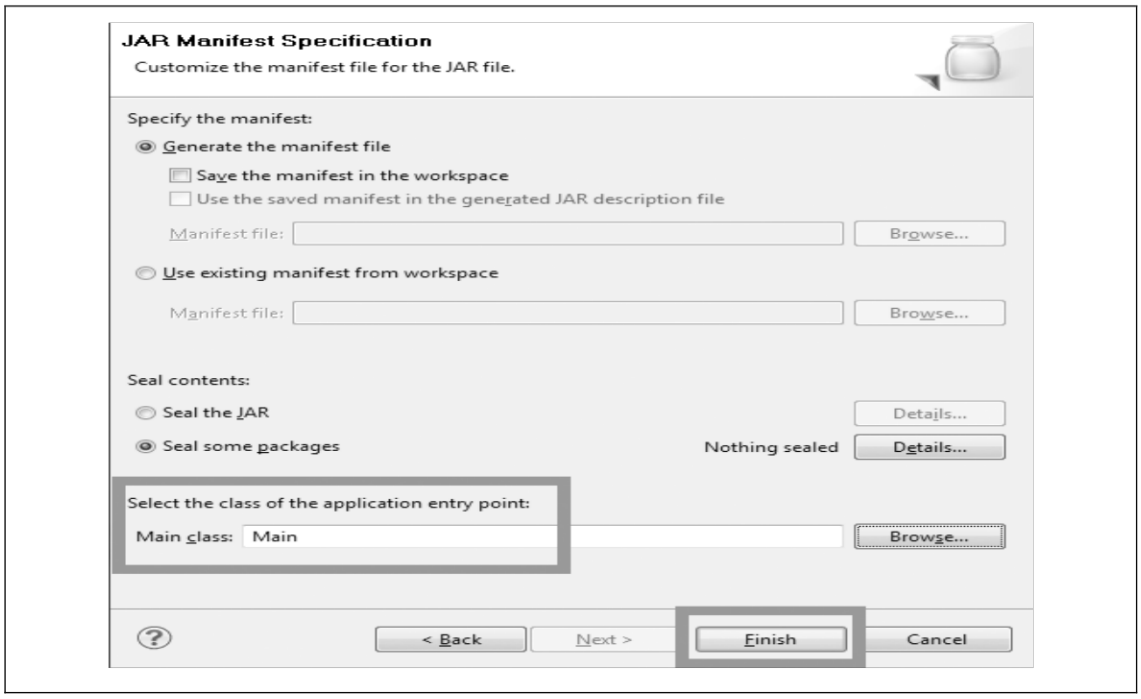
Un .jar est une extension propre aux archives Java (Java ARchive). Ce type de fichier contient tout ce dont a besoin la JVM pour lancer un programme.

- Commencer par effectuer un clic droit sur le projet et choisir l'option Export, comme le montre la première figure ci-dessous.
- Dans la gestion des exports. Il faut sélectionner JAR File puis cliquez sur Next.
- Sectionner les fichiers qu'on souhaite inclure dans l'archive : Dans le premier cadre, sélectionnez tous les fichiers qui composeront l'exécutable (.jar) puis dans le second cadre, indiquer à Eclipse l'endroit où créer l'archive et le nom à lui donner. Ensuite, cliquer sur Next.
- Cliquer sur Next pour spécifier l'emplacement de la méthode main dans le programme.
- Cliquer sur Browse pour afficher un pop-up listant les fichiers des programmes contenant une méthode main.
- Sélectionner le point de départ de l'application et valider Ensuite cliquer sur Finish .




JAR File Specification
The export destination will be relative to your workspace.
Select the resources to export:
Calculatrice
src
(default package)
.settings
.classpath
.project
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file: Calculatrice/Calculatrice.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Finish
Cancel
JAR Packaging Options
Define the options for the JAR export.
Select options for handling problems:
Export class files with compile errors
Export class files with compile warnings
Create source folder structure
< Back
Next >
Finish
Cancel


JAR Manifest Specification
Customize the manifest file for the JAR file.
Specify the manifest:
Generate the manifest file
Save the manifest in the workspace
Use the saved manifest in the generated JAR description file
Manifest file:
Browse...
Use existing manifest from workspace
Manifest file:
Browse...
Seal contents:
Seal the JAR
Details...
Seal some packages
Nothing sealed
Details...
Select the class of the application entry point:
Main class: Calculatrice
Browse...
< Back
Next >
Finish
Cancel
JAR Export
Select Main Class
Select the class which is the application's entry point:
Main


JAR Manifest Specification
Customize the manifest file for the JAR file.
Specify the manifest:
Generate the manifest file
Save the manifest in the workspace
Use the saved manifest in the generated JAR description file
Manifest file:
Browse...
Use existing manifest from workspace
Manifest file:
Browse...
Seal contents:
Seal the JAR
Details...
Seal some packages
Nothing sealed
Details...
Select the class of the application entry point:
Main class: Main
Browse...
< Back
Next >
Finish
Cancel

TP N°2 : Contrôles usuels des IHMs (sélecteur RGB)

La figure 1 (voir ci-dessous) représente une boite de dialogue qui permet de choisir une couleur en spécifiant chacune de ses composantes (rouge, vert, bleu), en utilisant soit les ascenseurs ou les champs de textes décimaux. La modification de l'état d'un composant doit modifier les autres composants en conséquence.

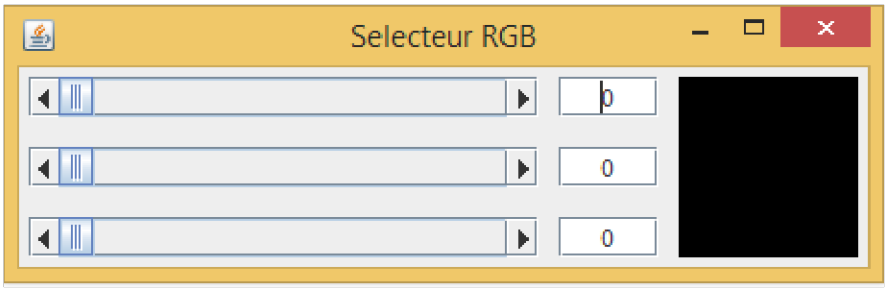


Figure 1 : Interface SelecteurRGB

1. Expliquez quels composants et gestionnaires de placement utiliser pour obtenir l'interface ci-dessus.
2. Donner le code qui assure la synchronisation entre la zone de texte et l'ascenseur correspondant à l'intensité rouge. (1ièr ascenseur et 1ière zone de texte). Assurer également l'affichage de la couleur choisie (utilisé la méthode « Color(int red, int green, int blue) »)
3. L'utilisateur peut commettre des erreurs quand il saisit des informations dans les champs de texte. On souhaite afficher un message (sous forme d'une boite de dialogue) pour informer l'utilisateur dès qu'une information saisie n'est pas correcte (valeur décimale en dehors de l'intervalle [0,255] ou caractère non numérique saisi). Donner le code permettant d'assurer cette fonctionnalité pour la zone de texte correspondant à l'intensité rouge.

Elément de correction

```
import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import org.w3c.dom.css.CSSPrimitiveValue;
import org.w3c.dom.css.RGBColor;
import java.awt.event.*;

class Panneau extends JPanel {
      Panneau(){
            //setLayout(new GridLayout(1,1,20,10));
            setPreferredSize(new Dimension(100, 100));
      }
      public void paintComponent(Graphics g){
                  Color c= new Color(0,0,0);g.setColor(c);
                  g.fillRect(5, 5,90,90);
                   //this.getWidth(), this.getHeight());
      }
}
class Fenetre extends JFrame{
            private JTextField txtR;  private JTextField txtG;
            private JTextField txtB;  private   JScrollBar sBarR;
            private   JScrollBar sBarG; private JScrollBar sBarB;
            private     Integer valeurR; private     Integer valeurG;
            private     Integer valeurB;
      public Fenetre(String titre) {
            setTitle(titre);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setSize(300, 200); setLocation(250, 150);

            sBarR= new JScrollBar(JScrollBar.HORIZONTAL, 0, 10, 0, 255);
            sBarR.setPreferredSize(new Dimension(255, 20));
```

```
sBarG= new JScrollBar(JScrollBar.HORIZONTAL, 0, 10, 0, 255);
sBarG.setPreferredSize(new Dimension(255, 20));
sBarB= new JScrollBar(JScrollBar.HORIZONTAL, 0, 10, 0, 255);
sBarB.setPreferredSize(new Dimension(255, 20));

JPanel pSBarR= new JPanel();JPanel pSBarG= new JPanel();
JPanel pSBarB= new JPanel();
pSBarR.add(sBarR);pSBarG.add(sBarG);pSBarB.add(sBarB);
JPanel pSBar= new JPanel(new GridLayout(3,1,5,5));
pSBar.add(pSBarR);pSBar.add(pSBarG);pSBar.add(pSBarB);

txtR= new JTextField("0");
txtR.setPreferredSize(new Dimension(50, 20));
txtG= new JTextField("0");
txtR.setHorizontalAlignment(SwingConstants.CENTER);
txtG.setPreferredSize(new Dimension(50, 20));
txtB= new JTextField("0");
txtB.setPreferredSize(new Dimension(50, 20));
txtG.setHorizontalAlignment(SwingConstants.CENTER);
txtB.setHorizontalAlignment(SwingConstants.CENTER);

JPanel pTxtR= new JPanel();JPanel pTxtG= new JPanel();
JPanel pTxtB= new JPanel();
pTxtR.add(txtR);pTxtG.add(txtG);pTxtB.add(txtB);
JPanel pTxt= new JPanel(new GridLayout(3,1,5,5));
pTxt.add(pTxtR);pTxt.add(pTxtG);pTxt.add(pTxtB);

getContentPane().add(pSBar, BorderLayout.WEST);
getContentPane().add(pTxt);

Panneau pan=new Panneau();
JPanel panpan= new JPanel(new BorderLayout());
JLabel l1= new JLabel (" ");
```

```
            //panpan.add(l1,BorderLayout.NORTH);
            panpan.add(pan,BorderLayout.CENTER);
            //panpan.add(l1,BorderLayout.SOUTH);

            getContentPane().add(panpan, BorderLayout.EAST);

            this.pack();

            sBar.addAdjustmentListener(new AdjustmentListener() {
            public void adjustmentValueChanged(AdjustmentEvent ae) {
            if (sBar.getValueIsAdjusting()) return;
                  valeur=ae.getValue(); txt1.setText(valeur.toString());});
            btnInit.addActionListener( new ActionListener(){
            public void actionPerformed(ActionEvent arg0) {
            valeur=0; sBar.setValue(0); txt1.requestFocus(); }    }   );
            txt1.getDocument().addDocumentListener(new DocumentListener() {
            public void removeUpdate(DocumentEvent e) {
            valeur=Integer.parseInt(txt1.getText()); sBar.setValue(valeur);}
            public void insertUpdate(DocumentEvent e) {
            valeur=Integer.parseInt(txt1.getText()); sBar.setValue(valeur);;}
            public void changedUpdate(DocumentEvent e) { }     });

      }
}
public class Ex1 {
    public static void main(String[] args) {
      JFrame fn= new Fenetre("Selecteur RGB");
      fn.setVisible(true);
      }
}
```

TP N°3 : Contrôles usuels des IHMs (poids idéal)

La figure 1 (voir ci-dessous) représente une application qui permet de calculer le poid idéal d'un individu en fonction de à sa taille (prise en cm). Cette dernière peut être saisie dans la zone de texte corespondante ou choisie directement en utilisant l'ascenseur (curseur). Le poid édéal est déterminer par la formule suivante :

- Poid = Taille * Taille / 30 pour un indévidu masculin
- Poid = Taille * Taille / 28 pour un indévidu féminin

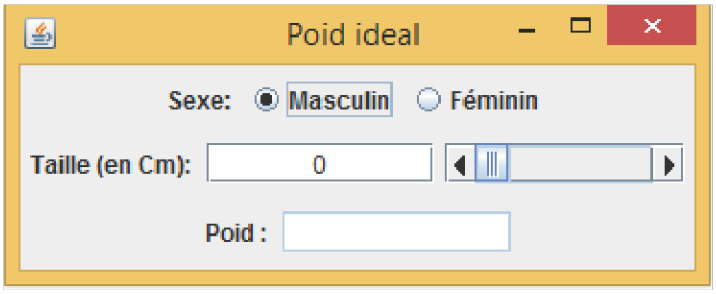
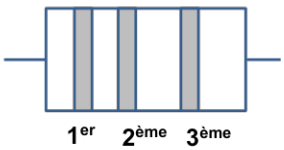


Figure 1 : Interface poid idéal

1. Ecrire le code de l'application sachant qu'il faut :
2. Assurer la synchronisation entre la zone de texte correspondant à la Taille et son ascenseur
3. Interdire la saisie des caractères non numériques dans la zone de texte de la Taille
4. Calculer automatiquement le poid à cahque changement du sexe ou de la Taille.

TP N°4 : Contrôles usuels des IHMs(couleur des résistances)

Nous désirons une application qui permet de déterminer les couleurs correspondantes à la valeur d'une résistance. En effet, il est fréquent d'avoir à retrouver une résistance à partir de sa valeur, Pour ce faire on utilise un codage sous forme d'anneaux colorés. Nous nous limitons dans ce qui suit aux 3 anneaux suivant :



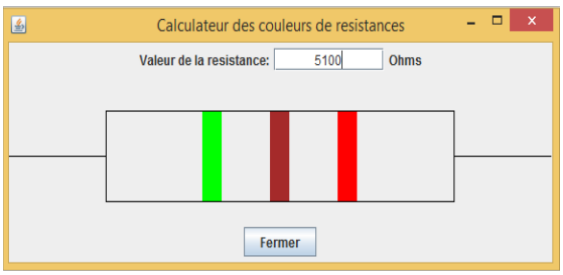
Couleur	Chiffre (a,b ou n)
Noir	0
Marron	1
Rouge	2
Orange	3
Jaune	4
Vert	5
Bleu	6
Violet	7
Gris	8
Blanc	9

- Le 1er anneau précise le premier chiffre significatif « a » de la valeur de R.
- Le 2ème anneau indique le deuxième chiffre significatif « b » de la valeur de R.
- Le 3ème anneau précise le facteur multiplicatif 10n (nombre de zéro qui suivent les deux premiers chiffres).

Les codes colleurs retenus pour cet exercice sont représentés par le tableau ci-contre.

La valeur de la résistance R s'exprime en ohms sous la forme : $R = ab*10^n$. Par exemple pour une résistance R= 350Ω càd $R=35*10^1$, on trouve : a=3 qui correspond à une couleur «Oronge » du 1^{er} anneau , b=5 qui correspond à une couleur «Vert » du $2^{ème}$ anneau et n =1 qui correspond à une couleur «Marron» du $3^{ème}$ anneau.

Nous considérons pour cet exercice l'interface graphique ci-dessous (les contrôles de l'interface sont décrits dans le tableau) :



Type de contrôle	Nom du contrôle
JTextField	txvtR
JPanel	panDessin
JComboBox	btnfermer

La classe « Panneau » permet de dessiner la forme de la résistance, le code correspondant est le suivant :

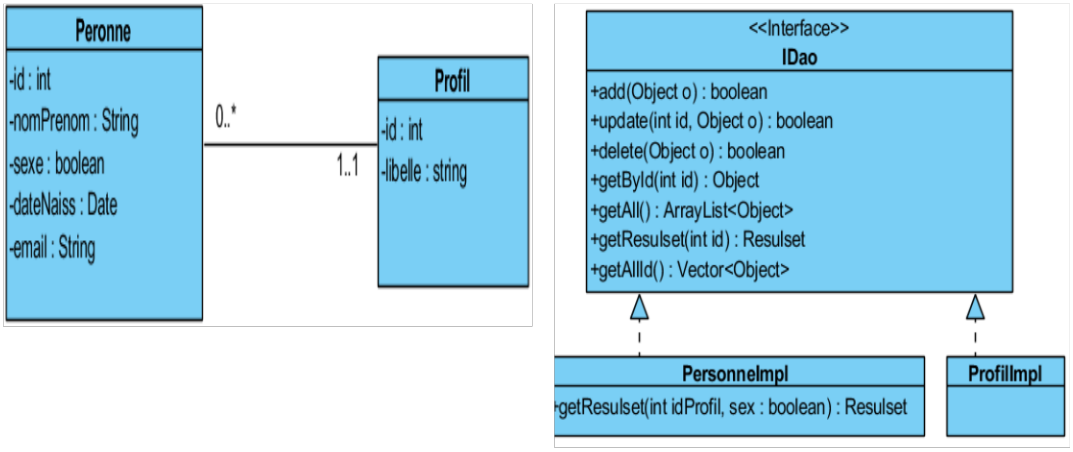
```
public class Panneau  extends JPanel{
  //tracage de la forme d'une ressistance
  private int color1 ;   private int color2 ;private int color3;
  private Color [] tabColor= { Color.BLACK, new Color(165,42,42),
          Color.RED,Color.ORANGE,Color.YELLOW,Color.GREEN,
          Color.BLUE, new Color(238, 130, 238),
          Color.GRAY, Color.WHITE };
  public void paintComponent(Graphics g){
    int posX=this.getX();int posY=this.getY();
    g.setColor(Color.BLACK);
    g.drawLine(posX,posY+40,posX+100,posY+40);
    g.drawLine(posX+460,posY+40,posX+560,posY+40);
    g.setColor(tabColor[color1]);g.fillRect(posX+200, posY, 20, 80);
    g.setColor(tabColor[color2]);g.fillRect(posX+270, posY, 20, 80);
    g.setColor(tabColor[color3]);g.fillRect(posX+340, posY, 20, 80);
    g.setColor(Color.BLACK);g.drawRect(posX+100, posY, 360, 80); }

public void setColor1(int color1) {this.color1 = color1;}
public void setColor2(int color2) {this.color2 = color2;}
public void setColor3(int color3) {this.color3 = color3;}
}
```

1. Ecrire l'extrait du code permettant d'interdire la saisie des caractères non numériques dans la zone de texte « txtR ». Cette zone de texte doit contenir au maximum 9 chiffres.
2. Donner le code permettant le calcul et la mise à jour (affichage) automatique de la couleur des anneaux à chaque changement de la valeur de la résistance.

TP N°5 : IHM et JDBC (gestion des profils)

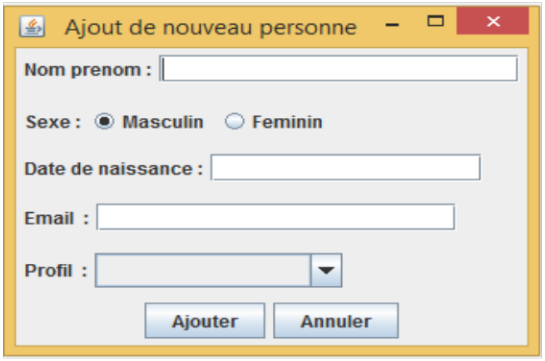
Le but de cet exercice est de créer une application de gestion des profils du personnel d'une entreprise industrielle. Les informations relatives aux personnes sont stockées dans une base de données « GestionPersonnes ».

Le diagramme de classe ci-dessous représente la structure des 2 tables « personnes » et «profils» utilisées dans cet exercice (le type des champs id est auto-increment, le champs sexe comporte la valeur 'Masculin' ou 'Feminin', les méthodes « getSexe » et « setSexe » de la classe « Personne » assure la transition String↔ boolean). L'interface «IDao» et les classes «ProfilsImpl» et « PersonneImpl » permettant d'assurer les différents services relatifs aux personnes et aux profils.



1. Définir la classe « Connexion » permettant d'obtenir une connexion unique à la base de données « GestionPersonnes ».
2. Définir les classes et l'intetrface du diagramme ci-dessus à savoir que :
    - Une méthode « add » permettant d'ajouter un nouvel enregistrement dans la table « personnes » ou « Profil ».
    - Une méthode « getAllId » permet de retourner la liste des ids de la table « personnes » ou « profils ».
    - la méthode « getResulset(int idProfil, sex :boolean) » permet de retourner le jeu d'enregistrements correspondant à l'id et au sexe fournis comme arguments.

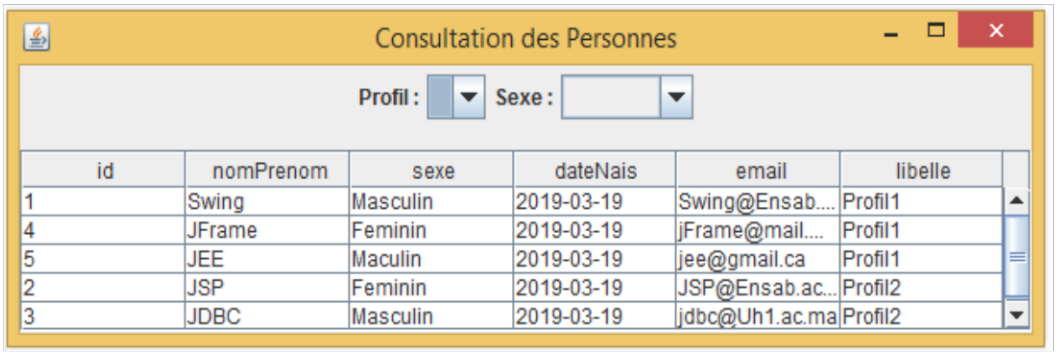
3. On considère l'interface graphique suivante permettant l'ajout de l'enregistrement, regroupant les informations d'une nouvelle personne, dans la table « personnes ». Les contrôles de l'interface sont décrits dans le tableau



Type de contrôle	Nom du contrôle
JTextField	txtNomPr
	txtDateNais
	txtEmail
JRadioButton	rbMas

Ecrire la classe correspondante à l'interface graphique ci-dessus en définissant le code à exécuter lors de l'évènement clic du bottons « Ajouter » (il faut s'assurer que les informations de l'interface sont valides).

4. On considère l'interface graphique suivante permettant la consultation des informations des personnes selon des critères de sexe et profil.



Ecrire la classe correspondante à l'interface graphique ci-dessus en réalisant les tâches suivantes :

- Creation de l'aspect visuel de l'interface graphique (avec le remplissage des JComboBox)
- Le remplissage de la grille (objet JTable) lors de l'affichage de l'interface et le changement des critères de sexe et profil (Vous pouvez utiliser la classe « ResultSetTableModel extends AbstractTableModel » étudiée dans le cours sans la redéfinir)

Elément de correction

La classe Personne

```
package bean;
import java.text.*; import java.util.*;
public  abstract class Personne {
      private String id; private String nom;  private String prenom;  private
      String mail; private String telephone;
      private String calculId(){
        String idStr =nom.substring(0,4) ;
        SimpleDateFormat formatter =
            new SimpleDateFormat ("MMYY", Locale.FRANCE);
        idStr +=formatter.format ( new Date () )  ;  return idStr;
      }
      public Personne(String nom,String prenom,String mail,String telephone){
            this.nom = nom;this.id=calculId(); this.prenom = prenom;
             this.mail = mail;this.telephone = telephone;
      }
      public String getId() { return id; }
      public String getNom() {  return nom; }
      public void setNom(String nom) {  this.nom = nom;this.id=calculId(); }
      public String getPrenom() { return prenom; }
      public void setPrenom(String prenom) {  this.prenom = prenom; }
      public String getMail() { return mail; }
      public void setMail(String mail) { this.mail = mail; }
      public String getTelephone() {      return telephone; }
      public void setTelephone(String telephone){this.telephone=telephone;}
      public String toString() {
         return " [id=" + id + ", nom=" + nom + ", prenom=" + prenom
                  + ", mail=" + mail + ", telephone=" + telephone + "]";
      }
}
```

La classe Profil

```
package bean;
public class Profil {
      private String id; private String libelle;
      public Profil(String id, String libelle) {
            this.id = id;  this.libelle = libelle;  }
      public void setId(String id) { this.id = id;  }
      public String getId() { return id;  }
      public String getLibelle() { return libelle;}
      public void setLibelle(String libelle) { this.libelle = libelle; }
      public String toString() {
            return " [id=" + id + ", libelle=" + libelle + "]"; }
}
```

La classe Utilisateur

```
package bean;
public class Utilisateur extends Personne{
      private String login; private String password;  private String service;
      private Profil profil;
      public Utilisateur(String nom,String prenom,String mail,String
            telephone,String login,String password,String service,
            Profil profil) {
         super(nom, prenom, mail, telephone);
         this.login = login; this.password = password;
         this.service = service; this.profil = profil;
      }
      public String getLogin() { return login; }
      public String getPassword() { return password; }
      public void setPassword(String password) {     this.password        =
      password; }
      public String getService() { return service; }
      public void setService(String service) {  this.service = service; }
      public Profil getProfil() {  return profil;  }
      public void setProfil(Profil profil) {  this.profil = profil;    }
```

```
      public String toString() {
        return ( super.toString()+"[login="+login+", password=" + password
                  +", service=" + service +", profil="+profil+"]");
      }
}
```

L'interface IDao

```
package dao;
import java.io.*;
import java.util.ArrayList;
public interface IDao {
      public void create(Object obj);
      public ArrayList getAll();
      public void delete(String id);
      public void update(String id);
      public Object getById(String id);
      public  void enregistrer ()throws FileNotFoundException , IOException;
}
```

La classe ProfilImpl

```
package Imp;
import java.io.*; import java.util.*;
import bean.Profil; import dao.IDao;
public class ProfilImpl implements IDao {
      private ArrayList profiles; String nomFichier;
      public   ProfilImpl(String   fichier)throws   FileNotFoundException   ,
      IOException  {
      this.nomFichier = fichier ;
      FileInputStream fis ;  ObjectInputStream ois = null ;
      try {
            fis = new FileInputStream ( fichier );
            ois = new ObjectInputStream ( fis);
            profiles = ( ArrayList ) ois.readObject (); }
      catch ( Exception e) { profiles = new ArrayList ();  }
      }
```

```
	public void create(Object obj){ profiles.add( obj);  }
	public ArrayList getAll(){ return profiles;  }
	public void delete(String id){  profiles.remove(getById(id));  }
	public void update(String id){
		Scanner sc = new Scanner(System.in);
		System.out.print("\nEntrez le nouveau libelle: ");
		((Profil)getById(id)).setLibelle(sc.nextLine());
		sc.close(); }
	public Object getById(String  id){
	for(int i = 0; i < profiles.size(); i++){
		if(((Profil)profiles.get(i)).getId().equals(id)  ){
		return profiles.get(i);  }  }
	   return null;
	}
	public String getNomFichier() { return nomFichier; }
	public void enregistrer() throws FileNotFoundException , IOException {
		FileOutputStream fos = new FileOutputStream ( nomFichier );
		ObjectOutputStream oos = new ObjectOutputStream ( fos);
		oos.writeObject ( profiles );
	}
}
```

La classe UtilisateurImpl

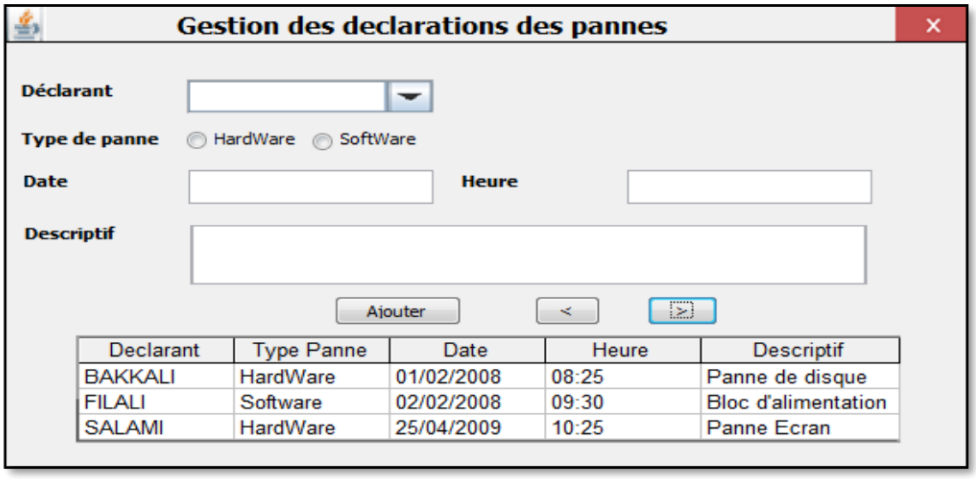
```
package Imp;
import java.io.*;  import java.util.*;
import bean.Utilisateur; import dao.IDao;
public class UtilisateurImpl implements IDao{
	private ArrayList users; String nomFichier;
	public UtilisateurImpl (String fichier)throws FileNotFoundException ,
	IOException  {
		this.nomFichier = fichier ;
		 FileInputStream fis ;  ObjectInputStream ois = null ;
	try {
		fis = new FileInputStream ( fichier );
```

```
        ois = new ObjectInputStream ( fis);
        users = ( ArrayList ) ois.readObject (); }
    catch ( Exception e) { users = new ArrayList ();  }
    }
    public void create(Object obj){ users.add(obj); }
    public ArrayList getAll(){ return users;  }
    public void delete(String id){  users.remove(getById(id));  }
    public void update(String id){
        Scanner sc = new Scanner(System.in);
        System.out.print("\nEntrez le nouveau nom: ");
        ((Utilisateur)getById(id)).setNom(sc.nextLine());
        System.out.print("\nEntrez le nouveau prenom: ");
        ((Utilisateur)getById(id)).setPrenom(sc.nextLine());
        System.out.print("\nEntrez le nouveau mail: ");
        ((Utilisateur)getById(id)).setMail(sc.nextLine());
        System.out.print("\nEntrez le nouveau telephone: ");
        ((Utilisateur)getById(id)).setTelephone(sc.nextLine());
        System.out.print("\nEntrez le nouveau password: ");
        ((Utilisateur)getById(id)).setPassword(sc.nextLine());
        System.out.print("\nEntrez le nouveau service: ");
        ((Utilisateur)getById(id)).setService(sc.nextLine());
        System.out.print("\nEntrez le nouveau libelle du profil a
                    affecter a cet " +     "utilisateur: ");
        ((Utilisateur)getById(id)).getProfil().setLibelle(sc.nextLine());
        sc.close();;
    }
    public Object getById(String  id){
        for(int i = 0; i < users.size(); i++){
              if(((Utilisateur)users.get(i)).getId().equals(id)  ){
                    return users.get(i); } }
        return null;
    }
```

```
	public Object getByName(String  nom, String pr){
		for(int i = 0; i < users.size(); i++){
		  if( ((Utilisateur)users.get(i)).getNom().equals(nom) &&
		      ((Utilisateur)users.get(i)).getPrenom().equals(pr)     ){
		      return users.get(i);  }
		}
		return null;
	}
	public String getNomFichier() { return nomFichier; }
	public void enregistrer() throws FileNotFoundException , IOException {
		FileOutputStream fos = new FileOutputStream ( nomFichier );
		ObjectOutputStream oos = new ObjectOutputStream ( fos);
		oos.writeObject ( users );
	}
}
```

TP N°6 : IHM et JDBC (gestion des pannes)

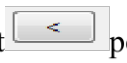
Le but de ce TP est de créer une application de gestion de déclarations des pannes dans un centre informatique. Pour des raisons de simplification, on considère l'interface graphique suivante :



Les informations relatives aux pannes sont stockées dans une base de données « GestionPannes » dont le schéma réduit est le suivant :

- Declarants(NomPrDec :varchar(20), fonctionDec : varchar(15))
- Pannes(numPan : auto_Increment, typePan : :varchar(8) : Date, heurePan: Time, descripPan: varchar, Declarant# : varchar(5))

Ecrire le code relatif à l'interface graphique ci-dessus en réalisant les tâches suivantes :

- Créer l'aspect visuel de l'interface graphique
- Réaliser les fonctionnalités suivantes :
  - Le remplissage du comboBox relatif aux déclarants des pannes et le remplissage de la grille (objet JTable) lors de l'affichage de l'interface
  - Un clic sur le bouton Ajouter permet d'ajouter les informations saisies dans les champs, dans la table Pannes et dans la grille de l'interface.
  - Le bouton précédent  permet d'afficher sur l'interface les informations relatives à la ligne précédente à la ligne courante et de la sélectionnée dans la grille et d'afficher

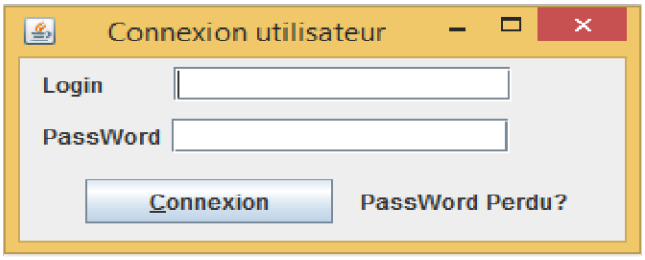
TP N°7 : IHM et JDBC (gestion de forum)

Soit le schéma suivant représentant une base de données « BDForum » utilisée de gérer un forum électronique :

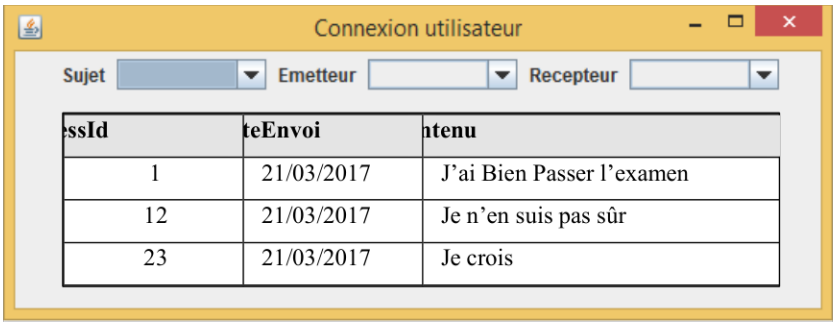
- Utilisateurs (login, dateInscr, pseudo)
- Messages (messId, emetteur#, sujet, dateEnvoi, reponseA#, contenu)

A. Considérons dans un premier lieu l'interface graphique de la figure ci-dessous qui assure la connexion d'un utilisateur.

1. Définir la classe de connexion la base de données « BDForum ».
2. Réaliser l'aspect visuelle et fonctionnel de l'interface.



B. Considérons ensuite l'interface graphique de la figure ci-dessous permettant à l'administrateur de consulter les messages de forum.

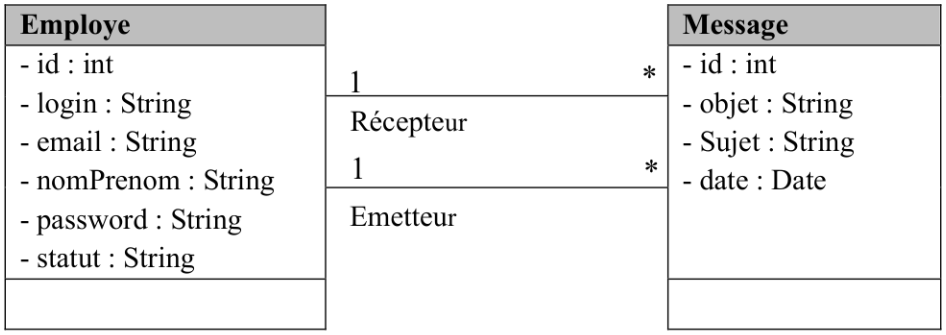


Réaliser l'aspect visuelle et fonctionnel de l'interface en assurant :

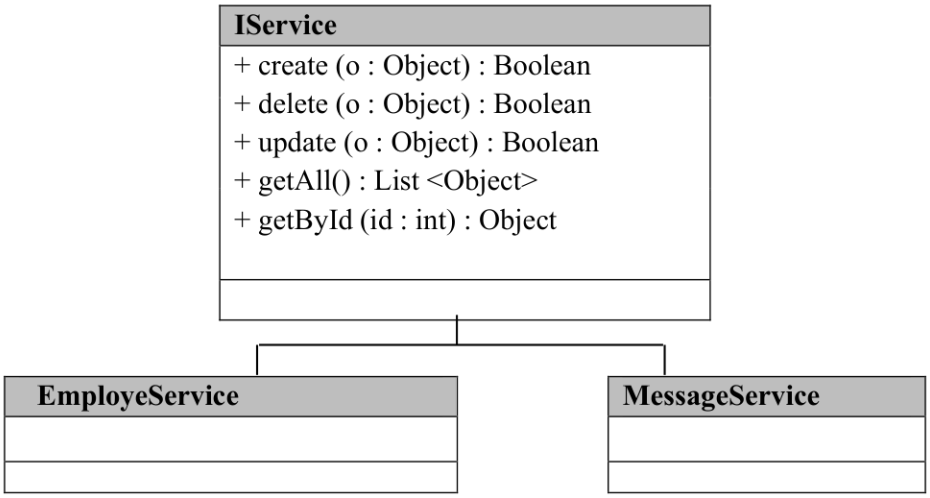
- Le remplissage des comboBox.
- Lors de la sélection d'un sujet, d'un émetteur et d'un récepteur, afficher dans un JTable les messages correspondant à ces 3 critères.

TP N°8 : IHM et JDBC (système de messagerie)

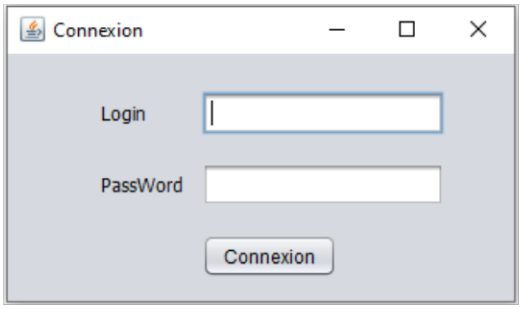
Une société souhaite mettre en place un système de messagerie entre ses employés. Les informations de ce système sont placées dans une base de données MySQL nommée « GestMessages ». Le diagramme suivant représente un extrait du diagramme de classe élaboré par l'équipe chargée de l'analyse et de la conception :

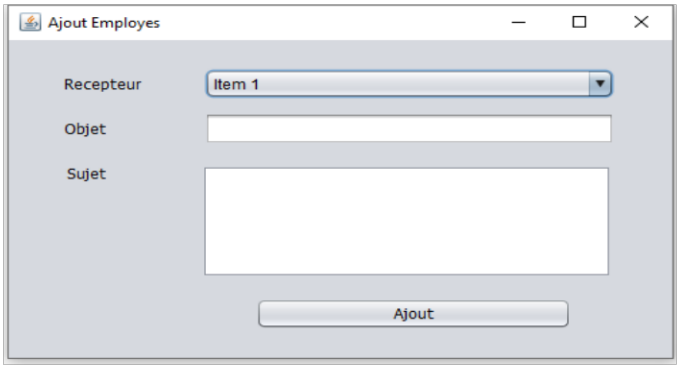


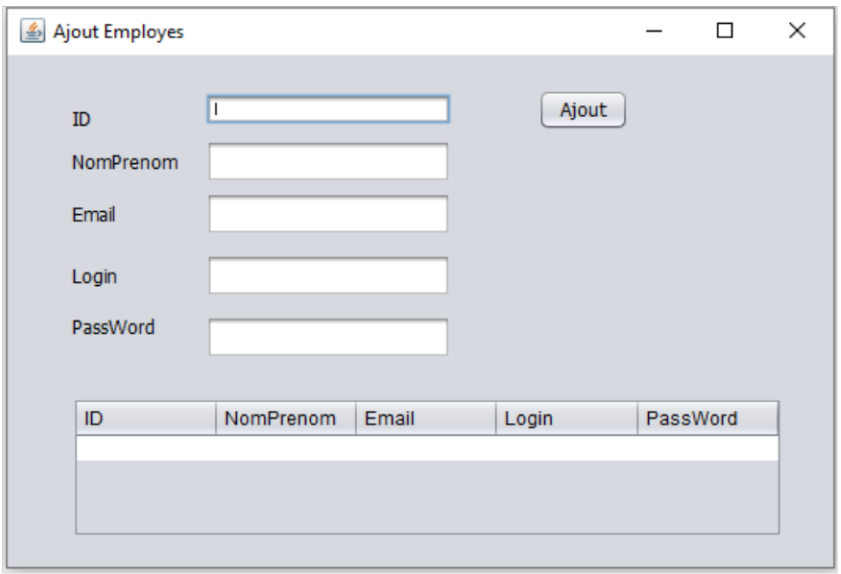
1. Créer, dans un package « dao », une classe « Connexion » permettant d'obtenir une connexion unique (instance unique pour l'ensemble des connexions) à la base de données « GestMessages ».
2. On considère l'interface IService et les classes « EmployeService » et « MessageService » placés dans le package « Metier ».



3. On considère les interfaces graphiques (à placer dans un package Vue) suivantes :







Réaliser l'aspect visuel de ces interfaces et programmer leurs différents évènements les tâches associées aux différents des interfaces.