

```

2 import java.awt.Color;
3 import java.awt.Desktop;
4 import java.awt.Dimension;
5 import java.io.*;
6 import java.util.ArrayList;
7 import java.util.Date;
8 import java.util.Iterator;
9 import java.util.Properties;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12 import javax.activation.DataHandler;
13 import javax.mail.Address;
14 import javax.mail.BodyPart;
15 import javax.mail.Flags;
16 import javax.mail.Flags.Flag;
17 import javax.mail.Folder;
18 import javax.mail.MessagingException;
19 import javax.mail.Multipart;
20 import javax.mail.NoSuchProviderException;
21 import javax.mail.Session;
22 import javax.mail.Store;
23 import javax.mail.URLName;
24 import javax.mail.internet.MimeBodyPart;
25 import javax.mail.internet.MimeMessage;
26 import javax.swing.*;
27
28 //Fenetre principale
29 public class Accueil extends javax.swing.JFrame {
30
31
32     public Accueil() {
33         initComponents();
34         this.setSize(new Dimension(1200,650));
35         this.setLocationByPlatform(true);
36         this.setLocationRelativeTo(null);
37     }
38
39
40     /* Cette connexion consiste en la connexion au serveur de messagerie,
41      * recuperation des messages et affichages de leurs entetes*/
42
43     public void connect(boolean auth){
44
45         // Afficher la boite de dialogue pour entrer les paramaitres de connexions.
46         if(auth)
47         {
48             dialog = new Authentification(this,true);
49         }
50
51
52         // Construire une URL de connexion à partir des paramètres de connexion de la boite de dialogue.
53         StringBuffer connectionUrl = new StringBuffer();
54         if(dialog.getServeurSMTP().getText().equals("smtp.gmail.com"))
55         {connectionUrl.append("pop3s" + "://");}
56         else
57         {connectionUrl.append("pop3" + "://");}
58         connectionUrl.append(dialog.getNomUser().getText() + ":");
59         connectionUrl.append(dialog.getMotPasse() + "@");
60         connectionUrl.append(dialog.getServeurPOP().getText() + "/"");
61         // this.getProfil().setText(dialog.getNomUser().getText());
62
63         // Etablir une session JavaMail et se connecter au serveur.
64         Store store = null;
65         try {
66             // configurer l'objet "Properties" à partir de la boite de dialog.
67             props=new Properties();
68             if(dialog.getServeurSMTP().getText().equals("smtp.gmail.com"))
69             {
70                 props.setProperty("mail.smtp.starttls.enable", "true");
71                 props.setProperty("mail.smtp.socketFactory.port", dialog.getPortSMTP().getText());
72                 props.setProperty("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
73             }
74
75             props.setProperty("mail.smtp.host",dialog.getServeurSMTP().getText());
76             props.setProperty("mail.smtp.port",dialog.getPortSMTP().getText());
77             props.setProperty("mail.smtp.password",dialog.getMotPasse());
78             props.setProperty("mail.smtp.auth", "true");
79
80             // Etablir la session javamail à partir de l'objet props et SMTPAuthenticator
81             session = Session.getDefaultInstance(props,new
SMTPAuthenticator(dialog.getNomUser().getText(),dialog.getMotPasse(),true));
82
83             // Se connecter au serveur e-mail.
84             URLName urln = new URLName(connectionUrl.toString());
85             store = session.getStore(urln);
86             store.connect();
87
88         } catch (Exception ex) {
89             // afficher un message d'erreur.
90
91             Erreur("connexion impossible.", true);
92         }
93
94
95         try {
96             //Recuperer le dossier des messages reçus du serveur
97             folder = store.getFolder("INBOX");
98
99             // Ouvrir le dossier "INBOX" en mode lecture ecriture.
100             folder.open(Folder.READ_WRITE);
101
102             // Recuperer la liste des messages à partir du dossier.
103             messages = folder.getMessage();
104
105             // Affichage des messages reçus et envoyés
106             TaskMessagesReçus task=new TaskMessagesReçus(this);
107             task.execute();
108             afficherMessagesEnvoyés(construireEnvoyes());
109
110         }
111         catch (Exception e) {

```



```

113 // afficher un message d'erreur.
114 Erreur("Impossible De Telecharger Les Messages.", true);
115 }
116
117
118
119 }
120
121
122 // retourne un tableau de messages à partir du dossier contenant les msg envoyés stockés
123 public javax.mail.Message[] construireEnvoyes() throws FileNotFoundException, MessagingException, IOException
124 {File listFiles[] = null;
125 listFiles=listFiles(getfichierElementsEnvoyes());
126 javax.mail.Message msgs[]=new javax.mail.Message[listFiles.length];
127
128 for(int i=0;i<listFiles.length;i++)
129 {
130     InputStream source = new FileInputStream(listFiles[i]);
131     MimeMessage message = new MimeMessage(session, source);
132     msgs[i]=message;
133 }
134 return msgs;
135 }
136
137
138 //retourne les fichiers contenus dans un repertoire donné en parametre
139 public File[] listFiles(String directoryPath){
140 File[] files = null;
141 File directoryToScan = new File(directoryPath);
142 files = directoryToScan.listFiles();
143 return files;
144 }
145
146
147 //Affichage des messages reçus
148 public void afficherMessagesReçus(javax.mail.Message msgs[]) throws Exception
149 {
150     String[] listInfo=new String[3];
151     listeMessageReçus=new ArrayList<Message>(); // listMessage est une liste de panneau
152                                             //où chaque panneau va contenir une en-tete de message
153
154     int j=0;
155     for(int i = msgs.length - 1; i >= 0; i--){
156
157         // Recuperer l'adresse de l'emetteur
158         Address[] senders = msgs[i].getFrom();
159         if (senders != null || senders.length > 0) {
160             listInfo[0]=senders[0].toString();
161         } else {
162             listInfo[0]= "[none]";
163         }
164         // recuperer l'objet du message
165         String subject = msgs[i].getSubject();
166         if (subject != null && subject.length() > 0) {
167             listInfo[1]=subject.toString();
168         } else {
169             listInfo[1]= "[none]";
170         }
171         //Recupere la date
172         Date date = msgs[i].getSentDate();
173         if (date != null) {
174             listInfo[2]=date.toString();
175         } else {
176             listInfo[2]= "[none]";
177         }
178         //Création du JPanel qui va contenir l'entete du message
179         Message msg=new Message(listInfo,i);
180
181         Dimension taille=msg.getPreferredSize();
182         msg.setBounds(0, j*(taille.height+1),taille.width, taille.height);
183         msg.addMouseListener(new java.awt.event.MouseAdapter() {
184
185             public void mouseReleased(java.awt.event.MouseEvent evt){
186                 try {
187                     messageMouseReleased1(evt);
188                 } catch (Exception ex) {
189                     ex.printStackTrace();
190                 }
191             }
192         });
193
194         //inserer le pannel du message dans le panneau des messages reçus
195
196         boiteMessage1.add(msg);
197         taille.setSize(taille.width, (j+1)*(taille.height+2));
198         boiteMessage1.setPreferredSize(taille);
199         listeMessageReçus.add(msg);
200         boiteMessage1.validate();
201         j++;
202     }
203 }
204
205
206 public void afficherMessagesEnvoyés(javax.mail.Message msgs[]) throws Exception
207 {
208     String[] listInfo=new String[3];
209     listeMessageEnvoyés=new ArrayList<Message>();
210
211     int j=0;
212     for(int i = msgs.length - 1; i >= 0; i--){
213
214         Address[] senders = msgs[i].getFrom();
215         // Recuperer l'adresse de l'emetteur
216         if (senders != null || senders.length > 0) {
217             listInfo[0]=senders[0].toString();
218         } else {
219             listInfo[0]= "[none]";
220         }
221         // recuperer l'objet du message
222
223

```



```

225 String subject = msgs[i].getSubject();
226 if (subject != null && subject.length() > 0) {
227     listInfo[1]=subject.toString();
228 } else {
229     listInfo[1]= "[none]";
230 }
231 //Recupere la date
232 Date date = msgs[i].getSentDate();
233 if (date != null) {
234     listInfo[2]=date.toString();
235 } else {
236     listInfo[2]= "[none]";
237 }
238 Message msg=new Message(listInfo,i);
239
240 Dimension taille=msg.getPreferredSize();
241 msg.setBounds(0, j*(taille.height+1),taille.width, taille.height);
242 msg.addMouseListener(new java.awt.event.MouseAdapter() {
243     public void mouseReleased(java.awt.event.MouseEvent evt){
244         try {
245             messageMouseReleased2(evt);
246         } catch (Exception ex) {
247             ex.printStackTrace();
248         }
249     }
250 });
251
252 boiteMessage2.add(msg);
253 taille.setSize(taille.width, (j+1)*(taille.height+2));
254 boiteMessage2.setPreferredSize(taille);
255 listeMessageEnvoyes.add(msg);
256 boiteMessage2.validate();
257 j++;
258 }
259 }
260
261 // crée le nom du dossier des fichiers contenant les msg envoyés et le renvoie
262 public String getfichierElementsEnvoyes() {
263     fichierElementsEnvoyes="C:/"+dialog.getNomUser().getText()+"_"+dialog.getServeurSMTP().getText();
264     File file=new File(fichierElementsEnvoyes);
265     if(!file.exists())
266     {
267         file.mkdir();
268     }
269     return fichierElementsEnvoyes;
270 }
271
272 public String getCarnet() throws IOException {
273
274     carnet="C:/"+dialog.getNomUser().getText()+".txt";
275     File f = new File(carnet);
276
277     try {
278         fW = new BufferedWriter(new FileWriter(f));
279         fR = null;
280     }
281     catch (IOException e) {
282
283     }
284
285
286     return carnet;
287 }
288
289
290
291
292
293 // afficher un message d'erreur et quitter si necessaire.
294 private void Erreur(String message, boolean exit) {
295     JOptionPane.showMessageDialog(this, message, "Erreur",
296         JOptionPane.ERROR_MESSAGE);
297     if (exit)
298         System.exit(0);
299 }
300
301 //Affichege du contenu d'un message reçu au click de l'utilisateur
302 private void messageMouseReleased1(java.awt.event.MouseEvent evt)
303 {
304     Message msgSelectionné= (Message) evt.getSource();
305     try{
306         msgSelectionné.setBackground(Color.gray);
307
308         // recuperer le numero du message selectionné
309         int num=msgSelectionné.getId();
310
311         //Ouvrir la boite de dialogue permettant d'afficher le msg
312         LireMessage dialog= new LireMessage(this,true,messages[num]);
313
314         //Afficher le message
315         dialog.getmetteur().setText(messages[num].getFrom()[0].toString());
316         dialog.getdate().setText(messages[num].getSentDate().toString());
317         dialog.getobjet().setText(messages[num].getSubject().toString());
318         //Afficher les destinataires
319         int j=0;
320         do
321         {
322             dialog.getdests().setText(messages[num].getAllRecipients()[j].toString()+"|");
323             j++;
324         }while (messages[num].getAllRecipients().length<j);
325
326         //récupérer le contenu du message
327         Object content = messages[num].getContent();
328         //Le contenu est un multipart
329         if (content instanceof Multipart) {
330             Multipart multipart = (Multipart) content;
331             //parcourir tous les bodypart du messages
332             for (int x = 0; x < multipart.getCount(); x++) {
333                 BodyPart bodyPart = multipart.getBodyPart(x);

```



```

337 String disposition = bodyPart.getDisposition();
338 if (disposition != null && (disposition.equals(BodyPart.ATTACHMENT)))
339 {
340     //si le bodypart est une pièce jointe on affiche le nom du fichier
341     dialog.getSave().setEnabled(true);
342     DataHandler handler = bodyPart.getDataHandler();
343
344 } //
345
346 if (disposition == null) { // Body uniquement
347
348 // Vérifier si de type plain
349 String contentType=bodyPart.getContentType();
350 if ((contentType.length() >= 10) &&
351
352     (contentType.toLowerCase().substring(
353
354         0, 10).equals("text/plain"))) {
355
356     InputStream is= bodyPart.getInputStream();
357
358
359
360 String msg_body=(new StringUtils()).inputStreamToString(is);
361 dialog.setTextArea(msg_body);
362
363 }
364
365
366     }
367
368 } else {
369     dialog.setTextArea( content.toString());
370
371 }
372
373     dialog.setVisible(true);
374     msgSelectionné.setBackground(Color.white);
375
376     Flags flags=messages[num].getFlags();
377     Flags.Flag[] flag=flags.getSystemFlags();
378     for (int h = 0; h < flag.length; h++) {
379         }
380 }catch(Exception e){
381
382     Erreur("Echec Lors Du Telechargement Du Message",false);
383     msgSelectionné.setBackground(Color.white);
384 }
385
386 }
387
388 //Affichege du contenu d'un message reçu au click de l'utilisateur
389 private void messageMouseReleased2(java.awt.event.MouseEvent evt)
390
391 {
392     Message msgSelectionné=(Message)evt.getSource();
393     try{
394
395     msgSelectionné.setBackground(Color.gray);
396
397     // recuperer le numero du message selectionné
398     int num=msgSelectionné.getId();
399
400
401     javax.mail.Message messages2[]=construireEnvoyes();
402
403     //Ouvrir la boite de dialogue permettant d'afficher le msg
404     LireMessage dialog= new LireMessage(this,true,messages2[num]);
405
406     dialog.getmetteur().setText(messages2[num].getFrom()[0].toString());
407     dialog.getdate().setText(messages2[num].getSentDate().toString());
408     dialog.getobjet().setText(messages2[num].getSubject().toString());
409     int j=0;
410     do
411     {
412         dialog.getdests().setText(messages2[num].getAllRecipients()[j].toString()+"|");
413         j++;
414     }while (messages2[num].getAllRecipients().length<j);
415
416     //récupérer le contenu du message
417     Object content = messages2[ num].getContent();
418     //Le contenu est un multipart
419     if (content instanceof Multipart) {
420         Multipart multipart = (Multipart) content;
421         //parcourir tous les bodypart du messages
422         for (int x = 0; x < multipart.getCount(); x++) {
423             BodyPart bodyPart = multipart.getBodyPart(x);
424             String disposition = bodyPart.getDisposition();
425             if (disposition != null && (disposition.equals(BodyPart.ATTACHMENT)))
426             {
427                 //si le bodypart est une pièce jointe on affiche le nom du fichier
428                 dialog.getSave().setEnabled(true);
429                 DataHandler handler = bodyPart.getDataHandler();
430
431
432             } //
433
434             if (disposition == null) { // Body uniquement
435
436 // Vérifier si de type plain
437 String contentType=bodyPart.getContentType();
438 if ((contentType.length() >= 10) &&
439
440     (contentType.toLowerCase().substring(
441
442         0, 10).equals("text/plain"))) {
443
444     InputStream is= bodyPart.getInputStream();
445
446
447

```



```

449     String msg_body=(new StringUtils()).inputStreamToString(is);
450     dialog.setJTextArea(msg_body);
451 }
452
453     }
454 }
455
456 } else {
457     dialog.setJTextArea( content.toString());
458 }
459
460
461     dialog.setVisible(true);
462     msgSelectionné.setBackground(Color.white);
463
464 }catch(Exception e){
465     Erreur("Echec Lors Du Telechargement Du Message",false);
466     msgSelectionné.setBackground(Color.white);
467 }
468 }
469
470 @SuppressWarnings("unchecked")
471 // <editor-fold defaultstate="collapsed" desc="Generated Code">///GEN-BEGIN:initComponents
472 private void initComponents() {
473
474     jPanel2 = new javax.swing.JPanel();
475     jLabel1 = new javax.swing.JLabel();
476     Répondre = new javax.swing.JButton();
477     Transferer = new javax.swing.JButton();
478     Supprimer = new javax.swing.JButton();
479     jButton2 = new javax.swing.JButton();
480     Déconnexion = new javax.swing.JButton();
481     jPanel1 = new javax.swing.JPanel();
482     jPanel3 = new javax.swing.JPanel();
483     jPanel5 = new javax.swing.JPanel();
484     jButton1 = new javax.swing.JButton();
485     Nouveau = new javax.swing.JButton();
486     Carnet = new javax.swing.JButton();
487     dateChooserPanel1 = new datechooser.beans.DateChooserPanel();
488     jTabbedPane = new javax.swing.JTabbedPane();
489     Recu = new javax.swing.JPanel();
490     jScrollPane2 = new javax.swing.JScrollPane();
491     boîteMessage1 = new javax.swing.JPanel();
492     jPanel4 = new javax.swing.JPanel();
493     from = new javax.swing.JLabel();
494     object = new javax.swing.JLabel();
495     date = new javax.swing.JLabel();
496     piece = new javax.swing.JLabel();
497     Envoye = new javax.swing.JPanel();
498     jScrollPane1 = new javax.swing.JScrollPane();
499     boîteMessage2 = new javax.swing.JPanel();
500     jPanel6 = new javax.swing.JPanel();
501     from1 = new javax.swing.JLabel();
502     object1 = new javax.swing.JLabel();
503     date1 = new javax.swing.JLabel();
504     piece1 = new javax.swing.JLabel();
505
506     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
507     setTitle("UMail Accueil");
508     setBackground(new java.awt.Color(255, 255, 255));
509     setMinimumSize(new java.awt.Dimension(300, 157));
510     setName("accueil"); // NOI18N
511     setResizable(false);
512
513     jPanel2.setBackground(new java.awt.Color(255, 255, 255));
514     jPanel2.setLayout(new java.awt.GridLayout(1, 0));
515
516     jLabel1.setForeground(new java.awt.Color(255, 255, 255));
517     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/logoUmail.png"))); // NOI18N
518     jPanel2.add(jLabel1);
519
520     Répondre.setBackground(new java.awt.Color(0, 102, 102));
521     Répondre.setFont(new java.awt.Font("Garamond", 1, 18));
522     Répondre.setForeground(new java.awt.Color(255, 255, 255));
523     Répondre.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Répondre.png"))); // NOI18N
524     Répondre.setText("Répondre");
525     Répondre.addActionListener(new java.awt.event.ActionListener() {
526         public void actionPerformed(java.awt.event.ActionEvent evt) {
527             RépondreActionPerformed(evt);
528         }
529     });
530     jPanel2.add(Répondre);
531
532     Transferer.setBackground(new java.awt.Color(0, 102, 102));
533     Transferer.setFont(new java.awt.Font("Garamond", 1, 18));
534     Transferer.setForeground(new java.awt.Color(255, 255, 255));
535     Transferer.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Transférer.png"))); // NOI18N
536     Transferer.setText("Transférer");
537     Transferer.addActionListener(new java.awt.event.ActionListener() {
538         public void actionPerformed(java.awt.event.ActionEvent evt) {
539             TransfererActionPerformed(evt);
540         }
541     });
542     jPanel2.add(Transferer);
543
544     Supprimer.setBackground(new java.awt.Color(0, 102, 102));
545     Supprimer.setFont(new java.awt.Font("Garamond", 1, 18));
546     Supprimer.setForeground(new java.awt.Color(255, 255, 255));
547     Supprimer.setIcon(new javax.swing.ImageIcon(getClass().getResource("/corbeille.png"))); // NOI18N
548     Supprimer.setText("Supprimer");
549     Supprimer.addActionListener(new java.awt.event.ActionListener() {
550         public void actionPerformed(java.awt.event.ActionEvent evt) {
551             SupprimerActionPerformed(evt);
552         }
553     });
554     jPanel2.add(Supprimer);
555
556     jButton2.setBackground(new java.awt.Color(0, 102, 102));
557     jButton2.setFont(new java.awt.Font("Garamond", 1, 18)); // NOI18N
558     jButton2.setForeground(new java.awt.Color(255, 255, 255));
559     jButton2.setIcon(new

```



```

560 javax.swing.ImageIcon(getClass().getResource("/aide-point-interrogation-icone-5530-48.png")); // NOI18N
561 jButton2.setText("Aide");
562 jButton2.addActionListener(new java.awt.event.ActionListener() {
563     public void actionPerformed(java.awt.event.ActionEvent evt) {
564         jButton2ActionPerformed(evt);
565     }
566 });
567 jPanel12.add(jButton2);
568 Déconnexion.setBackground(new java.awt.Color(0, 102, 102));
569 Déconnexion.setFont(new java.awt.Font("Garamond", 1, 18));
570 Déconnexion.setForeground(new java.awt.Color(255, 255, 255));
571 Déconnexion.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Exit.png"))); // NOI18N
572 Déconnexion.setText("Déconnexion");
573 Déconnexion.addActionListener(new java.awt.event.ActionListener() {
574     public void actionPerformed(java.awt.event.ActionEvent evt) {
575         DéconnexionActionPerformed(evt);
576     }
577 });
578 jPanel12.add(Déconnexion);
579
580 getContentPane().add(jPanel12, java.awt.BorderLayout.PAGE_START);
581
582 jPanel11.setBackground(new java.awt.Color(255, 255, 255));
583 jPanel11.setLayout(new java.awt.BorderLayout());
584
585 jPanel13.setBackground(new java.awt.Color(255, 255, 255));
586
587 jPanel15.setLayout(new java.awt.GridLayout(3, 1));
588
589 jButton1.setBackground(new java.awt.Color(0, 102, 102));
590 jButton1.setFont(new java.awt.Font("Garamond", 1, 18));
591 jButton1.setForeground(new java.awt.Color(255, 255, 255));
592 jButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/dossier-mises-a-jour-icone-6204-32.png"))); // NOI18N
593 jButton1.setText("Rafraîchir");
594 jButton1.addActionListener(new java.awt.event.ActionListener() {
595     public void actionPerformed(java.awt.event.ActionEvent evt) {
596         jButton1ActionPerformed(evt);
597     }
598 });
599 jPanel15.add(jButton1);
600
601 Nouveau.setBackground(new java.awt.Color(0, 102, 102));
602 Nouveau.setFont(new java.awt.Font("Garamond", 1, 18));
603 Nouveau.setForeground(new java.awt.Color(255, 255, 255));
604 Nouveau.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Mail.png"))); // NOI18N
605 Nouveau.setText("Nouveau");
606 Nouveau.setPreferredSize(new java.awt.Dimension(60, 60));
607 Nouveau.addActionListener(new java.awt.event.ActionListener() {
608     public void actionPerformed(java.awt.event.ActionEvent evt) {
609         NouveauActionPerformed(evt);
610     }
611 });
612 jPanel15.add(Nouveau);
613
614 Carnet.setBackground(new java.awt.Color(0, 102, 102));
615 Carnet.setFont(new java.awt.Font("Garamond", 1, 18));
616 Carnet.setForeground(new java.awt.Color(255, 255, 255));
617 Carnet.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Carnet.png"))); // NOI18N
618 Carnet.setText("Carnet");
619 Carnet.addActionListener(new java.awt.event.ActionListener() {
620     public void actionPerformed(java.awt.event.ActionEvent evt) {
621         CarnetActionPerformed(evt);
622     }
623 });
624 jPanel15.add(Carnet);
625
626 dateChooserPanel1.setCurrentView(new datechooser.view.appearance.AppearancesList("Umail",
627     new datechooser.view.appearance.ViewAppearance("umail",
628         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
629             new java.awt.Color(0, 0, 0),
630             new java.awt.Color(0, 0, 255),
631             false,
632             true,
633             new datechooser.view.appearance.swing.ButtonPainter()),
634         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
635             new java.awt.Color(0, 0, 0),
636             new java.awt.Color(0, 0, 255),
637             true,
638             true,
639             new datechooser.view.appearance.swing.ButtonPainter()),
640         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
641             new java.awt.Color(0, 0, 255),
642             new java.awt.Color(0, 0, 255),
643             false,
644             true,
645             new datechooser.view.appearance.swing.ButtonPainter()),
646         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
647             new java.awt.Color(128, 128, 128),
648             new java.awt.Color(0, 0, 255),
649             false,
650             true,
651             new datechooser.view.appearance.swing.LabelPainter()),
652         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
653             new java.awt.Color(0, 0, 0),
654             new java.awt.Color(0, 0, 255),
655             false,
656             true,
657             new datechooser.view.appearance.swing.LabelPainter()),
658         new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
659             new java.awt.Color(0, 0, 0),
660             new java.awt.Color(255, 0, 0),
661             false,
662             false,
663             new datechooser.view.appearance.swing.ButtonPainter()),
664         (datechooser.view.BackRenderer)null,
665         false,
666         true),
667     new datechooser.view.appearance.ViewAppearance("Umail",
668         new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(255, 0, 0),
669             new java.awt.Color(0, 51, 51),
670             (javax.swing.border.Border)null,

```



```

672 new java.awt.Font("Arial", java.awt.Font.BOLD, 12),
673 new java.awt.Color(0, 0, 153),
674 0.0f),
675 new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(0, 102, 102),
676 new java.awt.Color(141, 134, 42),
677 (javax.swing.border.Border)null,
678 new java.awt.Font("Serif", java.awt.Font.BOLD, 14),
679 new java.awt.Color(0, 0, 102),
680 0.4f),
681 new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(44, 153, 143),
682 new java.awt.Color(51, 51, 51),
683 (javax.swing.border.Border)null,
684 new java.awt.Font("Arial", java.awt.Font.BOLD, 12),
685 new java.awt.Color(0, 0, 153),
686 0.5f),
687 new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(255, 255, 255),
688 new java.awt.Color(0, 102, 102),
689 (javax.swing.border.Border)null,
690 new java.awt.Font("Serif", java.awt.Font.BOLD + java.awt.Font.ITALIC, 10),
691 new java.awt.Color(0, 0, 255),
692 0.0f),
693 new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(0, 0, 0),
694 new java.awt.Color(255, 255, 255),
695 (javax.swing.border.Border)null,
696 new java.awt.Font("Serif", java.awt.Font.BOLD, 12),
697 new java.awt.Color(0, 0, 255),
698 0.4f),
699 new datechooser.view.appearance.custom.CustomCellAppearance(new java.awt.Color(255, 255, 255),
700 new java.awt.Color(255, 0, 0),
701 (javax.swing.border.Border)null,
702 new java.awt.Font("Serif", java.awt.Font.PLAIN, 12),
703 new java.awt.Color(255, 0, 0),
704 0.0f),
705 new datechooser.view.BackRenderer( 2,
706 "/C:/Users/Nahla/Documents/UMail fella/src/test 2.png"),
707 true,
708 true),
709 new datechooser.view.appearance.ViewAppearance("custom",
710 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
711 new java.awt.Color(0, 0, 0),
712 new java.awt.Color(0, 0, 255),
713 false,
714 true,
715 new datechooser.view.appearance.swing.ButtonPainter()),
716 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
717 new java.awt.Color(0, 0, 0),
718 new java.awt.Color(0, 0, 255),
719 true,
720 true,
721 new datechooser.view.appearance.swing.ButtonPainter()),
722 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
723 new java.awt.Color(0, 0, 255),
724 new java.awt.Color(0, 0, 255),
725 false,
726 true,
727 new datechooser.view.appearance.swing.ButtonPainter()),
728 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
729 new java.awt.Color(128, 128, 128),
730 new java.awt.Color(0, 0, 255),
731 false,
732 true,
733 new datechooser.view.appearance.swing.LabelPainter()),
734 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
735 new java.awt.Color(0, 0, 0),
736 new java.awt.Color(0, 0, 255),
737 false,
738 true,
739 new datechooser.view.appearance.swing.LabelPainter()),
740 new datechooser.view.appearance.swing.SwingCellAppearance(new java.awt.Font("Tahoma", java.awt.Font.PLAIN, 11),
741 new java.awt.Color(0, 0, 0),
742 new java.awt.Color(255, 0, 0),
743 false,
744 false,
745 new datechooser.view.appearance.swing.ButtonPainter()),
746 (datechooser.view.BackRenderer)null,
747 false,
748 true));
749 dateChooserPanel1.setCalendarBackground(new java.awt.Color(255, 255, 255));
750 dateChooserPanel1.setLocked(true);
751
752 javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
753 jPanel3.setLayout(jPanel3Layout);
754 jPanel3Layout.setHorizontalGroup(
755     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
756     .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, 195, Short.MAX_VALUE)
757     .addComponent(dateChooserPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 195, Short.MAX_VALUE)
758 );
759 jPanel3Layout.setVerticalGroup(
760     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
761     .addGroup(jPanel3Layout.createSequentialGroup()
762         .addGap(44, 44, 44)
763         .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
764             javax.swing.GroupLayout.PREFERRED_SIZE)
765         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
766         .addComponent(dateChooserPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 313, javax.swing.GroupLayout.PREFERRED_SIZE)
767         .addContainerGap(263, Short.MAX_VALUE))
768 );
769
770 jPanel1.add(jPanel3, java.awt.BorderLayout.LINE_START);
771
772 JTabbedPane1.setBackground(new java.awt.Color(234, 224, 190));
773 JTabbedPane1.setFont(new java.awt.Font("Garamond", 1, 14));
774
775 Recu.setLayout(new java.awt.BorderLayout());
776
777 JScrollPane2.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLB
778
779 boiteMessage1.setBackground(new java.awt.Color(255, 255, 255));
780
781 javax.swing.GroupLayout boiteMessage1Layout = new javax.swing.GroupLayout(boiteMessage1);
782 boiteMessage1.setLayout(boiteMessage1Layout);
783 boiteMessage1Layout.setHorizontalGroup(

```




```

895     .addContainerGap();
896
897     Envoye.add(jPanel6, java.awt.BorderLayout.PAGE_START);
898
899     jTabbedPane1.addTab("Messages Envoyés ", Envoye);
900
901     jPanel1.add(jTabbedPane1, java.awt.BorderLayout.CENTER);
902
903     getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
904
905     pack();
906     } // </editor-fold> // GEN-END: initComponents
907
908     private void DéconnexionActionPerformed(java.awt.event.ActionEvent evt) { // GEN-FIRST: event DéconnexionActionPerformed
909     try {
910         folder.close(true);
911         System.exit(0);
912     } catch (Exception e) {
913         Erreur("Folder not closed!!", false);
914     }
915
916     // TODO add your handling code here: // GEN-LAST: event DéconnexionActionPerformed
917     }
918
919     private void NouveauActionPerformed(java.awt.event.ActionEvent evt) { // GEN-FIRST: event NouveauActionPerformed
920
921     editeur = new SendEmail(this, true); // TODO add your handling code here: // GEN-LAST: event NouveauActionPerformed
922     }
923
924     // transférer un message
925     private void TransfererActionPerformed(java.awt.event.ActionEvent evt) { // GEN-FIRST: event TransfererActionPerformed
926     if (jTabbedPane1.getSelectedComponent().equals(Recu)) {
927         int i = 0; boolean selected = false;
928         while (i < listeMessageReçus.size() && !selected)
929         {
930             if (listeMessageReçus.get(i).getSelectionner().isSelected()) {
931                 selected = true;
932             }
933             i++;
934         }
935         if (selected == false) { Erreur("Aucun Message Sélectionné", false); }
936         else {
937             int j = listeMessageReçus.get(i - 1).getId();
938             editeur = new SendEmail(this, true);
939             Object content;
940
941             try {
942                 content = messages[j].getContent();
943                 if (content instanceof Multipart) {
944                     Multipart multipart = (Multipart) content;
945                     // parcourir tous les bodypart du messages
946                     for (int x = 0; x < multipart.getCount(); x++) {
947                         BodyPart bodyPart = multipart.getBodyPart(x);
948                         String disposition = bodyPart.getDisposition();
949                         if (disposition != null && (disposition.equals(BodyPart.ATTACHMENT)))
950                         {
951                             //
952                         }
953                     }
954
955                     if (disposition == null) { // Body uniquement
956
957                         // Vérifier si de type plain
958                         String contentType = bodyPart.getContentType();
959                         if ((contentType.length() >= 10) &&
960                             (contentType.toLowerCase().substring(
961                                 0, 10).equals("text/plain"))) {
962
963                             InputStream is = bodyPart.getInputStream();
964
965                             String msg_body = (new StringUtils()).inputStreamToString(is);
966                             editeur.getHTMLEditor().setHTMLContent(msg_body);
967
968                             //
969                         }
970                     }
971                 } else {
972                     editeur.getHTMLEditor().setHTMLContent(content.toString());
973                 }
974             } catch (IOException ex) {
975             } catch (MessagingException ex) {
976             }
977
978             // TODO add your handling code here: // GEN-LAST: event TransfererActionPerformed
979         }
980     }
981
982     // supprimer le messages sélectionné par l'utilisateur
983     private void SupprimerActionPerformed(java.awt.event.ActionEvent evt) { // GEN-FIRST: event SupprimerActionPerformed
984     // si l'onglet des messages reçus est actif
985     if (jTabbedPane1.getSelectedComponent().equals(Recu)) {
986         boolean select = false;
987         for (int i = 0; i < listeMessageReçus.size(); i++) {
988             if (listeMessageReçus.get(i).getSelectionner().isSelected()) {
989                 supprimerMessageReçu(i); select = true;
990             }
991         }
992     }
993     }
994
995     //
996
997     //
998
999     //
1000
1001     //
1002
1003     //
1004
1005

```



```

1007 // si aucun message n'est sélectionné afficher un msg d'erreur
1008 if(select==false){Erreur("Aucun Message Sélectionné",false);}
1009 }else { // si l'onglet des messages envoyés est actif
1010 if(jTabbedPanel.getSelectedComponent().equals(Envoye)){
1011     boolean select2=false;
1012     for(int i=0;i<listeMessageEnvoyés.size();i++){
1013         if(listeMessageEnvoyés.get(i).getSelectionner().isSelected()){
1014             supprimerMessageEnvoyé(i);select2=true;
1015         }
1016     }
1017     // si aucun message n'est sélectionné afficher un msg d'erreur
1018     if(select2==false){Erreur("Aucun Message Sélectionné",false);}
1019 }
1020
1021 ///GEN-LAST:event SupprimerActionPerformed
1022 //pour supprimer un message reçu
1023 public void supprimerMessageReçu(int i){
1024
1025     try{
1026         //attribuer au message sélectionné un flag "Flags.Flag.DELETED"
1027         messages[messages.length-i-1].setFlag(Flags.Flag.DELETED,true);
1028         //Supprimer le messages du panneau des messages reçus
1029         listeMessageReçus.remove(i);
1030         //réafficher le panneau
1031         Iterator<Message> it =listeMessageReçus.iterator();
1032         boiteMessage1.removeAll();
1033         int j=0;
1034         while(it.hasNext()){
1035             Message msg=it.next();
1036             Dimension taille=msg.getPreferredSize();
1037             msg.setBounds(0, j*(taille.height+1),taille.width, taille.height);
1038             boiteMessage1.add(msg);
1039             j++;
1040         }
1041         boiteMessage1.setVisible(false);
1042         boiteMessage1.setVisible(true);
1043         boiteMessage1.validate();
1044     }
1045     catch (Exception e){
1046         Erreur("Echec De La Suppression",false);
1047     }
1048 }
1049
1050 //pour supprimer un messages envoyé
1051 public void supprimerMessageEnvoyé(int i){
1052     //recuperer un tableau de fichier à partir du dossier des messages envoyés
1053     File[] listFiles=listFiles(fichierElementsEnvoyes);
1054     //supprimer le fichier du messages sélectionné
1055     listFiles[i].delete();
1056     //supprimer le messages du panneau des msg envoyés
1057     listeMessageEnvoyés.remove(i);
1058     // réafficher le panneau
1059     Iterator<Message> it =listeMessageEnvoyés.iterator();
1060     boiteMessage2.removeAll();
1061     int j=0;
1062     while(it.hasNext()){
1063         Message msg=it.next();
1064         Dimension taille=msg.getPreferredSize();
1065         msg.setBounds(0, j*(taille.height+1),taille.width, taille.height);
1066         boiteMessage2.add(msg);
1067         j++;
1068     }
1069     boiteMessage2.setVisible(false);
1070     boiteMessage2.setVisible(true);
1071     boiteMessage2.validate();
1072 }
1073
1074
1075 private void CarnetActionPerformed(java.awt.event.ActionEvent evt) {///GEN-FIRST:event CarnetActionPerformed
1076 // TODO add your handling code here:
1077 Carnet_View carnet;
1078
1079     carnet = new Carnet_View(this,true);
1080     carnet.setVisible(true);
1081
1082
1083
1084 }
1085 ///GEN-LAST:event_CarnetActionPerformed
1086
1087 private void RépondreActionPerformed(java.awt.event.ActionEvent evt) {///GEN-FIRST:event_RépondreActionPerformed
1088
1089     if(jTabbedPanel.getSelectedComponent().equals(Recu)) {
1090         int i=0;boolean selected =false;
1091         while(i<listeMessageReçus.size() && !selected)
1092         {
1093             if(listeMessageReçus.get(i).getSelectionner().isSelected()){
1094                 selected=true;
1095             }
1096             i++;
1097         }
1098         if(selected==false){Erreur("Aucun Message Sélectionné",false);}
1099         else{
1100
1101             editeur= new SendEmail(this, true);
1102             int j = listeMessageReçus.get(i-1).getId();
1103             try {
1104                 editeur.setdestinataire(messages[j].getFrom()[0].toString());
1105             } catch (MessagingException ex) {
1106                 Logger.getLogger(Accueil.class.getName()).log(Level.SEVERE, null, ex);
1107             }
1108             editeur.setVisible(true);
1109         }
1110     }
1111 }
1112 ///GEN-LAST:event RépondreActionPerformed
1113
1114 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {///GEN-FIRST:event_jButton1ActionPerformed
1115     try{
1116         folder.close(true);
1117     }catch (Exception e){

```



```

1119     }
1120
1121     this.connect(false);
1122     this.setVisible(true);
1123     }//GEN-LAST:event_jButton1ActionPerformed
1124
1125 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
1126 // TODO add your handling code here:
1127     try {
1128
1129         File chmFile = new File("Aide.chm");
1130         if (chmFile.exists()) {
1131
1132             if (Desktop.isDesktopSupported()) {
1133                 Desktop.getDesktop().open(chmFile);
1134             } else {
1135
1136             }
1137
1138         } else {
1139
1140         }
1141
1142     } catch (Exception ex) {
1143
1144     }//GEN-LAST:event_jButton2ActionPerformed
1145
1146
1147 public Session getSession() {
1148     return session;
1149 }
1150
1151 public Authentication getDialog() {
1152     return dialog;
1153 }
1154
1155
1156
1157
1158
1159 public static void main(String args[]) {
1160     /* Set the Nimbus look and feel */
1161     //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
1162     /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
1163      * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
1164      */
1165     try {
1166         for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
1167             if ("Nimbus".equals(info.getName())) {
1168                 javax.swing.UIManager.setLookAndFeel(info.getClassName());
1169                 break;
1170             }
1171         }
1172     } catch (ClassNotFoundException ex) {
1173         java.util.logging.Logger.getLogger(Accueil.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
1174     } catch (InstantiationException ex) {
1175         java.util.logging.Logger.getLogger(Accueil.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
1176     } catch (IllegalAccessException ex) {
1177         java.util.logging.Logger.getLogger(Accueil.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
1178     } catch (javax.swing.UnsupportedLookAndFeelException ex) {
1179         java.util.logging.Logger.getLogger(Accueil.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
1180     }
1181     //</editor-fold>
1182     try {
1183         UIManager.setLookAndFeel("com.jtattoo.plaf.acryl.AcrylLookAndFeel");
1184
1185     } catch (ClassNotFoundException e) {
1186         // TODO Auto-generated catch block
1187         e.printStackTrace();
1188     } catch (InstantiationException e) {
1189         // TODO Auto-generated catch block
1190         e.printStackTrace();
1191     } catch (IllegalAccessException e) {
1192         // TODO Auto-generated catch block
1193         e.printStackTrace();
1194     } catch (UnsupportedLookAndFeelException e) {
1195         // TODO Auto-generated catch block
1196         e.printStackTrace();
1197     }
1198
1199     /* Create and display the form */
1200     java.awt.EventQueue.invokeLater(new Runnable() {
1201
1202         public void run() {
1203             Accueil client = new Accueil();
1204             client.connect(true);
1205             client.setVisible(true);
1206
1207         }
1208     });
1209 }
1210
1211
1212
1213 public javax.mail.Message[] getMessages() {
1214     return messages;
1215 }
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227 private javax.mail.Message[] messages;
1228 private Folder folder;

```



```

1230     private Session session;
1231     private Properties props;
1232     private Authentication dialog;
1233     private SendEmail editeur;
1234     private String fichierElementsEnvoyes;
1235     private String fichierMessagesReçus;
1236     private String carnet;
1237     private RoundButton rButton;
1238     private ArrayList<Message> listeMessageReçus;
1239     private ArrayList<Message> listeMessageEnvoyés;
1240     private BufferedWriter fW;
1241     private BufferedReader fR;
1242     ImageIcon iconPhoto = new ImageIcon("piece-jointe-16.png");
1243     java.awt.Image uneImage= iconPhoto.getImage();
1244
1245     // Variables declaration - do not modify//GEN-BEGIN:variables
1246     private javax.swing.JButton Carnet;
1247     private javax.swing.JButton Déconnexion;
1248     private javax.swing.JPanel Envoye;
1249     private javax.swing.JButton Nouveau;
1250     private javax.swing.JPanel Recu;
1251     private javax.swing.JButton Répondre;
1252     private javax.swing.JButton Supprimer;
1253     private javax.swing.JButton Transferer;
1254     private javax.swing.JPanel boiteMessage1;
1255     private javax.swing.JPanel boiteMessage2;
1256     private javax.swing.JLabel date;
1257     private javax.swing.JLabel date1;
1258     private datechooser.beans.DateChooserPanel dateChooserPanel1;
1259     private javax.swing.JLabel from;
1260     private javax.swing.JLabel from1;
1261     private javax.swing.JButton jButton1;
1262     private javax.swing.JButton jButton2;
1263     private javax.swing.JLabel jLabel1;
1264     private javax.swing.JPanel jPanel1;
1265     private javax.swing.JPanel jPanel2;
1266     private javax.swing.JPanel jPanel3;
1267     private javax.swing.JPanel jPanel4;
1268     private javax.swing.JPanel jPanel5;
1269     private javax.swing.JPanel jPanel6;
1270     private javax.swing.JScrollPane jScrollPane1;
1271     private javax.swing.JScrollPane jScrollPane2;
1272     private javax.swing.JTabbedPane jTabbedPane1;
1273     private javax.swing.JLabel object;
1274     private javax.swing.JLabel object1;
1275     private javax.swing.JLabel piece;
1276     private javax.swing.JLabel piece1;
1277     // End of variables declaration//GEN-END:variables
1278
1279 }
1280
1281

```

