|  |
| --- |
|  |
| Laboratoire de Technologies de e-commerce |
| Partie “Technologie Java Standard” évaluation 2 |

|  |
| --- |
| Seel Océane, Fink Jérôme |

Table des matières

**Code d’applic\_mail2**

Connexion2

Envoi de messages5

Affichage message + pièces jointes10

**Test de conformité18**

**Test d’homogénéité20**

**Corrélation22**

Code d’Applic\_mail

Connexion :

1. **package** applic\_mail;
3. **import** javax.mail.MessagingException;
4. **import** javax.mail.NoSuchProviderException;
5. **import** javax.mail.Store;
6. **import** javax.swing.SwingUtilities;
8. /\*\*
9. \*
10. \* @author John
11. \*/
12. **public** **class** connexionPanel **extends** javax.swing.JPanel {
14. /\*\*
15. \* Creates new form connexionPanel
16. \*/
17. **public** connexionPanel() {
18. initComponents();
19. errorLabel.setVisible(**false**);
20. }
22. /\*\*
23. \* This method is called from within the constructor to initialize the form.
24. \* WARNING: Do NOT modify this code. The content of this method is always
25. \* regenerated by the Form Editor.
26. \*/
27. @SuppressWarnings("unchecked")
28. // <editor-fold defaultstate="collapsed" desc="Generated Code">
29. **private** **void** initComponents() {
31. titreLabel = **new** javax.swing.JLabel();
32. loginLabel = **new** javax.swing.JLabel();
33. loginTextField = **new** javax.swing.JTextField();
34. passwordLabel = **new** javax.swing.JLabel();
35. passwordField = **new** javax.swing.JPasswordField();
36. errorLabel = **new** javax.swing.JLabel();
37. connexionButton = **new** javax.swing.JButton();
39. titreLabel.setFont(**new** java.awt.Font("Tahoma", 0, 24)); // NOI18N
40. titreLabel.setText("Messagerie connexion");
42. loginLabel.setFont(**new** java.awt.Font("Tahoma", 0, 14)); // NOI18N
43. loginLabel.setText("Login : ");
45. passwordLabel.setFont(**new** java.awt.Font("Tahoma", 0, 14)); // NOI18N
46. passwordLabel.setText("Mot de passe : ");
48. errorLabel.setForeground(**new** java.awt.Color(255, 0, 0));
49. errorLabel.setText("jLabel1");
51. connexionButton.setText("Connexion");
52. connexionButton.addActionListener(**new** java.awt.event.ActionListener() {
53. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
54. connexionButtonActionPerformed(evt);
55. }
56. });
58. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(**this**);
59. **this**.setLayout(layout);
60. layout.setHorizontalGroup(
61. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
62. .addGroup(layout.createSequentialGroup()
63. .addGap(189, 189, 189)
64. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
65. .addGroup(layout.createSequentialGroup()
66. .addComponent(loginLabel)
67. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
68. .addComponent(loginTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, 199, javax.swing.GroupLayout.PREFERRED\_SIZE))
69. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
70. .addComponent(errorLabel)
71. .addGroup(layout.createSequentialGroup()
72. .addGap(62, 62, 62)
73. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
74. .addComponent(titreLabel)
75. .addComponent(connexionButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 232, javax.swing.GroupLayout.PREFERRED\_SIZE)))
76. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
77. .addComponent(passwordLabel)
78. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
79. .addComponent(passwordField, javax.swing.GroupLayout.PREFERRED\_SIZE, 199, javax.swing.GroupLayout.PREFERRED\_SIZE))))
80. .addContainerGap(215, Short.MAX\_VALUE))
81. );
82. layout.setVerticalGroup(
83. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
84. .addGroup(layout.createSequentialGroup()
85. .addContainerGap()
86. .addComponent(titreLabel)
87. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
88. .addGroup(layout.createSequentialGroup()
89. .addGap(71, 71, 71)
90. .addComponent(loginLabel))
91. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
92. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
93. .addComponent(loginTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))
94. .addGap(18, 18, 18)
95. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
96. .addComponent(passwordField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
97. .addComponent(passwordLabel))
98. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
99. .addComponent(connexionButton)
100. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
101. .addComponent(errorLabel)
102. .addContainerGap(125, Short.MAX\_VALUE))
103. );
104. }// </editor-fold>
106. **private** **void** connexionButtonActionPerformed(java.awt.event.ActionEvent evt) {
108. errorLabel.setVisible(**false**);
109. **if**(loginTextField.getText().isEmpty())
110. {
111. errorLabel.setText("Vous devez rentrer un login");
112. errorLabel.setVisible(**true**);
113. **return**;
114. }
116. **if**(passwordField.getText().isEmpty())
117. {
118. errorLabel.setText("Vous devez rentrer un mot de passe");
119. errorLabel.setVisible(**true**);
120. **return**;
121. }
123. GUI\_Mail container = (GUI\_Mail)SwingUtilities.getWindowAncestor(**this**);
124. Store st = **null**;
125. **try** {
126. st = container.getSession().getStore("pop3");
127. st.connect(container.getHost(), loginTextField.getText(), passwordField.getText());
128. container.setStore(st);
129. }
130. **catch** (NoSuchProviderException ex)
131. {
132. errorLabel.setText("Impossible de se connecter");
133. errorLabel.setVisible(**true**);
134. ex.printStackTrace();
135. **return**;
136. }
137. **catch** (MessagingException ex)
138. {
139. ex.printStackTrace();
140. **return**;
141. }
143. container.setUser(loginTextField.getText());
144. loginTextField.setText("");
145. passwordField.setText("");
147. container.changeLayout("inbox");
148. }

Envoi de messages :

1. **public** **class** sendMessagePanel **extends** javax.swing.JPanel {
3. **private** MimeBodyPart pieceJointe;
5. /\*\*
6. \* Creates new form sendMessagePanel
7. \*/
8. **public** sendMessagePanel() {
9. initComponents();
10. errorLabel.setVisible(**false**);
11. okLabel.setVisible(**false**);
12. pieceJointeLabel.setVisible(**false**);
13. }
15. /\*\*
16. \* This method is called from within the constructor to initialize the form.
17. \* WARNING: Do NOT modify this code. The content of this method is always
18. \* regenerated by the Form Editor.
19. \*/
20. @SuppressWarnings("unchecked")
21. // <editor-fold defaultstate="collapsed" desc="Generated Code">
22. **private** **void** initComponents() {
24. jScrollPane1 = **new** javax.swing.JScrollPane();
25. messageTextArea = **new** javax.swing.JTextArea();
26. sujetTextField = **new** javax.swing.JTextField();
27. sujetLabel = **new** javax.swing.JLabel();
28. envoyerButton = **new** javax.swing.JButton();
29. destinataireTextField = **new** javax.swing.JTextField();
30. destinataireLabel = **new** javax.swing.JLabel();
31. errorLabel = **new** javax.swing.JLabel();
32. inboxButton = **new** javax.swing.JButton();
33. okLabel = **new** javax.swing.JLabel();
34. ajouterButton = **new** javax.swing.JButton();
35. pieceJointeLabel = **new** javax.swing.JLabel();
37. messageTextArea.setColumns(20);
38. messageTextArea.setRows(5);
39. jScrollPane1.setViewportView(messageTextArea);
41. sujetLabel.setText("Sujet : ");
43. envoyerButton.setText("Envoyer");
44. envoyerButton.addActionListener(**new** java.awt.event.ActionListener() {
45. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
46. envoyerButtonActionPerformed(evt);
47. }
48. });
50. destinataireLabel.setText("Destinataire : ");
52. errorLabel.setForeground(**new** java.awt.Color(255, 0, 51));
53. errorLabel.setText("jLabel1");
55. inboxButton.setText("Inbox");
56. inboxButton.addActionListener(**new** java.awt.event.ActionListener() {
57. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
58. inboxButtonActionPerformed(evt);
59. }
60. });
62. okLabel.setForeground(**new** java.awt.Color(51, 204, 0));
63. okLabel.setText("Le mail est envoyÃ©");
65. ajouterButton.setText("Ajouter piÃ¨ce jointe");
66. ajouterButton.addActionListener(**new** java.awt.event.ActionListener() {
67. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
68. ajouterButtonActionPerformed(evt);
69. }
70. });
72. pieceJointeLabel.setText("jLabel1");
74. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(**this**);
75. **this**.setLayout(layout);
76. layout.setHorizontalGroup(
77. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
78. .addGroup(layout.createSequentialGroup()
79. .addContainerGap()
80. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
81. .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 680, Short.MAX\_VALUE)
82. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
83. .addComponent(errorLabel)
84. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
85. .addComponent(okLabel)
86. .addGap(134, 134, 134)
87. .addComponent(inboxButton)
88. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
89. .addComponent(envoyerButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 115, javax.swing.GroupLayout.PREFERRED\_SIZE))
90. .addGroup(layout.createSequentialGroup()
91. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
92. .addComponent(destinataireLabel)
93. .addComponent(sujetLabel))
94. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
95. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
96. .addComponent(destinataireTextField, javax.swing.GroupLayout.Alignment.TRAILING)
97. .addComponent(sujetTextField)))
98. .addGroup(layout.createSequentialGroup()
99. .addComponent(ajouterButton)
100. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
101. .addComponent(pieceJointeLabel)
102. .addGap(0, 0, Short.MAX\_VALUE)))
103. .addContainerGap())
104. );
105. layout.setVerticalGroup(
106. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
107. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
108. .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
109. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
110. .addComponent(destinataireTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
111. .addComponent(destinataireLabel))
112. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
113. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
114. .addComponent(sujetTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
115. .addComponent(sujetLabel))
116. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
117. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
118. .addComponent(ajouterButton)
119. .addComponent(pieceJointeLabel))
120. .addGap(2, 2, 2)
121. .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 161, javax.swing.GroupLayout.PREFERRED\_SIZE)
122. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
123. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
124. .addComponent(envoyerButton)
125. .addComponent(errorLabel)
126. .addComponent(inboxButton)
127. .addComponent(okLabel))
128. .addGap(24, 24, 24))
129. );
130. }// </editor-fold>
132. **private** **void** envoyerButtonActionPerformed(java.awt.event.ActionEvent evt) {
134. errorLabel.setVisible(**false**);
135. okLabel.setVisible(**false**);
137. **if**(sujetTextField.getText().isEmpty())
138. {
139. errorLabel.setText("Vous devez indiquer le sujet du mail");
140. errorLabel.setVisible(**true**);
141. }
143. **if**(destinataireTextField.getText().isEmpty())
144. {
145. errorLabel.setText("Vous devez indiquer un destinataire");
146. errorLabel.setVisible(**true**);
147. }
149. GUI\_Mail container = (GUI\_Mail)SwingUtilities.getWindowAncestor(**this**);
150. Session sess = container.getSession();
152. MimeMessage messageComplet = **new** MimeMessage(sess);
154. **try**
155. {
156. messageComplet.setFrom(**new** InternetAddress(container.getMailAdress()));
157. messageComplet.setRecipient(Message.RecipientType.TO, **new** InternetAddress(destinataireTextField.getText()));
158. messageComplet.setSubject(sujetTextField.getText());
159. messageComplet.setSentDate(**new** Date());
161. Multipart contenu = **new** MimeMultipart();
163. MimeBodyPart corpsMessage = **new** MimeBodyPart();
164. corpsMessage.setText(messageTextArea.getText());
166. contenu.addBodyPart(corpsMessage);
168. **if**(pieceJointe != **null**)
169. contenu.addBodyPart(pieceJointe);
171. messageComplet.setContent(contenu);
172. Transport.send(messageComplet);
173. pieceJointe = **null**;
174. }
175. **catch** (AddressException ex)
176. {
177. errorLabel.setText("Erreur d'envois");
178. errorLabel.setVisible(**true**);
179. ex.printStackTrace();
180. }
181. **catch** (MessagingException ex)
182. {
183. errorLabel.setText("Erreur d'envois");
184. errorLabel.setVisible(**true**);
185. ex.printStackTrace();
186. }
188. resetForm();
189. okLabel.setVisible(**true**);
190. }
192. **private** **void** inboxButtonActionPerformed(java.awt.event.ActionEvent evt) {
194. resetForm();
196. GUI\_Mail container = (GUI\_Mail)SwingUtilities.getWindowAncestor(**this**);
197. container.changeLayout("inbox");
198. }
200. **private** **void** ajouterButtonActionPerformed(java.awt.event.ActionEvent evt) {
202. //On ajoute une piÃ¨ce jointe
203. **if**(ajouterButton.getText().equalsIgnoreCase("Ajouter piÃ¨ce jointe"))
204. {
205. JFileChooser jfc = **new** JFileChooser();
206. jfc.showDialog(**this**, "Choix piÃ¨ce jointe");
208. //Si aucun fichier n'a Ã©tÃ© selectionnÃ© on return
209. **if**(jfc.getSelectedFile() == **null**)
210. **return**;
212. String filePath = jfc.getSelectedFile().getAbsolutePath();
214. //Creation de la partie de message contenant la piÃ¨ce jointe
215. pieceJointe = **new** MimeBodyPart();
216. DataSource ds = **new** FileDataSource(filePath);
217. **try** {
218. pieceJointe.setDataHandler(**new** DataHandler(ds));
219. pieceJointe.setFileName(filePath);
220. } **catch** (MessagingException ex) {
221. Logger.getLogger(sendMessagePanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message erreur
222. }
224. //Changement bouton
225. pieceJointeLabel.setVisible(**true**);
226. pieceJointeLabel.setText(filePath);
227. ajouterButton.setText("Supprimer piece jointe");
228. **return**;
229. }
231. //Si on supprime la piÃ¨ce jointe
233. pieceJointeLabel.setVisible(**false**);
234. pieceJointe = **null**;
235. ajouterButton.setText("Ajouter piÃ¨ce jointe");
236. }
238. **private** **void** resetForm()
239. {
240. errorLabel.setVisible(**false**);
241. okLabel.setVisible(**false**);
242. pieceJointeLabel.setVisible(**false**);
243. pieceJointe = **null**;
245. ajouterButton.setText("Ajouter piÃ¨ce jointe");
246. destinataireTextField.setText("");
247. sujetTextField.setText("");
248. messageTextArea.setText("");
249. }

**Affichage messages + piéces jointes**

1. **public** **class** inboxPanel **extends** javax.swing.JPanel {
3. **private** Store storeMail;
4. **private** Message[] messageList;
5. **private** ArrayList<Message> messageAffiche;
6. **private** Folder fichierMail;
8. **private** File dossierPieceJointe;
10. /\*\*
11. \* Creates new form inboxPanel
12. \*/
13. **public** inboxPanel() {
14. initComponents();
15. inboxList.setModel(**new** DefaultListModel());
16. pieceJointeLabel.setVisible(**false**);
17. telechargerButton.setVisible(**false**);
18. }
20. **public** **void** setStore(Store s)
21. {
22. storeMail = s;
24. **try** {
25. fichierMail = storeMail.getFolder("INBOX");
26. fichierMail.open(Folder.READ\_WRITE);
27. } **catch** (MessagingException ex) {
28. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message erreur;
29. }
30. }
32. **public** **synchronized** **void** refreshMailList()
33. {
34. **try** {
35. //Si le fichier est dÃ©jÃ  ouvert on doit le fermer pour rafraichir
36. **if**(fichierMail.isOpen())
37. fichierMail.close(**true**);
39. fichierMail = storeMail.getFolder("INBOX");
40. fichierMail.open(Folder.READ\_WRITE);
41. messageList = fichierMail.getMessages();
42. } **catch** (MessagingException ex) {
43. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message d'erreur
44. }
46. messageAffiche = **new** ArrayList<>();
47. //On parcours la liste des messages Ã  l'envers (ordre de reception)
48. **for**(**int** i = messageList.length-1; i >= 0 ; i--)
49. {
50. messageAffiche.add(messageList[i]);
51. }
53. refreshJlist();
54. }
56. //Methode pour rafraichir l'affichage des mails dans le GUI
57. **private** **void** refreshJlist()
58. {
59. //On vide la liste
60. DefaultListModel l = (DefaultListModel) inboxList.getModel();
61. l.clear();
63. **try**
64. {
65. **for**(Message m : messageAffiche)
66. l.addElement(m.getSubject());
67. }
68. **catch** (MessagingException ex)
69. {
70. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message d'erreur
71. }
73. }
75. /\*\*
76. \* This method is called from within the constructor to initialize the form.
77. \* WARNING: Do NOT modify this code. The content of this method is always
78. \* regenerated by the Form Editor.
79. \*/
80. @SuppressWarnings("unchecked")
81. // <editor-fold defaultstate="collapsed" desc="Generated Code">
82. **private** **void** initComponents() {
84. jButton1 = **new** javax.swing.JButton();
85. jScrollPane1 = **new** javax.swing.JScrollPane();
86. inboxList = **new** javax.swing.JList();
87. inboxLabel = **new** javax.swing.JLabel();
88. nouveauButton = **new** javax.swing.JButton();
89. jScrollPane2 = **new** javax.swing.JScrollPane();
90. contenusTextArea = **new** javax.swing.JTextArea();
91. fromLabel = **new** javax.swing.JLabel();
92. receiveDateLabel = **new** javax.swing.JLabel();
93. logoutButton = **new** javax.swing.JButton();
94. supprimerButton = **new** javax.swing.JButton();
95. refreshButton = **new** javax.swing.JButton();
96. telechargerButton = **new** javax.swing.JButton();
97. pieceJointeLabel = **new** javax.swing.JLabel();
99. jButton1.setText("jButton1");
101. inboxList.addMouseListener(**new** java.awt.event.MouseAdapter() {
102. **public** **void** mouseClicked(java.awt.event.MouseEvent evt) {
103. inboxListMouseClicked(evt);
104. }
105. });
106. jScrollPane1.setViewportView(inboxList);
108. inboxLabel.setText("Inbox : ");
110. nouveauButton.setText("Nouveau");
111. nouveauButton.addActionListener(**new** java.awt.event.ActionListener() {
112. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
113. nouveauButtonActionPerformed(evt);
114. }
115. });
117. contenusTextArea.setEditable(**false**);
118. contenusTextArea.setColumns(20);
119. contenusTextArea.setLineWrap(**true**);
120. contenusTextArea.setRows(5);
121. jScrollPane2.setViewportView(contenusTextArea);
123. fromLabel.setText("From : ");
125. receiveDateLabel.setText("Date : ");
126. receiveDateLabel.setToolTipText("");
128. logoutButton.setText("Logout");
129. logoutButton.addActionListener(**new** java.awt.event.ActionListener() {
130. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
131. logoutButtonActionPerformed(evt);
132. }
133. });
135. supprimerButton.setText("Supprimer");
136. supprimerButton.addActionListener(**new** java.awt.event.ActionListener() {
137. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
138. supprimerButtonActionPerformed(evt);
139. }
140. });
142. refreshButton.setText("Rafraichir");
143. refreshButton.addActionListener(**new** java.awt.event.ActionListener() {
144. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
145. refreshButtonActionPerformed(evt);
146. }
147. });
149. telechargerButton.setText("TÃ©lÃ©charger la piÃ¨ce jointe");
150. telechargerButton.setToolTipText("");
151. telechargerButton.addActionListener(**new** java.awt.event.ActionListener() {
152. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
153. telechargerButtonActionPerformed(evt);
154. }
155. });
157. pieceJointeLabel.setText("jLabel1");
159. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(**this**);
160. **this**.setLayout(layout);
161. layout.setHorizontalGroup(
162. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
163. .addGroup(layout.createSequentialGroup()
164. .addContainerGap()
165. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
166. .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 152, javax.swing.GroupLayout.PREFERRED\_SIZE)
167. .addComponent(inboxLabel))
168. .addGap(18, 18, 18)
169. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
170. .addGroup(layout.createSequentialGroup()
171. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
172. .addGroup(layout.createSequentialGroup()
173. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
174. .addGroup(layout.createSequentialGroup()
175. .addComponent(nouveauButton)
176. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
177. .addComponent(supprimerButton))
178. .addComponent(fromLabel))
179. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))
180. .addGroup(layout.createSequentialGroup()
181. .addComponent(receiveDateLabel)
182. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 141, Short.MAX\_VALUE)
183. .addComponent(pieceJointeLabel)
184. .addGap(97, 97, 97)))
185. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
186. .addGroup(layout.createSequentialGroup()
187. .addGap(19, 19, 19)
188. .addComponent(refreshButton)
189. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
190. .addComponent(logoutButton))
191. .addComponent(telechargerButton, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 169, javax.swing.GroupLayout.PREFERRED\_SIZE))
192. .addGap(13, 13, 13))
193. .addComponent(jScrollPane2))
194. .addGap(21, 21, 21))
195. );
196. layout.setVerticalGroup(
197. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
198. .addGroup(layout.createSequentialGroup()
199. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
200. .addGroup(layout.createSequentialGroup()
201. .addGap(0, 0, Short.MAX\_VALUE)
202. .addComponent(telechargerButton))
203. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
204. .addGap(8, 8, 8)
205. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
206. .addComponent(nouveauButton)
207. .addComponent(logoutButton)
208. .addComponent(supprimerButton)
209. .addComponent(refreshButton))
210. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
211. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
212. .addComponent(fromLabel)
213. .addComponent(inboxLabel))
214. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
215. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
216. .addComponent(receiveDateLabel)
217. .addComponent(pieceJointeLabel))))
218. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
219. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**)
220. .addComponent(jScrollPane1)
221. .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT\_SIZE, 267, Short.MAX\_VALUE))
222. .addGap(17, 17, 17))
223. );
224. }// </editor-fold>
226. **private** **void** nouveauButtonActionPerformed(java.awt.event.ActionEvent evt) {
227. GUI\_Mail container = (GUI\_Mail)SwingUtilities.getWindowAncestor(**this**);
228. container.changeLayout("nouveauMessage");
229. }
231. **private** **void** logoutButtonActionPerformed(java.awt.event.ActionEvent evt) {
233. **try** {
234. fichierMail.close(**true**);
235. storeMail.close();
236. } **catch** (MessagingException ex) {
237. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message erreur
238. }
240. GUI\_Mail container = (GUI\_Mail)SwingUtilities.getWindowAncestor(**this**);
241. container.changeLayout("connexion");
242. }
244. **private** **void** supprimerButtonActionPerformed(java.awt.event.ActionEvent evt) {
245. **if**(inboxList.getSelectedIndex() < 0)
246. **return**;
247. //Recuperation du mail dans la liste
248. Message m = messageAffiche.get(inboxList.getSelectedIndex());
250. //On affiche une boite de dialog pour confirmer la suppression
251. **int** optionDialogue = JOptionPane.YES\_NO\_OPTION;
252. **try** {
253. optionDialogue = JOptionPane.showConfirmDialog(**null**, "Voulez vous vraiment supprimer ce mail : " + m.getSubject(),"Demande suppression" ,optionDialogue);
254. } **catch** (MessagingException ex) {
255. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message erreur
256. }
258. //Le mail est marque Ã  supprimer (pop3 supprime dÃ©finitivement Ã  la fermeture du fichier)
259. **if**(optionDialogue == JOptionPane.YES\_OPTION)
260. {
261. **try** {
262. m.setFlag(Flags.Flag.DELETED, **true**);
263. messageAffiche.remove(m);//On supprime le mail de la liste
264. refreshJlist();//On rafraichis l'affichage
265. } **catch** (MessagingException ex) {
266. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);//TO DO message d'erreur
267. }
268. }
269. }
271. **private** **void** refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
272. refreshMailList();
273. }
275. //Affichage des messages quand on en sÃ©lectionne un avec la souris
276. **private** **void** inboxListMouseClicked(java.awt.event.MouseEvent evt) {
277. **if**(inboxList.getSelectedIndex() < 0)
278. **return**;
280. pieceJointeLabel.setVisible(**false**);
281. telechargerButton.setVisible(**false**);
282. Message m = messageAffiche.get(inboxList.getSelectedIndex());
284. **try** {
285. **if**(m.isMimeType("text/plain"))//Le mail est juste du texte
286. contenusTextArea.setText(m.getContent().toString());
287. **else**//Le mail est composÃ© de plusieurs parties
288. {
289. Multipart contenu = (Multipart)m.getContent();
291. **int** nbrDeMorceaux = contenu.getCount();
293. **for**(**int** cpt = 0; cpt < nbrDeMorceaux; cpt++)
294. {
295. Part morceau = contenu.getBodyPart(cpt);
297. //RÃ©cupÃ©ration de l'emplacement dela piÃ¨ce jointe (dans le mail ou sur un serveur distant)
298. String disposition  = morceau.getDisposition();
300. **if**(morceau.isMimeType("text/plain") && disposition == **null**)//Si c'est du text
301. {
302. contenusTextArea.setText(morceau.getContent().toString());
303. }
304. //piece jointe
305. **if**(disposition != **null** && disposition.equalsIgnoreCase(Part.ATTACHMENT))
306. {
307. Path documentRecus = Paths.get(morceau.getFileName());
309. pieceJointeLabel.setVisible(**true**);
310. pieceJointeLabel.setText("Piece jointe : " + documentRecus.getFileName().toString());
311. telechargerButton.setVisible(**true**);
312. }
313. }
314. }
316. //Parcours de la liste des personnes qui ont envoyÃ© un mail
317. String from = "From : ";
318. **for**(Address a : m.getFrom())
319. {
320. from += " "+a.toString();
321. }
322. fromLabel.setText(from);
324. //Date de reception TO DO
325. SimpleDateFormat df = **new** SimpleDateFormat("dd/MM/yyyy HH:mm");
326. receiveDateLabel.setText(df.format(m.getSentDate()));
327. } **catch** (IOException ex) {
328. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);
329. } **catch** (MessagingException ex) {
330. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);
331. }
332. }

335. //Creation du dossier pour les pieces jointes du user en cours (si necessaire)
336. **public** **void** createDirectory(String user)
337. {
338. dossierPieceJointe = **new** File(user);
340. **if**(!dossierPieceJointe.exists())
341. dossierPieceJointe.mkdir();
342. }

345. **private** **void** telechargerButtonActionPerformed(java.awt.event.ActionEvent evt) {
346. Message m = messageAffiche.get(inboxList.getSelectedIndex());
348. **try**
349. {
350. Multipart contenu = (Multipart)m.getContent();
352. **int** nbrDeMorceaux = contenu.getCount();
354. **for**(**int** cpt = 0; cpt < nbrDeMorceaux; cpt++)
355. {
356. Part morceau = contenu.getBodyPart(cpt);
358. //RÃ©cupÃ©ration de l'emplacement dela piÃ¨ce jointe (dans le mail ou sur un serveur distant)
359. String disposition  = morceau.getDisposition();
361. **if**(disposition != **null** && disposition.equalsIgnoreCase(Part.ATTACHMENT))
362. {
363. InputStream is = morceau.getInputStream();
364. ByteArrayOutputStream baos = **new** ByteArrayOutputStream();
365. **int** carac;
367. **while**((carac = is.read()) != -1)
368. baos.write(carac);
370. baos.flush();
372. Path documentRecus = Paths.get(morceau.getFileName());
374. String fileName = dossierPieceJointe.getAbsolutePath() + System.getProperty("file.separator");
375. fileName += documentRecus.getFileName().toString();//On recupere juste le nom du fichier reÃ§us sans le path
377. FileOutputStream fos = **new** FileOutputStream(fileName);
379. baos.writeTo(fos);
380. fos.close();
382. Desktop.getDesktop().open(**new** File(fileName));
383. }
384. }
385. } **catch** (IOException ex) {
386. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);
387. } **catch** (MessagingException ex) {
388. Logger.getLogger(inboxPanel.**class**.getName()).log(Level.SEVERE, **null**, ex);
389. }
390. }

**Test de conformité**

*« Peut-on considérer que la hauteur des pieds est de 265cm pour la parcelle Est ? »*

**Données connues :**

**Echantillon de hauteur de pieds de la parcelle Est**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 278 | 260 | 217 | 247 | 358 | 251 | 324 | 328 |
| 300 | 266 | 231 | 313 | 279 | 236 | 256 | 293 |
| 228 | 269 | 287 | 196 | 218 | 249 | 321 | 319 |
| 280 | 265 | 296 | 240 | 311 | 283 | 272 | 246 |

n = 32

**Calcul des données manquantes pour le test de conformité :**

**Moyenne :**

**Formule :**

**Résultat :**

**Estimation de la variance de la population :**

**Formule** :

**Résultat** :

**Ecart-type de la population () :**

**Test de conformité :**

Comme n > 30 on fait notre test en utilisant la loi normale. Test bilatéral avec comme seuil de signification 0,05.

**Hypothèse** :

H0 :   
 H1 :

**Formule** :

**Résultat :**

Dans la table de la loi normale = 1,96.

Comme 1,114 < 1,96 on ne se situe pas dans la zone de rejet. L’hypothèse H0 est donc acceptée.

**Test d’homogénéité**

*« La hauteur des pieds est-elle similaire pour les parcelles Nord et Sud ? »*

**Données :**

N = 14 (petit échantillon -> loi de Student en bilatéral)

Nombre de degré de liberté = n1 + n2 – 2 = 14 + 14 - 2 = 26

|  |  |
| --- | --- |
| **Nord** | **Sud** |
| 199 | 215 |
| 205 | 242 |
| 173 | 197 |
| 233 | 265 |
| 206 | 227 |
| 261 | 244 |
| 155 | 222 |
| 214 | 238 |
| 174 | 235 |
| 207 | 217 |
| 196 | 184 |
| 224 | 166 |
| 237 | 231 |
| 210 | 209 |

Moyenne parcelle Nord (m1) = 206,71

Moyenne parcelle Sud (m2) = 220,85

**Test d’homogénéité :**

**Hypothèse :**

H0 : m1 = m2  
H1 : m1 m2

Seuil de signification 0,05

**Estimation de la variance pour les deux ensembles :**

**Ecart réduit :**

Dans la table de la loi de Student, la valeur critique pour est 2,055

Comme :

-2,055 < -1,2925 < 2,055

On ne se trouve pas dans la zone de rejet. La différence des moyennes n’est pas significative on accepte dont l’hypothèse H0.

**Corrélation**

*« La hauteur dépend-elle de la masse dans la parcelle est ? »*

**Données connues :**

**N = 32**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **xi (masse)** | **yi (hauteur)** |  |  |  |  |  |
| 2004 | 278 | 10194.739 | 31.28948 | 0.50757 | 0.151268 | 0.076779 |
| 1853 | 260 | 2503.10096 | 153.9163 | -0.25151 | -0.3355 | 0.08438 |
| 1623 | 217 | 78417.361 | 3069.858 | -1.40771 | -1.49833 | 2.109215 |
| 1660 | 247 | 59064.067 | 645.4801 | -1.22171 | -0.68705 | 0.83938 |
| 2206 | 358 | 91790.215 | 7326.281 | 1.523022 | 2.314672 | 3.525296 |
| 1838 | 251 | 4229.03096 | 458.2297 | -0.32691 | -0.57888 | 0.189242 |
| 2314 | 324 | 168895.519 | 2661.91 | 2.065936 | 1.395225 | 2.882447 |
| 2119 | 328 | 46642.609 | 3090.659 | 1.085674 | 1.503395 | 1.632197 |
| 2067 | 300 | 26885.833 | 761.4123 | 0.82427 | 0.746204 | 0.615074 |
| 1821 | 266 | 6729.08496 | 41.04068 | -0.41237 | -0.17324 | 0.07144 |
| 1771 | 231 | 17432.185 | 1714.482 | -0.66372 | -1.11973 | 0.743187 |
| 1967 | 313 | 4092.03296 | 1647.848 | 0.321571 | 1.097757 | 0.353007 |
| 1925 | 279 | 482.636961 | 43.47688 | 0.110438 | 0.17831 | 0.019692 |
| 1792 | 236 | 12327.883 | 1325.419 | -0.55815 | -0.98452 | 0.549511 |
| 1818 | 256 | 7230.27096 | 269.1667 | -0.42745 | -0.44367 | 0.189646 |
| 2147 | 293 | 59520.873 | 424.1005 | 1.226429 | 0.556906 | 0.683006 |
| 1706 | 228 | 38821.215 | 1971.919 | -0.99047 | -1.20086 | 1.189419 |
| 1914 | 269 | 120.318961 | 11.60288 | 0.055141 | -0.09212 | -0.00508 |
| 1860 | 287 | 1851.66696 | 212.9761 | -0.21632 | 0.394651 | -0.08537 |
| 1525 | 196 | 142907.437 | 5837.923 | -1.90036 | -2.06622 | 3.926559 |
| 1456 | 218 | 199836.715 | 2960.045 | -2.24722 | -1.47129 | 3.306301 |
| 1800 | 249 | 10615.387 | 547.8549 | -0.51794 | -0.63297 | 0.327836 |
| 2293 | 321 | 152075.821 | 2361.348 | 1.96037 | 1.314098 | 2.576117 |
| 2045 | 319 | 20155.197 | 2170.973 | 0.713677 | 1.260012 | 0.899241 |
| 1964 | 280 | 3717.21896 | 57.66428 | 0.30649 | 0.205353 | 0.062939 |
| 1794 | 265 | 11887.759 | 54.85328 | -0.5481 | -0.20029 | 0.109776 |
| 2098 | 296 | 38012.911 | 556.6627 | 0.980107 | 0.638034 | 0.625341 |
| 1812 | 240 | 8286.64296 | 1050.168 | -0.45761 | -0.87635 | 0.401028 |
| 2011 | 311 | 11657.305 | 1489.474 | 0.542759 | 1.043672 | 0.566462 |
| 1971 | 283 | 4619.78496 | 112.2265 | 0.341679 | 0.286481 | 0.097885 |
| 1966 | 272 | 3965.09496 | 0.16508 | 0.316544 | -0.01099 | -0.00348 |
| 1757 | 246 | 21325.053 | 697.2927 | -0.7341 | -0.71409 | 0.524213 |
|  |  |  |  |  |  |  |
| ***Total*** | ***Total*** | ***Total*** | ***Total*** |  |  | ***Total*** |
| 60897 | 8717 | 1266292.97 | 43757.72 |  |  | 29.08269 |
|  |  |  |  |  |  |  |
| ***Moyenne(*** | ***Moyenne(*** | ***Ecart-type()*** | ***Ecart-type(*** |  |  |  |
| 1903,031 | 272,4063 | 198.926256 | 36.97876 |  |  |  |

**Formules** :

**Ecart type :**

**𝒓type x:**

**Formule du coefficient de corrélation :**

**Valeur de r pour la parcelle Est :**

Le coefficient r étant très proche de 1, la corrélation est forte.

On peut donc conclure que la hauteur dépend bel et bien de la masse pour la parcelle Est.