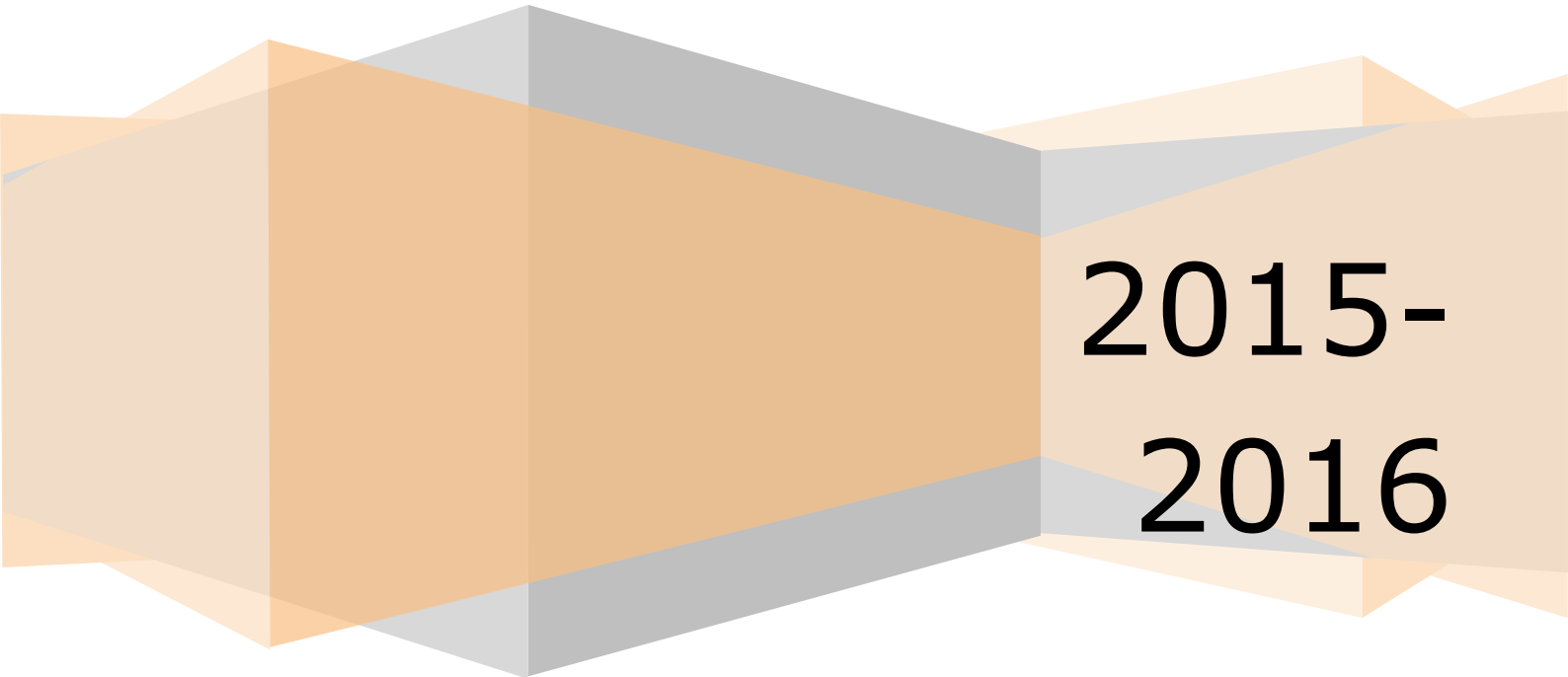


FINK Jérôme & SEEL Océane

Laboratoire E-commerce

Phase 1 : Data mining et informations
statistiques



**2015-
2016**

Sommaire

1. Code du client Applic_Data_Analysis	3
1.1 ApplicationDataAnalysis.java	3
1.2 Login.java	4
1.3 Menu.java	6
1.4 StatDescrCont.java	7
1.5 GrCouleurRep.java	9
1.6 GrCouleurComp.java	12
1.7 StatInferTestConf.java	14
1.8 StatInferTestHomog.java	15
1.9 StatInferTestAnova.java	16
1.10 ProtocolePIDEP.java	17
1.11 Utility.java	18
2. Explications des 3 requêtes d'inférence statistique	21
2.1 Test d'hypothèse de conformité	21
2.2 Test d'hypothèse d'homogénéité	21
2.3 Test d'hypothèse de type ANOVA	22

1. Code du client Applic_Data_Analysis

1.1 ApplicationDataAnalysis.java

```
1. package application_data_analysis;
2.
3. import java.awt.CardLayout;
4. import java.net.*;
5.
6.
7. public class ApplicationDataAnalysis extends javax.swing.JFrame
8. {
9.     public static Socket cliSock = null;
10.    public Boolean isConnected = false;
11.
12.    public ApplicationDataAnalysis()
13.    {
14.        initComponents();
15.
16.        this.setTitle("Data Analysis");
17.        Utility.InitialisationFlux();
18.
19.        // Lancement du login
20.        (new Login(this, true)).setVisible(true);
21.        System.out.println("isConnected = " + isConnected);
22.        if (!isConnected)
23.            System.exit(0);
24.    }
25.
26.    public void ChangePanel(String newPanel)
27.    {
28.        CardLayout card = (CardLayout)this.getContentPane().getLayout();
29.        card.show(this.getContentPane(), newPanel);
30.    }
31.
32.    public static void main(String args[]) {
33.        /* Set the Nimbus look and feel */
34.        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
35.        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
36.         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
37.         */
38.        try {
39.            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
40.                if ("Nimbus".equals(info.getName())) {
41.                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
42.                    break;
43.                }
44.            }
45.        } catch (ClassNotFoundException ex) {
46.            java.util.logging.Logger.getLogger(ApplicationDataAnalysis.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
47.        } catch (InstantiationException ex) {
48.            java.util.logging.Logger.getLogger(ApplicationDataAnalysis.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
49.        } catch (IllegalAccessException ex) {
```

```

50.         java.util.logging.Logger.getLogger(ApplicationDataAnalysis.class.getName
    ().log(java.util.logging.Level.SEVERE, null, ex);
51.     } catch (javax.swing.UnsupportedLookAndFeelException ex) {
52.         java.util.logging.Logger.getLogger(ApplicationDataAnalysis.class.getName
    ().log(java.util.logging.Level.SEVERE, null, ex);
53.     }
54.     //</editor-fold>
55.
56.     /* Create and display the form */
57.     java.awt.EventQueue.invokeLater(new Runnable() {
58.         public void run() {
59.             new ApplicationDataAnalysis().setVisible(true);
60.         }
61.     });
62. }
63. // Variables declaration - do not modify
64. private javax.swing.ButtonGroup ButtonGroup;
65. private application_data_analysis.GrCouleurComp GrCouleurComp;
66. private application_data_analysis.GrCouleurRep GrCouleurRep;
67. private application_data_analysis.Menu Menu;
68. private application_data_analysis.StatDescrCont StatDescrCont;
69. private application_data_analysis.StatInferTestAnova StatInferTestAnova;
70. private application_data_analysis.StatInferTestConf StatInferTestConf;
71. private application_data_analysis.StatInferTestHomog StatInferTestHomog;
72. // End of variables declaration
73. }

```

1.2 Login.java

```

1. package application_data_analysis;
2.
3. import java.io.*;
4. import java.security.*;
5. import java.util.Date;
6.
7.
8. public class Login extends javax.swing.JDialog
9. {
10.     public Login(java.awt.Frame parent, boolean modal)
11.     {
12.         super(parent, modal);
13.         initComponents();
14.         this.setTitle("Login");
15.         ErrorLabel.setVisible(false);
16.     }
17.
18.     private void ConnexionButtonActionPerformed(java.awt.event.ActionEvent evt) {
19.
20.         try
21.         {
22.             if (PwdPF.getPassword().length == 0 || LoginTF.getText().isEmpty())
23.                 return;
24.
25.             // sels
26.             long temps = (new Date()).getTime();
27.             double aleatoire = Math.random();
28.             String Password = new String(PwdPF.getPassword());

```

```

29.
30.         // digest
31.         MessageDigest md = MessageDigest.getInstance("SHA-1");
32.         md.update>Password.getBytes());
33.         ByteArrayOutputStream baos = new ByteArrayOutputStream();
34.         DataOutputStream bdos = new DataOutputStream(baos);
35.         bdos.writeLong(temps);
36.         bdos.writeDouble(aleatoire);
37.         md.update(baos.toByteArray());
38.         byte[] pwdDigest = md.digest();
39.
40.         // envoi
41.         Utility.SendMsg(ProtocolePIDEP.LOGIN, "");
42.         Utility.dos.writeUTF(LoginTF.getText());
43.         Utility.dos.writeLong(temps);
44.         Utility.dos.writeDouble(aleatoire);
45.         Utility.dos.writeInt(pwdDigest.length);
46.         Utility.dos.write(pwdDigest);
47.         Utility.dos.flush();
48.
49.         // r ponse
50.         String reponse = Utility.ReceiveMsg();
51.         String[] parts = reponse.split("#");
52.
53.         if (parts[0].equals("OUI"))
54.         {
55.             ApplicationDataAnalysis a = (ApplicationDataAnalysis) this.getParent
56.             ();
57.             a.isConnected = true;
58.             this.dispose();
59.         }
60.         else
61.             ErrorLabel.setVisible(true);
62.         catch (NoSuchAlgorithmException ex)
63.         {
64.             System.err.println("Login : NoSuchAlgorithmException : " + ex.getMessage
65.             ());
66.         }
67.         catch (IOException ex)
68.         {
69.             System.err.println("Login : IOException : " + ex.getMessage());
70.         }
71.
72.
73.         /**
74.          * @param args the command line arguments
75.          */
76.         public static void main(String args[]) {
77.             /* Set the Nimbus look and feel */
78.             //<editor-
79.             fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
80.             /* If Nimbus (introduced in Java SE 6) is not available, stay with the defau
81.             lt look and feel.
82.             * For details see http://download.oracle.com/javase/tutorial/uiswing/lookan
83.             dfeel/plaf.html
84.             */
85.             try {
86.                 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
87.                 getInstalledLookAndFeels()) {
88.                     if ("Nimbus".equals(info.getName())) {
89.                         javax.swing.UIManager.setLookAndFeel(info.getClassName());

```

```

86.         break;
87.     }
88. }
89. } catch (ClassNotFoundException ex) {
90.     java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
91. } catch (InstantiationException ex) {
92.     java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
93. } catch (IllegalAccessException ex) {
94.     java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
95. } catch (javax.swing.UnsupportedLookAndFeelException ex) {
96.     java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
97. }
98. //</editor-fold>
99.
100.     /* Create and display the dialog */
101.     java.awt.EventQueue.invokeLater(new Runnable() {
102.         public void run() {
103.             Login dialog = new Login(new javax.swing.JFrame(), true);
104.             dialog.addWindowListener(new java.awt.event.WindowAdapter() {
105.
106.                 @Override
107.                 public void windowClosing(java.awt.event.WindowEvent e) {
108.
109.                     System.exit(0);
110.                 }
111.             });
112.             dialog.setVisible(true);
113.         }
114.     });
115.
116.     // Variables declaration - do not modify
117.     private javax.swing.JButton ConnexionButton;
118.     private javax.swing.JLabel ErrorLabel;
119.     private javax.swing.JLabel LoginLabel;
120.     private javax.swing.JTextField LoginTF;
121.     private javax.swing.JLabel PwdLabel;
122.     private javax.swing.JPasswordField PwdPF;
123.     // End of variables declaration
124. }

```

1.3 Menu.java

```

1. package application_data_analysis;
2.
3. import javax.swing.SwingUtilities;
4.
5.
6. public class Menu extends javax.swing.JPanel
7. {
8.     public Menu()
9.     {
10.         initComponents();
11.     }

```

```
12.
13.     private void StatDescrContButtonActionPerformed(java.awt.event.ActionEvent evt)
14.     {
15.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
16.         app.ChangePanel("StatDescrCont");
17.     }
18.     private void GrCouleurRepButtonActionPerformed(java.awt.event.ActionEvent evt) {
19.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
20.         app.ChangePanel("GrCouleurRep");
21.     }
22.
23.     private void GrCouleurCompButtonActionPerformed(java.awt.event.ActionEvent evt)
24.     {
25.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
26.         app.ChangePanel("GrCouleurComp");
27.     }
28.     private void StatInferTestConfButtonActionPerformed(java.awt.event.ActionEvent evt) {
29.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
30.         app.ChangePanel("StatInferTestConf");
31.     }
32.
33.     private void StatInferTestHomogButtonActionPerformed(java.awt.event.ActionEvent evt) {
34.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
35.         app.ChangePanel("StatInferTestHomog");
36.     }
37.
38.     private void StatInferTestAnovaButtonActionPerformed(java.awt.event.ActionEvent evt) {
39.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
40.         app.ChangePanel("StatInferTestAnova");
41.     }
42.
43.     private void QuitterButtonActionPerformed(java.awt.event.ActionEvent evt) {
44.         Utility.SendMsg(ProtocolePIDEP.LOGOUT, null);
45.         System.exit(0);
46.     }
47. }
```

1.4 StatDescrCont.java

```
1. package application_data_analysis;
2.
3. import javax.swing.SwingUtilities;
4.
5.
6. public class StatDescrCont extends javax.swing.JPanel
```

```

7.  {
8.      public StatDescrCont()
9.      {
10.         initComponents();
11.
12.         ButtonGroup.add(DechargesRB);
13.         ButtonGroup.add(ChargesRB);
14.
15.         // Cacher les labels
16.         ErrorSaisieLabel.setVisible(false);
17.         ErrorCalculLabel.setVisible(false);
18.         MoyenneReponseLabel.setVisible(false);
19.         ModeReponseLabel.setVisible(false);
20.         MedianeReponseLabel.setVisible(false);
21.         EcartTypeReponseLabel.setVisible(false);
22.     }
23.
24.     private void CalculerButtonActionPerformed(java.awt.event.ActionEvent evt) {
25.
26.         ErrorSaisieLabel.setVisible(false);
27.         ErrorCalculLabel.setVisible(false);
28.
29.         try
30.         {
31.             int nbContainers = Integer.parseInt(NbContainersTF.getText());
32.             if (nbContainers <= 0)
33.             {
34.                 ErrorSaisieLabel.setText("Doit  tre un entier positif !");
35.                 ErrorSaisieLabel.setVisible(true);
36.                 return;
37.             }
38.
39.             String mouvement;
40.             if (ChargesRB.isSelected())
41.                 mouvement = "OUT";
42.             else if (DechargesRB.isSelected())
43.                 mouvement = "IN";
44.             else
45.             {
46.                 ErrorSaisieLabel.setText("S lectionner 'Charg s' ou 'D charg s'");
47.                 ErrorSaisieLabel.setVisible(true);
48.                 return;
49.             }
50.
51.             String ChargeUtile = nbContainers + "#" + mouvement;
52.             Utility.SendMsg(ProtocolePIDEP.GET_STAT_DESCR_CONT, ChargeUtile);
53.
54.             // R ponse
55.             String reponse = Utility.ReceiveMsg();
56.             String[] parts = reponse.split("#");
57.
58.             if (parts[0].equals("NON")) // Erreur
59.             {
60.                 ErrorCalculLabel.setText(parts[1]);
61.                 ErrorCalculLabel.setVisible(true);
62.             }
63.             else
64.             {
65.                 MoyenneReponseLabel.setText(parts[0]);
66.                 MoyenneReponseLabel.setVisible(true);
67.                 ModeReponseLabel.setText(parts[1]);

```



```

68.         ModeReponseLabel.setVisible(true);
69.         MedianeReponseLabel.setText(parts[2]);
70.         MedianeReponseLabel.setVisible(true);
71.         EcartTypeReponseLabel.setText(parts[3]);
72.         EcartTypeReponseLabel.setVisible(true);
73.     }
74. }
75. catch (NumberFormatException ex)
76. {
77.     ErrorSaisieLabel.setText("Doit Être un entier positif !");
78.     ErrorSaisieLabel.setVisible(true);
79. }
80. }
81.
82. private void RetourMenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
83.     ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
84.     app.ChangePanel("Menu");
85. }
86. }

```

1.5 GrCouleurRep.java

```

1. package application_data_analysis;
2.
3. import java.io.IOException;
4. import java.io.ObjectInputStream;
5. import java.util.ArrayList;
6. import java.util.HashMap;
7. import javax.swing.SwingUtilities;
8. import javax.swing.JDialog;
9. import org.jfree.chart.ChartFactory;
10. import org.jfree.chart.ChartPanel;
11. import org.jfree.chart.JFreeChart;
12. import org.jfree.data.general.DefaultPieDataset;
13.
14.
15. public class GrCouleurRep extends javax.swing.JPanel
16. {
17.     public GrCouleurRep() {
18.         initComponents();
19.         ErrorAnneeLabel.setVisible(false);
20.         ErrorMoisLabel.setVisible(false);
21.         ErrorNoDataLabel.setVisible(false);
22.     }
23.
24.     private void MenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
25.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
26.         app.ChangePanel("Menu");
27.     }
28.
29.     private void CalculerAnneeButtonActionPerformed(java.awt.event.ActionEvent evt)
30.     {
31.         ErrorAnneeLabel.setVisible(false);

```

```

32.      ErrorMoisLabel.setVisible(false);
33.      ErrorNoDataLabel.setVisible(false);
34.
35.      try
36.      {
37.          int annee = Integer.parseInt(AnneeTF.getText());
38.
39.          if (annee <= 1000 || 9999 <= annee)
40.          {
41.              ErrorAnneeLabel.setVisible(true);
42.              return;
43.          }
44.
45.
46.          Utility.SendMsg(ProtocolePIDEP.GET_GR_COULEUR_REP, AnneeTF.getText());
47.          AnneeTF.setText("");
48.
49.
50.          String reponse = Utility.ReceiveMsg();
51.          String[] parts = reponse.split("#");
52.
53.          if(parts[0].equals("NON"))
54.          {
55.              ErrorNoDataLabel.setVisible(true);
56.              return;
57.          }
58.
59.
60.          ObjectInputStream ois = new ObjectInputStream(ApplicationDataAnalysis.cl
iSock.getInputStream());
61.          HashMap<String, Object> map = (HashMap<String, Object>) ois.readObject()
;
62.          ShowPieChart(map);
63.      }
64.      catch(NumberFormatException ex)
65.      {
66.          ErrorAnneeLabel.setVisible(true);
67.      }
68.      catch (IOException ex)
69.      {
70.          System.err.println("GrCouleurRep : IOException : " + ex.getMessage());
71.      }
72.      catch (ClassNotFoundException ex)
73.      {
74.          System.err.println("GrCouleurRep : ClassNotFoundException : " + ex.getMe
ssage());
75.      }
76.    }
77.
78.    private void CalculerMoisButtonActionPerformed(java.awt.event.ActionEvent evt) {
79.
80.        ErrorAnneeLabel.setVisible(false);
81.        ErrorMoisLabel.setVisible(false);
82.        ErrorNoDataLabel.setVisible(false);
83.
84.        try
85.        {
86.            int mois = Integer.parseInt(MoisTF.getText());
87.
88.            if (mois < 1 || 12 < mois)
89.            {
90.                ErrorMoisLabel.setVisible(true);

```

```

91.         return;
92.     }
93.
94.
95.     Utility.SendMsg(ProtocolePIDEP.GET_GR_COULEUR_REP, MoisTF.getText());
96.     MoisTF.setText("");
97.
98.
99.     String reponse = Utility.ReceiveMsg();
100.    String[] parts = reponse.split("#");
101.
102.    if(parts[0].equals("NON"))
103.    {
104.        ErrorNoDataLabel.setVisible(true);
105.        return;
106.    }
107.
108.
109.    ObjectInputStream ois = new ObjectInputStream(ApplicationDataAnal
110.    ysis.cliSock.getInputStream());
111.    HashMap<String, Object> map = (HashMap<String, Object>) ois.readO
112.    bject();
113.    ShowPieChart(map);
114.    }
115.    catch(NumberFormatException ex)
116.    {
117.        ErrorMoisLabel.setVisible(true);
118.        System.err.println("GrCouleurRep : NumberFormatException : " + ex
119.        .getMessage());
120.    }
121.    catch (IOException ex)
122.    {
123.        System.err.println("GrCouleurRep : IOException : " + ex.getMessag
124.        e());
125.    }
126.    catch (ClassNotFoundException ex)
127.    {
128.        System.err.println("GrCouleurRep : ClassNotFoundException : " + e
129.        x.getMessage());
130.    }
131.    }
132.
133.    public void ShowPieChart(HashMap<String, Object> map)
134.    {
135.        ArrayList<String> listDestinations = (ArrayList<String>)map.get("DEST
136.        INATIONS");
137.        ArrayList<Integer> listCount = (ArrayList<Integer>)map.get("COUNT");
138.
139.        DefaultPieDataset dpds = new DefaultPieDataset();
140.        for(int i = 0; i < listDestinations.size(); i++)
141.            dpds.setValue(listDestinations.get(i), listCount.get(i));
142.
143.        JFreeChart jfc = ChartFactory.createPieChart("RÃ©partition du nombre
144.        de containers par destination", dpds, true, true, true);
145.        ChartPanel cp = new ChartPanel(jfc);
146.        JDialog dialog = new JDialog();
147.        dialog.setSize(500, 500);
148.        dialog.setContentPane(cp);
149.        dialog.setTitle("RÃ©partition du nombre de containers par destination
150.        ");
151.        dialog.setVisible(true);
152.    }

```

```
145.     }
```

1.6 GrCouleurComp.java

```
1.  package application_data_analysis;
2.
3.  import java.io.IOException;
4.  import java.io.ObjectInputStream;
5.  import java.util.ArrayList;
6.  import java.util.HashMap;
7.  import javax.swing.SwingUtilities;
8.  import javax.swing.JDialog;
9.  import org.jfree.chart.ChartFactory;
10. import org.jfree.chart.ChartPanel;
11. import org.jfree.chart.JFreeChart;
12. import org.jfree.chart.axis.NumberTickUnit;
13. import org.jfree.chart.plot.CategoryPlot;
14. import org.jfree.chart.plot.PlotOrientation;
15. import org.jfree.data.category.DefaultCategoryDataset;
16. import org.jfree.chart.axis.NumberAxis;
17.
18.
19. public class GrCouleurComp extends javax.swing.JPanel
20. {
21.     public GrCouleurComp()
22.     {
23.         initComponents();
24.         ErrorAnneeLabel.setVisible(false);
25.         ErrorNoDataLabel.setVisible(false);
26.     }
27.
28.     private void CalculerButtonActionPerformed(java.awt.event.ActionEvent evt) {
29.
30.         ErrorAnneeLabel.setVisible(false);
31.         ErrorNoDataLabel.setVisible(false);
32.
33.         try
34.         {
35.             int annee = Integer.parseInt(AnneeTF.getText());
36.
37.             if (annee <= 1000 || 9999 <= annee)
38.             {
39.                 ErrorAnneeLabel.setVisible(true);
40.                 return;
41.             }
42.
43.             Utility.SendMsg(ProtocolePIDEP.GET_GR_COULEUR_COMP, AnneeTF.getText());
44.
45.             AnneeTF.setText("");
46.
47.             String reponse = Utility.ReceiveMsg();
48.             String[] parts = reponse.split("#");
49.
50.             if(parts[0].equals("NON"))
51.             {
52.                 ErrorNoDataLabel.setVisible(true);
53.                 return;
54.             }
55.         }
56.     }
57. }
```

```

53.         }
54.
55.         ObjectInputStream ois = new ObjectInputStream(ApplicationDataAnalysis.cl
iSock.getInputStream());
56.         HashMap<String, Object> map = (HashMap<String, Object>) ois.readObject()
;
57.         ShowBarChart(map);
58.     }
59.     catch(NumberFormatException ex)
60.     {
61.         ErrorAnneeLabel.setVisible(true);
62.     }
63.     catch (ClassNotFoundException ex)
64.     {
65.         System.err.println("GrCouleurRep : ClassNotFoundException : " + ex.getMe
ssage());
66.     }
67.     catch (IOException ex)
68.     {
69.         System.err.println("GrCouleurRep : IOException : " + ex.getMessage());
70.     }
71. }
72.
73. private void ShowBarChart(HashMap<String, Object> map)
74. {
75.     ArrayList<String> listDestinations = (ArrayList<String>)map.get("DESTINATION
S");
76.     ArrayList<Integer> listCount = (ArrayList<Integer>)map.get("COUNT");
77.     ArrayList<Integer> listTrimestres = (ArrayList<Integer>)map.get("TRIMESTRES"
);
78.
79.     DefaultCategoryDataset dcds = new DefaultCategoryDataset();
80.     for(int i = 0; i < listDestinations.size(); i++)
81.         dcds.setValue(listCount.get(i), listTrimestres.get(i), listDestinations.
get(i));
82.
83.     JFreeChart jfc = ChartFactory.createBarChart("R  partition du nombre de cont
ainers par destination par trimestre", "Destinations", "Occurences", dcds, PlotOrien
tation.VERTICAL, true, true, true);
84.     CategoryPlot plot = jfc.getCategoryPlot();
85.     NumberAxis rangeAxis = (NumberAxis)plot.getRangeAxis();
86.     rangeAxis.setTickUnit(new NumberTickUnit(1.0)); // Valeurs de l'axe par pas
de 1
87.     ChartPanel cp = new ChartPanel(jfc);
88.     JDialog dialog = new JDialog();
89.     dialog.setSize(500, 500);
90.     dialog.setContentPane(cp);
91.     dialog.setTitle("R  partition du nombre de containers par destination par tr
imestre");
92.     dialog.setVisible(true);
93. }
94.
95. private void MenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
96.     ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWin
dowAncestor(this);
97.     app.ChangePanel("Menu");
98. }
99. }

```

1.7 StatInferTestConf.java

```

1. package application_data_analysis;
2.
3. import javax.swing.SwingUtilities;
4.
5.
6. public class StatInferTestConf extends javax.swing.JPanel
7. {
8.     public StatInferTestConf()
9.     {
10.         initComponents();
11.         ErrorSaisieLabel.setVisible(false);
12.         pvalueLabel.setVisible(false);
13.         pvalueReponseLabel.setVisible(false);
14.         ResultatLabel.setVisible(false);
15.     }
16.
17.     private void RetourMenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
18.
19.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
20.         app.ChangePanel("Menu");
21.     }
22.
23.     private void TesterButtonActionPerformed(java.awt.event.ActionEvent evt) {
24.
25.         ErrorSaisieLabel.setVisible(false);
26.         pvalueLabel.setVisible(false);
27.         pvalueReponseLabel.setVisible(false);
28.         ResultatLabel.setVisible(false);
29.
30.         try
31.         {
32.             int nbContainers = Integer.parseInt(NbContainersTF.getText());
33.             if (nbContainers < 2)
34.             {
35.                 ErrorSaisieLabel.setVisible(true);
36.                 return;
37.             }
38.             Utility.SendMsg(ProtocolePIDEP.GET_STAT_INFER_TEST_CONF, NbContainersTF.getText());
39.
40.             // Réponse
41.             String reponse = Utility.ReceiveMsg();
42.             String[] parts = reponse.split("#");
43.
44.             if (!parts[0].equals("NON"))
45.             {
46.                 pvalueReponseLabel.setText(parts[0]);
47.                 pvalueReponseLabel.setVisible(true);
48.                 pvalueLabel.setVisible(true);
49.             }
50.             ResultatLabel.setText(parts[1]);
51.             ResultatLabel.setVisible(true);
52.         }
53.         catch (NumberFormatException ex)
54.         {
55.

```

```

56.         ErrorSaisieLabel.setVisible(true);
57.     }
58. }
59. }

```

1.8 StatInferTestHomog.java

```

1. package application_data_analysis;
2.
3. import javax.swing.SwingUtilities;
4.
5.
6. public class StatInferTestHomog extends javax.swing.JPanel
7. {
8.     public StatInferTestHomog()
9.     {
10.         initComponents();
11.         ErrorSaisieLabel.setVisible(false);
12.         pvalueLabel.setVisible(false);
13.         pvalueReponseLabel.setVisible(false);
14.         ResultatLabel.setVisible(false);
15.     }
16.
17.     private void RetourMenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
18.
19.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
20.         app.ChangePanel("Menu");
21.     }
22.     private void TesterButtonActionPerformed(java.awt.event.ActionEvent evt) {
23.
24.         ErrorSaisieLabel.setVisible(false);
25.         pvalueLabel.setVisible(false);
26.         pvalueReponseLabel.setVisible(false);
27.         ResultatLabel.setVisible(false);
28.
29.         try
30.         {
31.             int nbContainers = Integer.parseInt(NbContainersTF.getText());
32.             if (nbContainers <= 0)
33.             {
34.                 ErrorSaisieLabel.setText("Doit être un entier positif (>1) !");
35.                 ErrorSaisieLabel.setVisible(true);
36.                 return;
37.             }
38.
39.             if(DestinationATF.getText().isEmpty() || DestinationBTF.getText().isEmpty() || DestinationATF.getText().equals(DestinationBTF.getText()))
40.             {
41.                 ErrorSaisieLabel.setText("Entrer deux destinations différentes !");
42.                 ErrorSaisieLabel.setVisible(true);
43.                 return;
44.             }
45.

```

```

46.         String ChargeUtile = NbContainersTF.getText() + "#" + DestinationATF.get
      Text() + "#" + DestinationBTF.getText();
47.
48.         Utility.SendMsg(ProtocolePIDEP.GET_STAT_INFER_TEST_HOMOG, ChargeUtile);
49.
50.         // R ponse
51.         String reponse = Utility.ReceiveMsg();
52.         String[] parts = reponse.split("#");
53.
54.         if (!parts[0].equals("NON"))
55.         {
56.             pvalueReponseLabel.setText(parts[0]);
57.             pvalueReponseLabel.setVisible(true);
58.             pvalueLabel.setVisible(true);
59.         }
60.         ResultatLabel.setText(parts[1]);
61.         ResultatLabel.setVisible(true);
62.     }
63.     catch (NumberFormatException ex)
64.     {
65.         ErrorSaisieLabel.setText("Doit  tre un entier positif (>1) !");
66.         ErrorSaisieLabel.setVisible(true);
67.     }
68. }
69. }

```

1.9 StatInferTestAnova.java

```

1. package application_data_analysis;
2.
3. import javax.swing.SwingUtilities;
4.
5.
6. public class StatInferTestAnova extends javax.swing.JPanel
7. {
8.     public StatInferTestAnova()
9.     {
10.         initComponents();
11.         ErrorSaisieLabel.setVisible(false);
12.         pvalueLabel.setVisible(false);
13.         pvalueReponseLabel.setVisible(false);
14.         ResultatLabel.setVisible(false);
15.     }
16.
17.     private void RetourMenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
18.
19.         ApplicationDataAnalysis app = (ApplicationDataAnalysis)SwingUtilities.getWindowAncestor(this);
20.         app.ChangePanel("Menu");
21.     }
22.
23.     private void TesterButtonActionPerformed(java.awt.event.ActionEvent evt) {
24.
25.         ErrorSaisieLabel.setVisible(false);
26.         pvalueLabel.setVisible(false);
27.         pvalueReponseLabel.setVisible(false);

```



```

27.         ResultatLabel.setVisible(false);
28.
29.         try
30.         {
31.             int nbContainers = Integer.parseInt(NbContainersTF.getText());
32.             if (nbContainers < 2)
33.             {
34.                 ErrorSaisieLabel.setVisible(true);
35.                 return;
36.             }
37.
38.             Utility.SendMsg(ProtocolePIDEP.GET_STAT_INFER_TEST_ANOVA, NbContainersTF
39. .getText());
40.             // Réponse
41.             String reponse = Utility.ReceiveMsg();
42.             String[] parts = reponse.split("#");
43.
44.             if (!parts[0].equals("NON"))
45.             {
46.                 pvalueReponseLabel.setText(parts[0]);
47.                 pvalueReponseLabel.setVisible(true);
48.                 pvalueLabel.setVisible(true);
49.
50.             }
51.             ResultatLabel.setText(parts[1]);
52.             ResultatLabel.setVisible(true);
53.         }
54.         catch (NumberFormatException ex)
55.         {
56.             ErrorSaisieLabel.setVisible(true);
57.         }
58.     }
59. }

```

1.10 ProtocolePIDEP.java

```

1. package application_data_analysis;
2.
3.
4. public class ProtocolePIDEP
5. {
6.     public static final int LOGIN = 1;
7.     public static final int GET_STAT_DESCR_CONT = 2;
8.     public static final int GET_GR_COULEUR_REP = 3;
9.     public static final int GET_GR_COULEUR_COMP = 4;
10.    public static final int GET_STAT_INFER_TEST_CONF = 5;
11.    public static final int GET_STAT_INFER_TEST_HOMOG = 6;
12.    public static final int GET_STAT_INFER_TEST_ANOVA = 7;
13.    public static final int LOGOUT = 8;
14. }

```

1.11 Utility.java

```
1. package application_data_analysis;
2.
3. import java.io.*;
4. import java.net.*;
5. import java.util.Properties;
6.
7.
8. public final class Utility
9. {
10.     private static DataInputStream dis;
11.     public static DataOutputStream dos;
12.     private static String adresse;
13.     private static int port;
14.
15.
16.     public static void InitialisationFlux()
17.     {
18.         FichierProperties();
19.
20.         try
21.         {
22.             ApplicationDataAnalysis.cliSock = new Socket(adresse, port);
23.             dis = new DataInputStream(new BufferedInputStream(ApplicationDataAnalysis
24.             s.cliSock.getInputStream()));
25.             dos = new DataOutputStream(new BufferedOutputStream(ApplicationDataAnaly
26.             sis.cliSock.getOutputStream()));
27.         }
28.         catch (IOException e)
29.         {
30.             System.err.println("Utility : Erreur de cr ation de la socket, dis et d
31.             os (IO) : " + e);
32.         }
33.     }
34.
35.     private static void FichierProperties()
36.     {
37.         Properties prop = new Properties();
38.
39.         try
40.         {
41.             FileInputStream FIS = new FileInputStream("DataAnalysis.properties");
42.             prop.load(FIS);
43.         }
44.         catch(FileNotFoundException ex)
45.         {
46.             try
47.             {
48.                 FileOutputStream FOS = new FileOutputStream("DataAnalysis.properties
49.                 ");
50.
51.                 prop.setProperty("Adresse", "192.168.1.4");
52.                 prop.setProperty("Port", "31049");
53.
54.                 try
55.                 {
56.                     prop.store(FOS, null);
57.                 }
58.                 catch (IOException ex1)
59.                 {
60.                     System.err.println("Utility : Erreur de sauvegarde des propri t s
61.                     (IO) : " + ex1);
62.                 }
63.             }
64.             catch (IOException ex2)
65.             {
66.                 System.err.println("Utility : Erreur de cr ation du fichier de propri t s
67.                 (IO) : " + ex2);
68.             }
69.         }
70.     }
71. }
```

```

56.         System.err.println("Utility : Ecriture properties (IO) : " + ex1
.getMessage());
57.         System.exit(0);
58.     }
59. }
60.     catch (FileNotFoundException ex1)
61.     {
62.         System.err.println("Utility : Properties (FileNotFoundException) : "
+ ex1.getMessage());
63.         System.exit(0);
64.     }
65.
66.     }
67.     catch(IOException ex)
68.     {
69.         System.err.println("Utility : Lecture properties (IO) : " + ex.getMessag
e());
70.         System.exit(0);
71.     }
72.
73.     adresse = prop.getProperty("Adresse");
74.     port = Integer.parseInt(prop.getProperty("Port"));
75. }
76.
77. public static void SendMsg(int requete, String chargeUtile)
78. {
79.     chargeUtile = requete + "#" + chargeUtile;
80.     int taille = chargeUtile.length();
81.     String message = String.valueOf(taille) + "#" + chargeUtile;
82.
83.     try
84.     {
85.         dos.write(message.getBytes());
86.         dos.flush();
87.     }
88.     catch(IOException e)
89.     {
90.         System.err.println("Utility : Erreur d'envoi de msg (IO) : " + e);
91.     }
92. }
93.
94. public static String ReceiveMsg()
95. {
96.     byte b;
97.     StringBuffer taille = new StringBuffer();
98.     StringBuffer message = new StringBuffer();
99.
100.    try
101.    {
102.        while ((b = dis.readByte()) != (byte) '#')
103.        {
104.            if (b != (byte) '#')
105.                taille.append((char)b);
106.        }
107.
108.        for (int i = 0; i < Integer.parseInt(taille.toString()); i++)
109.        {
110.            b = dis.readByte();
111.            message.append((char)b);
112.        }
113.    }
114.    catch(IOException e)
115.    {

```

```
116.         System.err.println("Utility : Erreur de reception de msg (IO) : "  
    + e);  
117.     }  
118.  
119.     return message.toString();  
120. }  
121. }
```

2. Explications des 3 requêtes d'inférence statistique

2.1 Test d'hypothèse de conformité

Lors de cette requête, on a une hypothèse nulle qui, dans ce cas, suppose que le temps moyen de stationnement d'un container dans le parc est de 10 jours. Nous allons alors chercher un échantillon aléatoire de mouvements dans la base de données. Ainsi, on pourra instancier un TTest dont on va pouvoir exécuter la méthode

```
tTest(double mu, double[] echantillon)
```

Cette méthode va nous renvoyer une valeur appelée p-value. Il s'agit du pourcentage de risque de rejeter à tort l'hypothèse nulle.

Le test est bilatéral, c'est-à-dire qu'il importe peu de savoir si le paramètre est supérieur ou inférieur. Ce qui est important, c'est de savoir s'il diffère ou pas de la valeur supposée dans l'hypothèse nulle. Cela signifie également que la valeur critique correspond à une aire de rejet à gauche et à droite.

En partant du principe que le seuil de vérification est de 5%, la p-value peut être vérifiée. Le test étant bilatéral, la valeur de la p-value est testée en fonction de 2.5%.

Si celle-ci est inférieure à 0.025, cela signifie qu'il y a peu de chance d'avoir tort de rejeter l'hypothèse et donc on la rejette. On dit alors que l'échantillon n'est pas conforme à l'hypothèse nulle.

Par contre, si cette valeur est supérieure ou égale à 0.025, il y a trop de risque d'avoir tort de rejeter l'hypothèse. Autrement dit, on retient l'hypothèse et l'échantillon est dit conforme à l'hypothèse nulle.

2.2 Test d'hypothèse d'homogénéité

Lors de cette requête, on a une hypothèse nulle qui, dans ce cas, suppose que le temps moyen de stationnement d'un container pour une destination A est le même pour une destination B. Nous allons alors chercher deux échantillons aléatoires de mouvements dans la base de données, un pour la destination A et l'autre pour la destination B. Ainsi, on pourra instancier un TTest dont on va pouvoir exécuter la méthode

```
tTest(double[] echantillon1, double[] echantillon2)
```

Cette méthode va nous renvoyer une valeur appelée p-value. Il s'agit du pourcentage de risque de rejeter à tort l'hypothèse nulle.

Le test est bilatéral, c'est-à-dire qu'il importe peu de savoir si le paramètre est supérieur ou inférieur. Ce qui est important, c'est de savoir s'il diffère ou pas de la valeur supposée dans l'hypothèse nulle. Cela signifie également que la valeur critique correspond à une aire de rejet à gauche et à droite.

En partant du principe que le seuil de vérification est de 5%, la p-value peut être vérifiée. Le test étant bilatéral, la valeur de la p-value est testée en fonction de 2.5%.

Si celle-ci est inférieure à 0.025, cela signifie qu'il y a peu de chance d'avoir tort de rejeter l'hypothèse et donc on la rejette. On dit alors que l'échantillon n'est pas conforme à l'hypothèse nulle.

Par contre, si cette valeur est supérieure ou égale à 0.025, il y a trop de risque d'avoir tort de rejeter l'hypothèse. Autrement dit, on retient l'hypothèse et l'échantillon est dit conforme à l'hypothèse nulle.

2.3 Test d'hypothèse de type ANOVA

Lors de cette requête, on a une hypothèse nulle qui, dans ce cas, suppose que le temps moyen de stationnement d'un container est le même pour toutes les destinations. Nous allons alors chercher X échantillons aléatoires de mouvements dans la base de données, X étant le nombre de destinations possibles. Ainsi, on pourra instancier un `OneWayAnova` dont on va pouvoir exécuter la méthode

```
anovaPValue(Collection<double[]> collectionEchantillons)
```

Cette méthode va nous renvoyer une valeur appelée p-value. Il s'agit du pourcentage de risque de rejeter à tort l'hypothèse nulle.

Le test est bilatéral, c'est-à-dire qu'il importe peu de savoir si le paramètre est supérieur ou inférieur. Ce qui est important, c'est de savoir s'il diffère ou pas de la valeur supposée dans l'hypothèse nulle. Cela signifie également que la valeur critique correspond à une aire de rejet à gauche et à droite.

En partant du principe que le seuil de vérification est de 5%, la p-value peut être vérifiée. Le test étant bilatéral, la valeur de la p-value est testée en fonction de 2.5%.

Si celle-ci est inférieure à 0.025, cela signifie qu'il y a peu de chance d'avoir tort de rejeter l'hypothèse et donc on la rejette. On dit alors que l'échantillon n'est pas conforme à l'hypothèse nulle.

Par contre, si cette valeur est supérieure ou égale à 0.025, il y a trop de risque d'avoir tort de rejeter l'hypothèse. Autrement dit, on retient l'hypothèse et l'échantillon est dit conforme à l'hypothèse nulle.