|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Cryptography**

**“Euclid’s and Extended Euclid’s Algorithm”**

Abstract

Euclid’s Algorithm allows us to calculate de greatest common divisor between two numbers, and the extended version of this algorithm also allow us to express the gcd as a lineal combination of numbers. In this report I present a program that calculate gcd and give the numbers of the extended Euclidean algorithm.

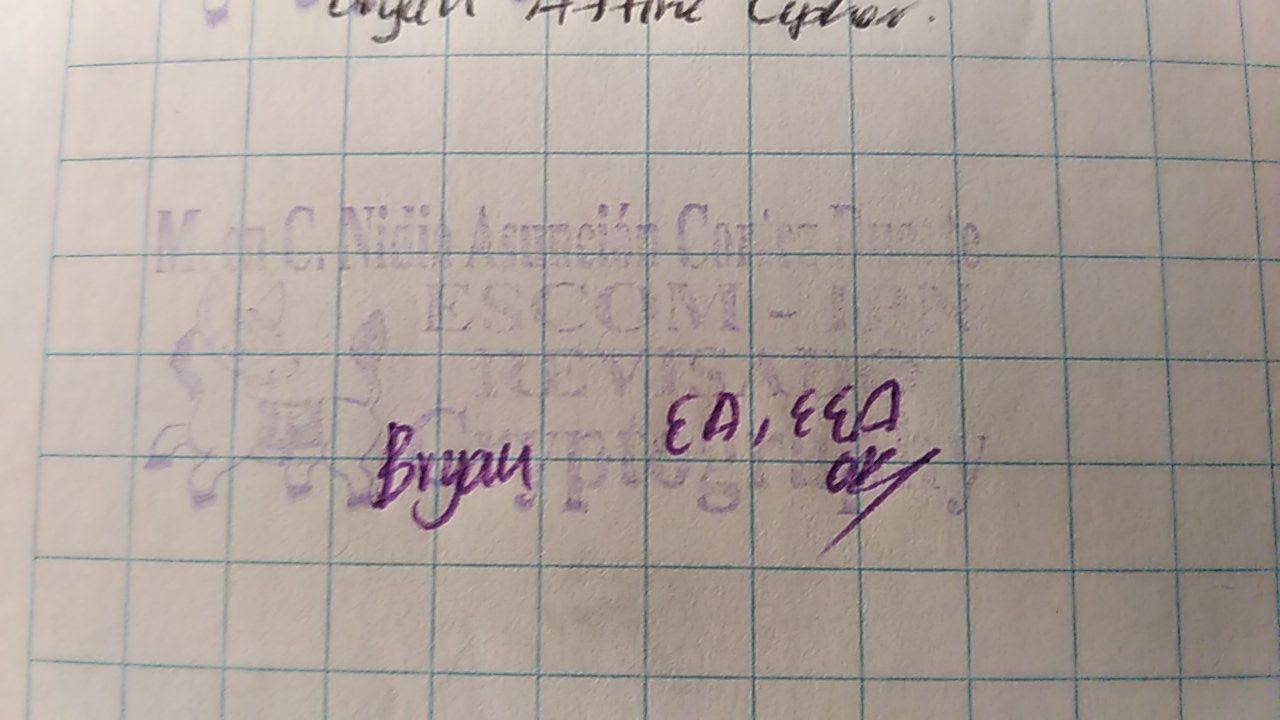
**By:**

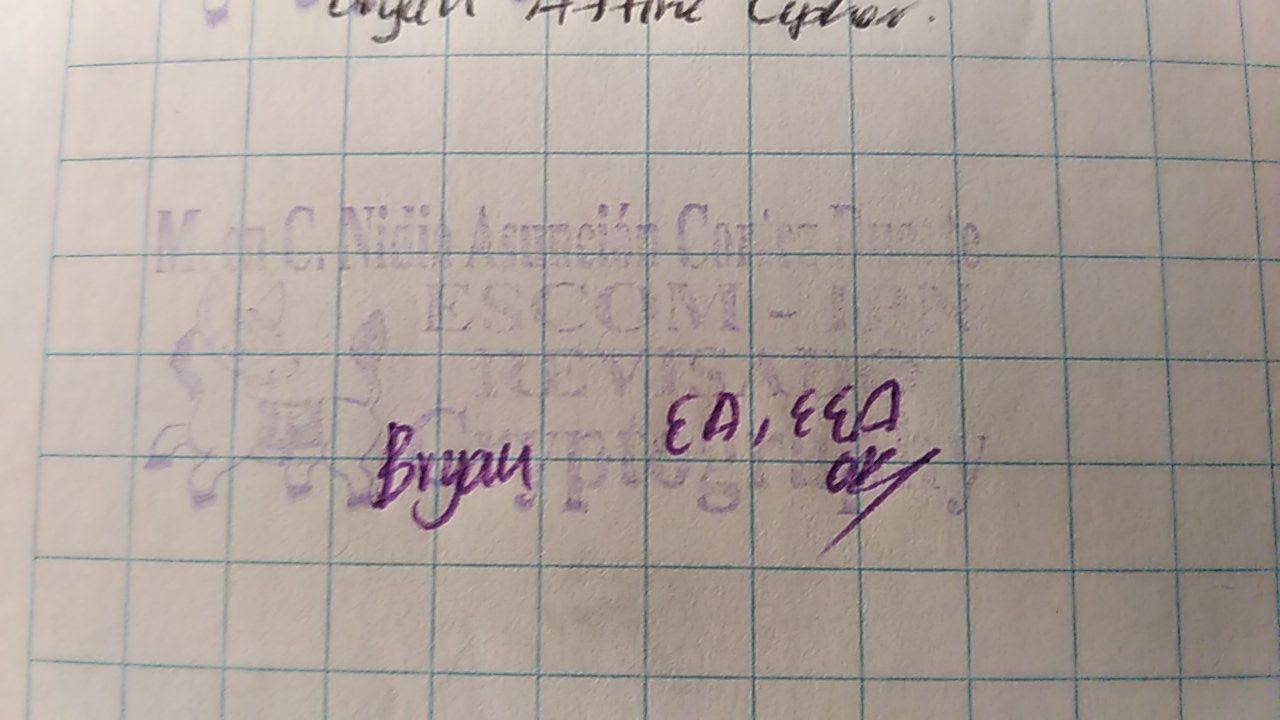
**Bryan Dominguez de la Rosa**

Professor:

MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

September 2018





**Index**

Contenido

[Introduction: 1](#_Toc525408952)

[Literature review: 1](#_Toc525408953)

[Software (libraries, packages, tools): 1](#_Toc525408954)

[Procedure: 2](#_Toc525408955)

[Results: 2](#_Toc525408956)

[Discussion: 4](#_Toc525408957)

[Conclusions: 4](#_Toc525408958)

[References: 4](#_Toc525408959)

[Code 4](#_Toc525408960)

# Introduction:

The Euclid’s algorithm is a classic and fundamental mathematical knowledge that allow us to find de greatest common divisor between two numbers. The extended Euclid’s algorithm express the gcd as a lineal combination. This two methods are used in the affine cipher with an alphabet of length N, because if the gcd(a,N) is different of 1, the cihper is not going to work correctly, and also we can find the decrypt formula with the Euclid’s extended algorithm.

# Literature review:

The cryptology is the science that deal with theoretical problems related with security in encrypted messages exchange from a sender to a receiver through a channel of communication (in informatic terms, this channel is usually a computer network). [1]

The greatest common divisor(g.c.d.) of the integers x and y is the largest integer d which divides both integers, denoted d = gcd(x, y).

The g.c.d. exists if at least one of the integers x and y is different of 0. Note that the g.c.d. is positive. (It’s often agreed, however, that gcd(0, 0) = 0) if gcd(x, y) = 1 then we say that x and y have no common divisors or that they are coprime. [2]

Using the Euclidean algorithm we can determine when a has an inverse modulo m by testing whether

gcd(a, m) = 1

But we still do not know how to determine the inverse when it exists. To do this we use a variant  
of Euclid’s gcd algorithm, called the extended Euclidean algorithm.

The extended Euclidean algorithm takes as input *a* and *b* and outputs *rm, sm* and *tm* such that  
*rm* = gcd(*a, b*) = *sma* + *tmb* [3]

# Software (libraries, packages, tools):

I use Java programing language to develop this practice, so the tools that I used were:

* NetBeans IDE 8.2
* Java Development Kit (jdk) 8
* All libraries used were of the Java Standard

# Procedure:

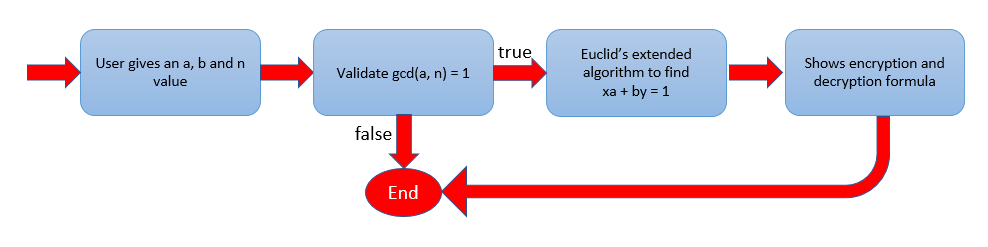


Fig. 1 Block diagram of the procedure of the program

The program displays an user interface that allows him to type the values of a, b and n. The program calculate gcd(a,n) and if it is different of one, the program shows a message saying that it’s not possible to calculate the formulas if gcd(a,n) is different of one.

If the values pass the the first validation, then the program calculate the Euclid’s extended algorithm to find x value and y value for

# Results:

The user interface is shown in Fig. 2. It is the main interface where the user type the values to use.

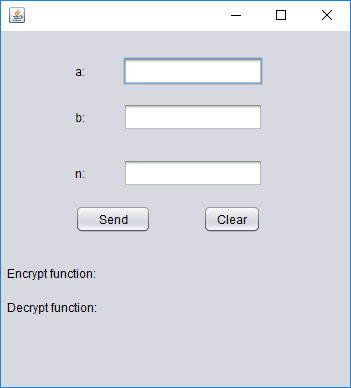


Fig. 2 The main interface of the program.

Once the user type the values required, the program is ready to start calculating, as can be seen is Fig 3.

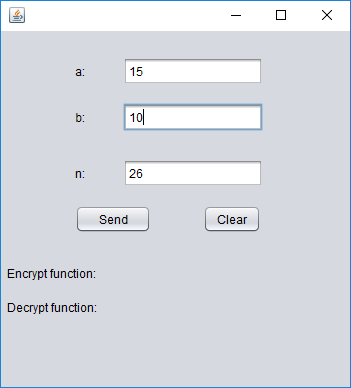


Fig. 3 How the program looks when it’s ready

When the user gives all the values and press the Send button, the program starts to calculate, as in Fig 4.

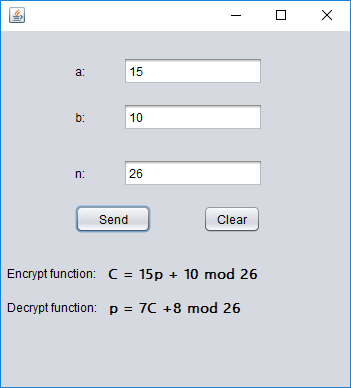


Fig. 4 Results given by the program

In the Fig. 4 we can see the two results that the program give us. The first one is the affine cipher encryption formula with the values given by the user, and in second place we have the affine cipher decryption formula that correspond to the given values.

# Discussion:

The validations and algorithms that this program present are really important to make the affine cipher work correctly, because if we don’t have a and n coprime, we are not going to be able to achieve a good performance in the encryption algorithm. And then the Euclid’s extended algorithm is an easy way to find which values we are going to use to decrypt the message ciphered by affine cipher.

# Conclusions:

This practice is a really interesting one, first because Euclid’s algorithm is a mathematical knowledge that every school teach, and the way to program it is very easy. And then if we talk about the extended algorithm, the level increases. One thing that could be done and I think it’s very simple, is to make an “animation” that prints every single operation of the two algorithms, in this way the program could be use to teach people how the algorithm really works, and to explain it step by step.

# References:

[1] S. Fernández, "La criptografía clásica", Sigma, no. 24, pp. 119-142, 2004.

[2] K. Ruohonen, Mathematical Cryptology. 2014.

[3] N. Smart, Cryptography. London: McGraw-Hill, 2003.

# Code

EuclidesValidation.java

1. **package** com.algorithm;
3. **public** **class** EuclidesValidation {
5. **private** **int** a;
6. **private** **int** b;
7. **private** **int** n;
8. **private** **double** inv\_a;
9. **private** **int** inv\_b;
11. **public** EuclidesValidation() {
12. }
14. **public** EuclidesValidation(**int** a, **int** b, **int** n) {
15. **this**.a = a;
16. **this**.b = b;
17. **this**.n = n;
18. **this**.inv\_b = (n - b);
19. }
21. **public** **int** getInvBetha(){
22. **return** **this**.inv\_b = (**int**) ((n-b)\*getInv\_a()%getN());
23. }
25. **public** **int** gdc() {
26. **int** aAux = getA();
27. **int** nAux = getN();
29. **while** (aAux != nAux) {
30. **if** (aAux < nAux) {
31. nAux = nAux - aAux;
32. } **else** {
33. aAux = aAux - nAux;
34. }
35. }
36. **return** (aAux);
37. }
39. **public** **void** euclidesExtendido() {
40. **double** aAux = getA();
41. **double** nAux = getN();
43. **double** x = 0, y = 0, d = 0;
44. **double** x2 = 1, x1 = 0, y2 = 0, y1 = 1;
45. **double** q = 0, r = 0;
47. **while** (nAux > 0) {
48. q = Math.floor(aAux / nAux);
49. r = aAux - q \* nAux;
50. x = x2 - q \* x1;
51. y = y2 - q \* y1;
52. aAux = nAux;
53. nAux = r;
54. x2 = x1;
55. x1 = x;
56. y2 = y1;
57. y1 = y;
58. }
59. inv\_a = x2;
60. }
62. **public** **int** getInv\_b() {
63. **return** (**this**.n - b);
64. }
66. **public** **int** getA() {
67. **return** a;
68. }
70. **public** **void** setA(**int** a) {
71. **this**.a = a;
72. }
74. **public** **int** getB() {
75. **return** b;
76. }
78. **public** **void** setB(**int** b) {
79. **this**.b = b;
80. }
82. **public** **double** getInv\_a() {
83. **return** inv\_a;
84. }
86. **public** **void** setInv\_a(**int** inv\_a) {
87. **this**.inv\_a = inv\_a;
88. }
90. **public** **int** getN() {
91. **return** n;
92. }
94. **public** **void** setN(**int** n) {
95. **this**.n = n;
96. }
98. }

MainFrame.java

1. **package** com.gui;
3. **import** com.algorithm.EuclidesValidation;
4. **import** javax.swing.JOptionPane;
6. /\*\*
7. \*
8. \* @author Bryan
9. \*/
10. **public** **class** MainFrame **extends** javax.swing.JFrame {
12. **public** MainFrame() {
13. setLocationRelativeTo(**null**);
14. initComponents();
15. }
17. /\*\*
18. \* This method is called from within the constructor to initialize the form.
19. \* WARNING: Do NOT modify this code. The content of this method is always
20. \* regenerated by the Form Editor.
21. \*/
22. @SuppressWarnings("unchecked")
23. // <editor-fold defaultstate="collapsed" desc="Generated Code">
24. **private** **void** initComponents() {
26. jLabel1 = **new** javax.swing.JLabel();
27. jLabel2 = **new** javax.swing.JLabel();
28. jLabel3 = **new** javax.swing.JLabel();
29. bTxt = **new** javax.swing.JTextField();
30. nTxt = **new** javax.swing.JTextField();
31. aTxt = **new** javax.swing.JTextField();
32. sendBtn = **new** javax.swing.JButton();
33. jLabel4 = **new** javax.swing.JLabel();
34. jLabel5 = **new** javax.swing.JLabel();
35. encryptLabel = **new** javax.swing.JLabel();
36. decryptLabel = **new** javax.swing.JLabel();
37. clearBtn = **new** javax.swing.JButton();
39. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);
41. jLabel1.setText("a:");
43. jLabel2.setText("b:");
45. jLabel3.setText("n:");
47. sendBtn.setText("Send");
48. sendBtn.addActionListener(**new** java.awt.event.ActionListener() {
49. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
50. sendBtnActionPerformed(evt);
51. }
52. });
54. jLabel4.setText("Encrypt function:");
56. jLabel5.setText("Decrypt function:");
58. encryptLabel.setFont(**new** java.awt.Font("Malgun Gothic Semilight", 1, 14)); // NOI18N
60. decryptLabel.setFont(**new** java.awt.Font("Malgun Gothic Semilight", 1, 14)); // NOI18N
62. clearBtn.setText("Clear");
63. clearBtn.addActionListener(**new** java.awt.event.ActionListener() {
64. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
65. clearBtnActionPerformed(evt);
66. }
67. });
69. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(getContentPane());
70. getContentPane().setLayout(layout);
71. layout.setHorizontalGroup(
72. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
73. .addGroup(layout.createSequentialGroup()
74. .addContainerGap(74, Short.MAX\_VALUE)
75. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
76. .addGroup(layout.createSequentialGroup()
77. .addComponent(sendBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 76, javax.swing.GroupLayout.PREFERRED\_SIZE)
78. .addGap(52, 52, 52)
79. .addComponent(clearBtn))
80. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
81. .addGroup(layout.createSequentialGroup()
82. .addComponent(jLabel3)
83. .addGap(38, 38, 38)
84. .addComponent(nTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, 140, javax.swing.GroupLayout.PREFERRED\_SIZE))
85. .addGroup(layout.createSequentialGroup()
86. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
87. .addComponent(jLabel2)
88. .addComponent(jLabel1))
89. .addGap(38, 38, 38)
90. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
91. .addComponent(aTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, 140, javax.swing.GroupLayout.PREFERRED\_SIZE)
92. .addComponent(bTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, 140, javax.swing.GroupLayout.PREFERRED\_SIZE)))))
93. .addGap(87, 87, 87))
94. .addGroup(layout.createSequentialGroup()
95. .addContainerGap()
96. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**)
97. .addGroup(layout.createSequentialGroup()
98. .addComponent(jLabel4)
99. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
100. .addComponent(encryptLabel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))
101. .addGroup(layout.createSequentialGroup()
102. .addComponent(jLabel5)
103. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
104. .addComponent(decryptLabel, javax.swing.GroupLayout.PREFERRED\_SIZE, 205, javax.swing.GroupLayout.PREFERRED\_SIZE)))
105. .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))
106. );
107. layout.setVerticalGroup(
108. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
109. .addGroup(layout.createSequentialGroup()
110. .addGap(26, 26, 26)
111. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
112. .addComponent(aTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
113. .addComponent(jLabel1))
114. .addGap(18, 18, 18)
115. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
116. .addComponent(bTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
117. .addComponent(jLabel2))
118. .addGap(28, 28, 28)
119. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
120. .addComponent(nTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)
121. .addComponent(jLabel3))
122. .addGap(18, 18, 18)
123. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
124. .addComponent(sendBtn)
125. .addComponent(clearBtn))
126. .addGap(32, 32, 32)
127. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**)
128. .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
129. .addComponent(encryptLabel, javax.swing.GroupLayout.PREFERRED\_SIZE, 16, javax.swing.GroupLayout.PREFERRED\_SIZE))
130. .addGap(18, 18, 18)
131. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**)
132. .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)
133. .addComponent(decryptLabel, javax.swing.GroupLayout.PREFERRED\_SIZE, 16, javax.swing.GroupLayout.PREFERRED\_SIZE))
134. .addContainerGap(72, Short.MAX\_VALUE))
135. );
137. pack();
138. }// </editor-fold>
140. **private** **void** sendBtnActionPerformed(java.awt.event.ActionEvent evt) {
141. **if** (!aTxt.getText().isEmpty() && !bTxt.getText().isEmpty() && !nTxt.getText().isEmpty()) {
142. **int** a = Integer.parseInt(aTxt.getText());
143. **int** b = Integer.parseInt(bTxt.getText());
144. **int** n = Integer.parseInt(nTxt.getText());
145. b = b % n;
146. EuclidesValidation ev = **new** EuclidesValidation(a, b, n);
147. **int** gcd;
148. **if** (n == 0) {
149. JOptionPane.showMessageDialog(**this**, "n cannot be 0",
150. "Info", JOptionPane.INFORMATION\_MESSAGE, **null**);
151. } **else** **if** (a > n) {
152. JOptionPane.showMessageDialog(**this**, "a must be less or equal than n",
153. "Info", JOptionPane.INFORMATION\_MESSAGE, **null**);
154. } **else** **if** ((gcd = ev.gdc()) != 1) {
155. JOptionPane.showMessageDialog(**this**, "Invalid a and n, gcd(a,n) is " + gcd
156. + ". To continue, gcd(a,n) must be 1", "Info", JOptionPane.INFORMATION\_MESSAGE, **null**);
157. } **else** {
158. ev.euclidesExtendido();
159. encryptLabel.setText("C = " + ev.getA() + "p + " + ev.getB() + " mod " + ev.getN());
160. decryptLabel.setText("p = " + (**int**) ev.getInv\_a() + "C +" + ev.getInvBetha() + " mod " + ev.getN());
161. }
162. } **else** {
163. JOptionPane.showMessageDialog(**this**,
164. "Type a value for a, n and b",
165. "Error", JOptionPane.ERROR\_MESSAGE, **null**);
166. }

169. }
171. **private** **void** clearBtnActionPerformed(java.awt.event.ActionEvent evt) {
172. aTxt.setText("");
173. bTxt.setText("");
174. nTxt.setText("");
175. encryptLabel.setText("");
176. decryptLabel.setText("");
177. }
179. /\*\*
180. \* @param args the command line arguments
181. \*/
182. **public** **static** **void** main(String args[]) {
183. /\* Set the Nimbus look and feel \*/
184. //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
185. /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
186. \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
187. \*/
188. **try** {
189. **for** (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
190. **if** ("Nimbus".equals(info.getName())) {
191. javax.swing.UIManager.setLookAndFeel(info.getClassName());
192. **break**;
193. }
194. }
195. } **catch** (ClassNotFoundException ex) {
196. java.util.logging.Logger.getLogger(MainFrame.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);
197. } **catch** (InstantiationException ex) {
198. java.util.logging.Logger.getLogger(MainFrame.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);
199. } **catch** (IllegalAccessException ex) {
200. java.util.logging.Logger.getLogger(MainFrame.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);
201. } **catch** (javax.swing.UnsupportedLookAndFeelException ex) {
202. java.util.logging.Logger.getLogger(MainFrame.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);
203. }
204. //</editor-fold>
206. /\* Create and display the form \*/
207. java.awt.EventQueue.invokeLater(**new** Runnable() {
208. **public** **void** run() {
209. **new** MainFrame().setVisible(**true**);
210. }
211. });
212. }
214. // Variables declaration - do not modify
215. **private** javax.swing.JTextField aTxt;
216. **private** javax.swing.JTextField bTxt;
217. **private** javax.swing.JButton clearBtn;
218. **private** javax.swing.JLabel decryptLabel;
219. **private** javax.swing.JLabel encryptLabel;
220. **private** javax.swing.JLabel jLabel1;
221. **private** javax.swing.JLabel jLabel2;
222. **private** javax.swing.JLabel jLabel3;
223. **private** javax.swing.JLabel jLabel4;
224. **private** javax.swing.JLabel jLabel5;
225. **private** javax.swing.JTextField nTxt;
226. **private** javax.swing.JButton sendBtn;
227. // End of variables declaration
228. }