

```

1  import java.io.*;
2  import java.net.*;
3  import java.util.Scanner;
4  import java.util.ArrayList;
5  import java.util.List;
6
7  public class Cliente {
8
9      public static void main(String[] args) throws Exception{
10         Socket client = new Socket("127.0.0.1", 6660);
11         Scanner in = new Scanner(System.in);
12
13         //declarações de variaveis
14         String usuario = null;
15         int d = 0;
16         int positivo = 0;
17         int ok = 0;
18         int dory = 1;
19         double resto = 0.0;
20
21
22         // declaração de ArrayList
23         List<Integer> vlr1 = new ArrayList<Integer>();
24         List<Integer> vlr2 = new ArrayList<Integer>();
25         List<Integer> vlr3 = new ArrayList<Integer>();
26         List<Integer> vlr4 = new ArrayList<Integer>();
27
28         System.out.println("Dollars Chat 2.0");
29         System.out.print("Informe seu nome de usuário: ");
30         usuario = in.nextLine();
31
32         //Armazenar a matriz.
33         //Abaixo, é criada um Array Bidimensional para armazenar nossa matriz.
34         int m[][] = new int[2][2];
35
36         // Lê os numero da seguinte forma:
37         //Quando o primeiro (for) roda o valor de "u" será 0;
38         // No segundo (for) o j tambem tera valor 0 no primeiro ciclo ou seja;
39         //O valor sera armazenado em m[0][0];
40         // Quando o segundo (for) rodar novamente sera em m[0][1];
41         //Voltando ao primeiro (for) tudo se repetira so que "u" valendo 1;
42         // Dai sera m[1][0] e m[1][1].
43         //Calculo da Determinante.
44         //Onde ao final, o valor da determinante ficará armazenada em "d".
45
46         System.out.print("Insira sua chave de 4 digitos: ");
47
48         for (int u = 0; u < 2; u++) {
49             for (int j = 0; j < 2; j++) {
50                 m[u][j] = in.nextInt();
51             }
52         }
53         d = ((m[0][0] * m[1][1]) - (m[0][1] * m[1][0]));
54         if (d == 0 || d % 2 == 0){
55             System.out.println("Matriz inválida");
56         }
57         else {
58             // verificação se matriz é válida.
59             // montar arreyes de deivisores da matriz.
60             if (m[0][0] < 0) {
61                 for (int i = 0; i >= m[0][0]; i--) {
62                     if (i <= -2) {
63                         resto = m[0][0] % i;
64                         if (resto == 0.0) {
65                             positivo = i * (-1);
66                             vlr1.add(positivo);
67                         }
68                     }
69                 }
70             }
71             } else {
72                 for (int i = 0; i <= m[0][0]; i++) {

```

```

74         if (i >= 2) {
75             resto = m[0][0] % i;
76             if (resto == 0.0) {
77                 vlr1.add(i);
78             }
79         }
80     }
81 }
82
83 if (m[0][1] < 0) {
84     for (int i = 0; i >= m[0][1]; i--) {
85         if (i <= -2) {
86             resto = m[0][1] % i;
87             if (resto == 0.0) {
88                 positivo = i * (-1);
89                 vlr2.add(positivo);
90             }
91         }
92     }
93
94 } else {
95     for (int i = 0; i <= m[0][1]; i++) {
96         if (i >= 2) {
97             resto = m[0][1] % i;
98             if (resto == 0.0) {
99                 vlr2.add(i);
100             }
101         }
102     }
103 }
104
105 if (m[1][0] < 0) {
106     for (int i = 0; i >= m[1][0]; i--) {
107         if (i <= -2) {
108             resto = m[1][0] % i;
109             if (resto == 0.0) {
110                 positivo = i * (-1);
111                 vlr3.add(positivo);
112             }
113         }
114     }
115
116 } else {
117     for (int i = 0; i <= m[1][0]; i++) {
118         if (i >= 2) {
119             resto = m[1][0] % i;
120             if (resto == 0.0) {
121                 vlr3.add(i);
122             }
123         }
124     }
125 }
126
127 if (m[1][1] < 0) {
128     for (int i = 0; i >= m[1][1]; i--) {
129         if (i <= -2) {
130             resto = m[1][1] % i;
131             if (resto == 0.0) {
132                 positivo = i * (-1);
133                 vlr4.add(positivo);
134             }
135         }
136     }
137 }
138
139 } else {
140     for (int i = 0; i <= m[1][1]; i++) {
141         if (i >= 2) {
142             resto = m[1][1] % i;
143             if (resto == 0.0) {
144                 vlr4.add(i);
145             }
146         }

```

```

147     }
148     }
149 }
150
151 // verificar divisores em comum
152 for (Integer i1 : vlr1) {
153     for (Integer i2 : vlr2) {
154         if (i1.equals(i2)) {
155             ok = 1;
156             break;
157         }
158     }
159
160     if (ok == 0) {
161         for (Integer i3 : vlr3) {
162             if (i1.equals(i3)) {
163                 ok = 1;
164                 break;
165             }
166         }
167     }
168     if (ok == 0) {
169         for (Integer i4 : vlr4) {
170             if (i1.equals(i4)) {
171                 ok = 1;
172                 break;
173             }
174         }
175     }
176 }
177
178 if (ok == 0) {
179
180     for (Integer i2 : vlr2) {
181         for (Integer i3 : vlr3) {
182             if (i2.equals(i3)) {
183                 ok = 1;
184                 break;
185             }
186         }
187
188         if (ok == 0) {
189             for (Integer i4 : vlr4) {
190                 if (i2.equals(i4)) {
191                     ok = 1;
192                     break;
193                 }
194             }
195         }
196     }
197 }
198
199 if (ok == 0) {
200     for (Integer i3 : vlr3) {
201         for (Integer i4 : vlr4) {
202             if (i3.equals(i4)) {
203                 ok = 1;
204                 break;
205             }
206         }
207     }
208 }
209 if (ok == 0) {
210     System.out.println("Matriz inválida. ");
211     in.close();
212 }
213 } // fim da validação.
214
215 if (ok != 0) {
216
217
218 while (dory == 1) {
219     System.out.printf("%s : ", usuario);

```

```

220 String mensagem = in.nextLine();
221
222
223 //Calculo da Determinante.
224 //Onde ao final, o valor da determinante ficará armazenada em "d".
225
226
227 //Comparação se a Matriz é valida.
228 //A condição para que exista a matriz inversa que será usada na
    descriptografiação é que;
229 //a determinante seja diferente de 0;
230
231 if (d != 0) {
232     String name = mensagem;
233
234
235     // Criar variavel contendo o tamanho do nome
236     int nameLenght = name.length();
237
238
239     // Caso o numero de letras seja impar, sera adicionado um ponto
    (.) ao final da frase.
240     if (nameLenght % 2 != 0) {
241         name = name + " ";
242         System.out.println(name);
243     }
244     //Atualizando valor da variavel tamanho de nome
245     nameLenght = name.length();
246     int loli[] = new int[nameLenght];
247
248     int cipher[] = new int[nameLenght];
249     //Separando os caracteres enquanto converte eles em
    valor numerico;
250     // Serão armazenados no Array loli[].
251     for(int w = 0; w < nameLenght ; w++){
252         char character = name.charAt(w);
253         int charValue = (int) character;
254         loli[w] = charValue;
255
256     }
257     //Cria o Array "cipher[]" para armazenar os numeros
    apos a criptografia.
258
259
260     for(int h = 0; h < nameLenght; h++) {
261         // Cifrando os numeros gerados anteriormente.
262
263
264         //São feitas as contas onde são utilizadas duas
    letras ao mesmo tempo na multiplicação das matrizes.
265         cipher[h] = (m[0][0]*loli[h]) + (m[0][1]*loli[h+1]);
266         cipher[h+1] = (m[1][0]*loli[h]) + (m[1][1]*loli[h+1]);
267
268         //Agora, comparamos se o valor esta dentro do
    intervalo 0 < cipher[h] < 256;
269         //Caso não esteja, fazemos o equivalente a uma
    multiplicação modular para descobrir sua posição;
270         //relativa dentro do intervalo.
271
272
273         //Caso seja maior que 256, atravez de subtrações
    sucessivas ele entrara no intervalo.
274         if (cipher[h] > 255) {
275             while (cipher[h] > 255) {
276                 cipher[h] = cipher[h] - 256;
277             }
278         }
279         //Caso seja menor que 0, atravez de somas sucessivas
    ele entrara no intervalo.
280         if (cipher[h] < 0) {
281             while(cipher[h] < 0) {
282                 cipher[h] = cipher[h] + 256;
283             }

```

```

284     }
285
286     //Acima, comparamos o cipher[h], porem trabalhamos
    com duas posições ao mesmo tempo;
    //No caso iremos tratar a posição cipher[h+1].
    h++;
287
288
289
290     //Caso seja maior que 256, atravez de subtrações
    sucessivas ele entrara no intervalo.
291     if (cipher[h] > 255) {
292         while (cipher[h] > 255) {
293             cipher[h] = cipher[h] - 256;
294         }
295     }
296
297     //Caso seja menor que 0, atravez de somas sucessivas
    ele entrara no intervalo.
298     if (cipher[h] < 0) {
299         while(cipher[h] < 0) {
300             cipher[h] = cipher[h] + 256;
301         }
302     }
303
304
305     }
306
307     byte[] bytesUsuario = usuario.getBytes("UTF8");
308
309     DataOutputStream saida = new
    DataOutputStream(client.getOutputStream());
    saida.writeInt(bytesUsuario.length);
    saida.write(bytesUsuario);
    saida.writeInt(nameLengt);
    for(int jabulani = 0; jabulani < nameLengt; jabulani++) {
310         saida.writeInt(cipher[jabulani]);
311     }
312
313
314
315
316
317
318     }else {
319
320     }
321
322     DataInputStream entrada = new
    DataInputStream(client.getInputStream());
    int tamanhoUsuario = entrada.readInt();
    byte[] byteUsuario = new byte[tamanhoUsuario];
    entrada.read(byteUsuario);
    int tamanhoMensagem = entrada.readInt();
    int cifrado[] = new int [tamanhoMensagem];
    for(int kiwi = 0; tamanhoMensagem > kiwi; kiwi++) {
323         cifrado[kiwi] = entrada.readInt();
324     }
325
326
327
328
329
330
331
332     //Modulo multiplicativo inverso
    int c = 0;
    int x = 0;
    int i = 1;
333
334
335
336
337     while (c == 0) {
338         x = ((256 * i) + 1)/ d;
339
340         if (d * x == (256 * i) + 1 ) {
341             c++;
342         }else {
343             i++;
344         }
345
346     }
347     while (x < 0) {
348         x = x + 256;
349     }
350
351

```

```

352 //Invertendo matriz
353 int inv[][] = new int [2][2];
354 inv[0][0] = m[1][1];
355 inv[0][1] = (m[0][1] * (-1));
356 inv[1][0] = (m[1][0] * (-1));
357 inv[1][1] = m[0][0];
358
359
360 int key[][] = new int [2][2];
361 key[0][0] = (inv[0][0] * x);
362 key[0][1] = (inv[0][1] * x);
363 key[1][0] = (inv[1][0] * x);
364 key[1][1] = (inv[1][1] * x);
365
366
367 int decifrada[] = new int [tamanhoMensagem];
368 String msg = "";
369
370 for(int pera = 0; pera < tamanhoMensagem; pera++) {
371     decifrada[pera] =
372         (key[0][0]*cifrado[pera])+(key[0][1]*cifrado[pera+1]);
373     decifrada[pera+1] =
374         (key[1][0]*cifrado[pera])+(key[1][1]*cifrado[pera+1]);
375
376     if (decifrada[pera] > 255) {
377         while (decifrada[pera] > 255) {
378             decifrada[pera] = decifrada[pera] - 256;
379         }
380     }
381
382     if (decifrada[pera] < 0) {
383         while(decifrada[pera] < 0) {
384             decifrada[pera] = decifrada[pera] + 256;
385         }
386     }
387     msg = msg +(char)decifrada[pera];
388     pera++;
389
390     if (decifrada[pera] > 255) {
391         while (decifrada[pera] > 255) {
392             decifrada[pera] = decifrada[pera] - 256;
393         }
394     }
395
396     if (decifrada[pera] < 0) {
397         while(decifrada[pera] < 0) {
398             decifrada[pera] = decifrada[pera] + 256;
399         }
400     }
401     msg = msg +(char)decifrada[pera];
402
403 }
404
405 String Usuario = new String(byteUsuario, "UTF-8");
406
407
408 System.out.printf("%s: ",Usuario);
409 System.out.println(msg);
410 }
411 }
412 in.close();
413 }
414 }
415 }
416

```