```java
import java.io.*;
import java.net.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;


public class Servidor {

    public static void main(String[] args) throws Exception{
        Scanner in = new Scanner(System.in);

        ServerSocket server = new ServerSocket(6660);
        Socket conexaoCliente = server.accept();

        //declarações de variaveis
        String usuario = null;
        int d = 0;
        int positivo = 0;
        int ok = 0;
        int dory = 1;
        double resto = 0.0;


        // declaração de ArrayList
        List<Integer> vlr1 = new ArrayList<Integer>();
        List<Integer> vlr2 = new ArrayList<Integer>();
        List<Integer> vlr3 = new ArrayList<Integer>();
        List<Integer> vlr4 = new ArrayList<Integer>();

        System.out.println("Dollars Chat 2.0");
        System.out.print("Informe seu nome de usuário: ");
         usuario = in.nextLine();

        //Armazenar a matriz.
        //Abaixo, é criada um Array Bidimensional para armazenar nossa matriz.
        int m[][] = new int[2][2];

        // Lê os numero da seguinte forma:
        //Quando o primeiro (for) roda o valor de "u" será 0;
        // No segundo (for) o j tambem tera valor 0 no primeiro ciclo ou seja;
        //O valor sera armazenado em m[0][0];
        // Quando o segundo (for) rodar novamente sera em m[0][1];
        //Voltando ao primeiro (for) tudo se repetira so que "u" valendo 1;
        // Dai sera m[1][0] e m[1][1].
        //Calculo da Determinante.
        //Onde ao final, o valor da determinante ficará armazenada em "d".

        System.out.print("Insira sua chave de 4 digitos: ");

        for (int u = 0; u < 2; u++) {
               for (int j = 0; j < 2; j++) {
                    m[u][j] = in.nextInt();
               }
           }
        d = ((m[0][0] * m[1][1]) - (m[0][1] * m[1][0]));
        if ( d == 0 ||  d % 2 == 0){
            System.out.println("Matriz inválida, determinante da matriz é 0"
                               + " " + "ou determinante é par.");
        }
        else {
            // verificação se matriz é válida.
            // montar arreys de deivisores da matriz.
            if (m[0][0] < 0) {
                for (int i = 0; i >= m[0][0]; i--) {
                    if (i <= -2) {
                        resto = m[0][0] % i;
                        if (resto == 0.0) {
                            positivo = i * (-1);
                            vlr1.add(positivo);
                        }
                    }
                }
```

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

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

```
219                 entrada.read(byteUsuario);
220                 int tamanhoMensagem = entrada.readInt();
221                 int cifrado[] = new int [tamanhoMensagem];
222                 for(int kiwi = 0; tamanhoMensagem > kiwi; kiwi++) {
223                     cifrado[kiwi] = entrada.readInt();
224                 }
225 
226 
227 
228               //Modulo multiplicativo inverso
229                 int c = 0;
230                 int x = 0;
231                 int i = 1;
232 
233                 while (c == 0) {
234                     x = ((256 * i) + 1)/ d;
235 
236                     if (d * x == (256 * i) + 1 ) {
237                         c++;
238                     }else {
239                         i++;
240                     }
241 
242                 }
243                 while (x < 0) {
244                     x = x + 256;
245                 }
246 
247 
248                 //Invertendo matriz
249                 int inv[][] = new int [2][2];
250                 inv[0][0] = m[1][1];
251                 inv[0][1] = (m[0][1] * (-1));
252                 inv[1][0] = (m[1][0] * (-1));
253                 inv[1][1] = m[0][0];
254 
255 
256                 int key[][] = new int [2][2];
257                 key[0][0] = (inv[0][0] * x);
258                 key[0][1] = (inv[0][1] * x);
259                 key[1][0] = (inv[1][0] * x);
260                 key[1][1] = (inv[1][1] * x);
261 
262 
263                 int decifrada[] = new int [tamanhoMensagem];
264                 String msg = "";
265 
266                 for(int pera = 0; pera < tamanhoMensagem; pera++) {
267                     decifrada[pera] =
                        (key[0][0]*cifrado[pera])+(key[0][1]*cifrado[pera+1]);
268                     decifrada[pera+1] =
                        (key[1][0]*cifrado[pera])+(key[1][1]*cifrado[pera+1]);
269 
270                     if (decifrada[pera] > 255) {
271                         while (decifrada[pera] > 255) {
272                             decifrada[pera] = decifrada[pera] - 256;
273                         }
274                     }
275 
276                     if (decifrada[pera] < 0) {
277                         while(decifrada[pera] < 0) {
278                             decifrada[pera] = decifrada[pera] + 256;
279                         }
280                     }
281                     msg = msg +(char)decifrada[pera];
282 
283                     pera++;
284 
285                     if (decifrada[pera] > 255) {
286                         while (decifrada[pera] > 255) {
287                             decifrada[pera] = decifrada[pera] - 256;
288                         }
289                     }
```

```
290
291                     if (decifrada[pera] < 0) {
292                         while(decifrada[pera] < 0) {
293                             decifrada[pera] = decifrada[pera] + 256;
294                             }
295                     }
296
297                     msg = msg +(char)decifrada[pera];
298
299                 }
300
301                 String Usuario = new String(byteUsuario, "UTF-8");
302
303
304                 System.out.printf("%s: ",Usuario);
305                 System.out.println(msg);
306
307                 //Mandar mesensagem!
308                 System.out.printf("%s : ",usuario);
309                 String mensagem = in.nextLine();
310
311                 //Comparação se a Matriz é valida.
312                 //A condição para que exista a matriz inversa que será usada na
                    descriptografação é que;
313                 //a determinante seja diferente de 0;
314
315                 if (d != 0) {
316                     String name = mensagem;
317
318
319                  // Criar variavel contendo o tamanho do nome
320                     int nameLenght = name.length();
321
322
323                  // Caso o numero de letras seja impar, sera adicionado um ponto (.) ao
                     final da frase.
324                     if (nameLenght % 2 != 0) {
325                         name = name + " ";
326
327                     }
328                  //Atualizando valor da variavel tamanho de nome
329                     nameLenght = name.length();
330                     int loli[] = new int[nameLenght];
331
332                     int cipher[] = new int[nameLenght];
333                  //Separando os caracteres enquanto converte eles em valor numerico;
334                     // Serão armazenados no Array loli[].
335                         for(int w = 0; w < nameLenght ; w++){
336                         char character = name.charAt(w);
337                         int charValue = (int) character;
338                         loli[w] = charValue;
339
340                     }
341                       //Cria o Array "chipher[] para armazenar os numeros apos a
                        criptografia.
342
343
344                     for(int h = 0; h < nameLenght; h++) {
345                 // Cifrando os numeros gerados anteriormente.
346
347
348                         //São feitas as contas onde são utilizadas duas letras ao mesmo
                        tempo na multiplicação das matrizes.
349                         cipher[h] = (m[0][0]*loli[h]) + (m[0][1]*loli[h+1]);
350                         cipher[h+1] = (m[1][0]*loli[h])+ (m[1][1]*loli[h+1]);
351
352                         //Agora, comparamos se o valor esta dentro do intervalo 0 <
                        cipher[h] < 256;
353                         //Caso não esteja, fazemos o equivalente a uma multiplicação
                        modular para descobrir sua posição;
354                         //relativa dentro do intervalo.
355
356
```

```
357                     //Caso seja maior que 256, atravez de subtrações sucessivas ele
                        entrara no intervalo.
358                     if (cipher[h] > 256) {
359                         while (cipher[h] > 256) {
360                             cipher[h] = cipher[h] - 256;
361                         }
362                     }
363                     //Caso seja menor que 0, atravez de somas sucessivas ele entrara
                        no intervalo.
364                     if (cipher[h] < 0) {
365                         while(cipher[h] < 0) {
366                             cipher[h] = cipher[h] + 256;
367                         }
368                     }
369                     //Acima, comparamos o cipher[h], porem trabalhamos com duas
                        posições ao mesmo tempo;
370                     //No caso iremos tratar a posição cipher[h+1].
371                     h++;
372 
373                     //Caso seja maior que 256, atravez de subtrações sucessivas ele
                        entrara no intervalo.
374                     if (cipher[h] > 256) {
375                         while (cipher[h] > 256) {
376                             cipher[h] = cipher[h] - 256;
377                         }
378                     }
379                     //Caso seja menor que 0, atravez de somas sucessivas ele entrara
                        no intervalo.
380                     if (cipher[h] < 0) {
381                         while(cipher[h] < 0) {
382                             cipher[h] = cipher[h] + 256;
383                         }
384                     }
385                 }
386 
387                 byte[] bytesUsuario = usuario.getBytes("UTF8");
388 
389                 DataOutputStream saida = new
                    DataOutputStream(conexaoCliente.getOutputStream());
390                 saida.writeInt(bytesUsuario.length);
391                 saida.write(bytesUsuario);
392                 saida.writeInt(nameLenght);
393                 for(int jabulani = 0; jabulani < nameLenght; jabulani++) {
394                     saida.writeInt(cipher[jabulani]);
395                 }
396 
397             }
398             }
399         }
400             in.close();
401     }
402 }
403 
```