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

public class Cliente {

    public static void main(String[] args) throws Exception{
        Socket client = new Socket("127.0.0.1", 6660);
        Scanner in = new Scanner(System.in);

        //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");
            }
            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);
                            }
                        }
                    }

                } else {

                    for (int i = 0; i <= m[0][0]; i++) {
```

```
                        if (i >= 2) {
                            resto = m[0][0] % i;
                            if (resto == 0.0) {
                                vlr1.add(i);
                            }
                        }
                    }
                }

                if (m[0][1] < 0) {
                    for (int i = 0; i >= m[0][1]; i--) {
                        if (i <= -2) {
                            resto = m[0][1] % i;
                            if (resto == 0.0) {
                                positivo = i * (-1);
                                vlr2.add(positivo);
                            }
                        }
                    }

                } else {
                    for (int i = 0; i <= m[0][1]; i++) {
                        if (i >= 2) {
                            resto = m[0][1] % i;
                            if (resto == 0.0) {
                                vlr2.add(i);
                            }
                        }
                    }
                }

                if (m[1][0] < 0) {
                    for (int i = 0; i >= m[1][0]; i--) {
                        if (i <= -2) {
                            resto = m[1][0] % i;
                            if (resto == 0.0) {
                                positivo = i * (-1);
                                vlr3.add(positivo);
                            }
                        }
                    }

                } else {

                    for (int i = 0; i <= m[1][0]; i++) {
                        if (i >= 2) {
                            resto = m[1][0] % i;
                            if (resto == 0.0) {
                                vlr3.add(i);
                            }
                        }
                    }
                }

                if (m[1][1] < 0) {
                    for (int i = 0; i >= m[1][1]; i--) {
                        if (i <= -2) {
                            resto = m[1][1] % i;
                            if (resto == 0.0) {
                                positivo = i * (-1);
                                vlr4.add(positivo);
                            }
                        }
                    }

                } else {

                    for (int i = 0; i <= m[1][1]; i++) {
                        if (i >= 2) {
                            resto = m[1][1] % i;
                            if (resto == 0.0) {
                                vlr4.add(i);
                            }
```

```
                    }
                }
            }

            // verificar divisores em comum
            for (Integer i1 : vlr1) {
                for (Integer i2 : vlr2) {
                    if (i1.equals(i2)) {
                        ok = 1;
                        break;
                    }
                }

                if (ok == 0) {
                    for (Integer i3 : vlr3) {
                        if (i1.equals(i3)) {
                            ok = 1;
                            break;
                        }
                    }
                }
                if (ok == 0) {
                    for (Integer i4 : vlr4) {
                        if (i1.equals(i4)) {
                            ok = 1;
                            break;
                        }
                    }
                }
            }

            if (ok == 0) {

                for (Integer i2 : vlr2) {
                    for (Integer i3 : vlr3) {
                        if (i2.equals(i3)) {
                            ok = 1;
                            break;
                        }
                    }

                    if (ok == 0) {
                        for (Integer i4 : vlr4) {
                            if (i2.equals(i4)) {
                                ok = 1;
                                break;
                            }
                        }
                    }
                }
            }

            if (ok == 0) {
                for (Integer i3 : vlr3) {
                    for (Integer i4 : vlr4) {
                        if (i3.equals(i4)) {
                            ok = 1;
                            break;
                        }
                    }
                }
            }
            if (ok == 0) {
                System.out.println("Matriz inválida. ");
                in.close();
            }
        } // fim da validação.

            if (ok != 0) {


            while (dory == 1) {
            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
                    descriptografação 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 "chipher[] 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                                     }
```

```
                                }

                                //Acima, comparamos o cipher[h], porem trabalhamos com duas posições ao mesmo tempo;
                                //No caso iremos tratar a posição cipher[h+1].
                                h++;

                                //Caso seja maior que 256, atravez de subtrações sucessivas ele entrara no intervalo.
                                if (cipher[h] > 255) {
                                    while (cipher[h] > 255) {
                                        cipher[h] = cipher[h] - 256;
                                    }
                                }

                                //Caso seja menor que 0, atravez de somas sucessivas ele entrara no intervalo.
                                if (cipher[h] < 0) {
                                    while(cipher[h] < 0) {
                                        cipher[h] = cipher[h] + 256;
                                    }
                                }


                            }

                            byte[] bytesUsuario = usuario.getBytes("UTF8");

                            DataOutputStream saida = new DataOutputStream(client.getOutputStream());
                            saida.writeInt(bytesUsuario.length);
                            saida.write(bytesUsuario);
                            saida.writeInt(nameLenght);
                            for(int jabulani = 0; jabulani < nameLenght; jabulani++) {
                                saida.writeInt(cipher[jabulani]);
                            }


                        }else {

                        }

                            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++) {
                                cifrado[kiwi] = entrada.readInt();
                            }

                           //Modulo multiplicativo inverso
                            int c = 0;
                            int x = 0;
                            int i = 1;

                            while (c == 0) {
                                x = ((256 * i) + 1)/ d;

                                if (d * x == (256 * i) + 1 ) {
                                    c++;
                                }else {
                                    i++;
                                }

                            }
                            while (x < 0) {
                                x = x + 256;
                            }
```

```
                        //Invertendo matriz
                        int inv[][] = new int [2][2];
                        inv[0][0] = m[1][1];
                        inv[0][1] = (m[0][1] * (-1));
                        inv[1][0] = (m[1][0] * (-1));
                        inv[1][1] = m[0][0];


                        int key[][] = new int [2][2];
                        key[0][0] = (inv[0][0] * x);
                        key[0][1] = (inv[0][1] * x);
                        key[1][0] = (inv[1][0] * x);
                        key[1][1] = (inv[1][1] * x);


                        int decifrada[] = new int [tamanhoMensagem];
                        String msg = "";

                        for(int pera = 0; pera < tamanhoMensagem; pera++) {
                            decifrada[pera] =
                            (key[0][0]*cifrado[pera])+(key[0][1]*cifrado[pera+1]);
                            decifrada[pera+1] =
                            (key[1][0]*cifrado[pera])+(key[1][1]*cifrado[pera+1]);

                            if (decifrada[pera] > 255) {
                                while (decifrada[pera] > 255) {
                                    decifrada[pera] = decifrada[pera] - 256;
                                }
                            }

                            if (decifrada[pera] < 0) {
                                while(decifrada[pera] < 0) {
                                    decifrada[pera] = decifrada[pera] + 256;
                                }
                            }
                            msg = msg +(char)decifrada[pera];

                            pera++;

                            if (decifrada[pera] > 255) {
                                while (decifrada[pera] > 255) {
                                    decifrada[pera] = decifrada[pera] - 256;
                                }
                            }

                            if (decifrada[pera] < 0) {
                                while(decifrada[pera] < 0) {
                                    decifrada[pera] = decifrada[pera] + 256;
                                    }
                            }

                            msg = msg +(char)decifrada[pera];

                        }

                        String Usuario = new String(byteUsuario, "UTF-8");


                        System.out.printf("%s: ",Usuario);
                        System.out.println(msg);
                }
            }
                    in.close();
        }

    }
```