Diffchecker

- 127 Removals + 107 Additions

2_1.java

```
1  import java.util.*;
2  import java.io.*;
3
4  public class Decrypt {
5
           public static void main (String args[]) throws IOExc
6  eption
7      {
8                  //final String key = args[0].toUpperCase;
9          //final File in = new File(args[1]);
10         //final String out = args[2];
11
12         final String key = "ABCDEFGH";
                                  // variable to store encrypti
   on key
           final File in = new File("input.txt");
13 // variable to store input file
           final String outFile = "output.txt";
14 // name of output file
15
16         final int SQUARE = 8;
17
18         int hash = key.hashCode();
                                // hashed key
19         Random random = new Random(hash);
                           // created random generator using hash
   ed value as seed
           Scanner scanner = new Scanner(in);
20 // created a scanner from the input file
21         String fileString
   ="";                                    // created a variable
    to hold the contents of the input file
           while(scanner.hasNext())
22 // appended every value from the scanner to the variable
23         {
24             fileString += scanner.nextLine();
25         }
26         fileString = fileString.toUpperCase();
            // converted the input text to uppercase
27
28
   String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ";        // c
   reated a variable to store the proper alphabet
           String origAlphabet = alphabet;
29 // made another variable to store the alphabet for later
           alphabet = alphabetEncryptor(hash, alphabet);
30 // replaced the alphabet with an encrypted version
31         //System.out.println(alphabet);
```

```
1  import java.util.*;
2  import java.io.*;
3
4  public class Decryptor {
5      int hash;
6      Random random;
7      Scanner scanner;
8      String fileString;
9
10     public Decryptor(){
11         final String key = "ABCDEFGH";
12         final File in = new File("input.txt");
13         final String outFile = "output.txt";
14
15         final int SQUARE = 8;
16
17         hash = key.hashCode();
18         random = new Random(hash);
19         scanner = new Scanner(in);
20         fileString = "";
21         while (scanner.hasNext())
22         {
23             fileString += scanner.nextLine();
24         }
25         fileString = fileString.toUpperCase();
26
27         String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ";
28         String origAlphabet = alphabet;
29         alphabet = alphabetEncryptor(hash, alphabet);
30
31         int[] pattern = pattern(hash, SQUARE);
```

```
32      int[] pattern = pattern(hash, SQUARE);
// generated the pattern used to read the array
33      //System.out.println(fileString);
34
35      String decryptedString = "";
                        // location to store the decrypte
d string
        String encryptedString = "";
36  // location to store the re-sorted string
37      while(fileString.length() > 0)
38      {
39          char[][] encryptedArray = encryptedArray(SQUARE,
 fileString, pattern);
// temporary array of 64 encrypted characters
40          if(fileString.length() >= 64)

// removes the 64 characters from the string

41          {
42              fileString =
fileString.substring((SQUARE*SQUARE), fileString.length());

43          }
44          else
                                                // if the
re is less than 64 characters left in the string
            {
45  // clear the string
46              fileString = "";
47          }
48          for(int i = 0; i < SQUARE; i++)
                    // for every row
49          {
50              for(int j = 0; j < SQUARE; j++)
                // for every column
51              {
52
encryptedString += encryptedArray[i][j];
      // add all of the sorted characters to the string
53              }
54          }
55      }
56      while(encryptedString.length() > 0)
                // while there are values to decrypt
57      {
            decryptedString += origAlphabet.charAt(alphabet.
58  indexOf(encryptedString.charAt(0)));
   // match the letter to the original and add it to the new
string
59
encryptedString = encryptedString.substring(1, encryptedStri
ng.length());
     // remove the first character of the encrypted string

60      }
```

```
32
33
34      String decryptedString = "";
35      String encryptedString = "";
36      while (fileString.length() > 0) {
37          char[][] encryptedArray = encryptArray(SQUARE,
 fileString, pattern);

38

39          if (fileString.length() >= 64)
40          {
41              fileString =
fileString.substring((SQUARE * SQUARE), fileString.length
());
42          } else
43          {

44              fileString = "";
45          }
46          for (int i = 0; i < SQUARE; i++)

47          {
48              for (int j = 0; j < SQUARE; j++)

49              {
50
encryptedString += encryptedArray[i][j];

51              }
52          }
53      }
54      while (encryptedString.length() > 0)

55      {
56          decryptedString += origAlphabet.charAt(alphabet.
indexOf(encryptedString.charAt(0)));

57

58

59
encryptedString = encryptedString.substring(1, encryptedStri
ng.length());
60

61      }
```

```
61          System.out.println(decryptedString);
62
63          PrintWriter out = new PrintWriter(outFile);
64          out.println(decryptedString);
65          out.close();
66
67      }
68
69      /**
70       * @param hash
71       * @param alphabet
72       * @return
73       */
74      public static String alphabetEncryptor(int hash, String alphabet)
75      {
76          Random random = new Random(hash);
77          for(int i = 0; i < 100; i++)
                            // swap two random letters in the alphabet 100 times
78          {
79              int indLetter1 = random.nextInt(27);
80              int indLetter2 = random.nextInt(27);
81              char letter1 = alphabet.charAt(indLetter1);
82              char letter2 = alphabet.charAt(indLetter2);
83              alphabet = alphabet.replace(letter1, '$');
84              alphabet = alphabet.replace(letter2, letter1);
85              alphabet = alphabet.replace('$', letter2);
86          }
87          return alphabet;
88      }
89
90      /**
91       * @param hash
92       * @param size
93       * @return
94       */
95      public static int[] pattern(int hash, int size)
96      {
97          int[] pattern = new int[size];
98          for(int i = 0; i < size; i++)
99          {
100             pattern[i] = i;
101         }
102         Random random = new Random(hash);
```

```
62          System.out.println(decryptedString);
63          PrintWriter out = new PrintWriter(outFile);
64          out.println(decryptedString);
65          out.close();
66      }
67
68      public static void main(String args[]) throws IOExceptio
n {
69          new Decryptor();
70      }
71
72      public String alphabetEncryptor(int hash, String alphabe
t) {
73          char[] chars = alphabet.toCharArray();
74          Random random = new Random(hash);
75          for (int i = 0; i < 100; i++)
76          {
77              int indLetter1 = random.nextInt(27);
78              int indLetter2 = random.nextInt(27);
79              char temp = chars[indLetter1];
80              chars[indLetter1] = chars[indLetter2];
81              chars[indLetter2] = temp;
82          }
83          return new String(chars);
84      }
85
86      public char[][] encryptArray(int SQUARE, String fileStri
ng, int[] pattern) {
87          char[][] arr = new char[SQUARE][SQUARE];
88
89          for (int j = 0; j < SQUARE; j++) {
90              for (int i = 0; i < SQUARE; i++) {
91                  if (fileString.length() == 0)
92                  {
93                      break;
94                  }
95                  arr[i][pattern[j]] = fileString.charAt(0);
96
97                  fileString = fileString.substring(1, fileStr
ing.length());
98              }
99          }
100         return arr;
101     }
102
103     public int[] pattern(int hash, int size) {
104         int[] pattern = new int[size];
105         int i = 0;
106         while (i < size){
107             pattern[i] = i++;
108         }
109         Random random = new Random(hash);
```

```
            // set the random generator
        for(int i = 0; i < 100; i++)
// swap two random numbers in the order 100 times
        {
            int indPattern1 = random.nextInt(8);
            int indPattern2 = random.nextInt(8);
            int tempPattern = pattern[indPattern1];
            pattern[indPattern1] = pattern[indPattern2];
            pattern[indPattern2] = tempPattern;
        }
        return pattern;
    }

    /**
     * @param SQUARE
     * @param fileString
     * @param pattern
     */
    public static char[][] encryptedArray(int SQUARE, String fileString, int[] pattern)
    {
        char[][] encryptedArray= new char[SQUARE][SQUARE]; // created a square 2d array to store the encrypted characters
        for(int j = 0; j < SQUARE; j++)
        {
            for(int i = 0; i < SQUARE; i++)
            {
                if(fileString.length() == 0)            // fill the array while there are values left in the string
                {
                    break;
                }
                encryptedArray[i][pattern[j]] = fileString.charAt(0);       // filled the array with the first character from the file
                fileString = fileString.substring(1, fileString.length());  // removed the character from the string
            }
        }
        return encryptedArray;
    }
}
```

```
        for (i = 0; i < 100; i++)
        {
            int indPattern1 = random.nextInt(8);
            int indPattern2 = random.nextInt(8);
            int tempPattern = pattern[indPattern1];
            pattern[indPattern1] = pattern[indPattern2];
            pattern[indPattern2] = tempPattern;
        }
        return pattern;
    }
}
```