Andrew Hayes
21321503
a.hayes18@nuigalway.ie

CT255 Assignment 3
Steganography

2022-11-13

1 Problem 1

1.1 Code

```
1  import java.io.BufferedReader;
2  import java.io.BufferedWriter;
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.util.regex.Pattern;
7
8  public class Stegano1
9  {
10     public static void main(String[] args) {
11         // Strings to hold the arguments. mode is either A or E for "Add" or "Extract".
12         String mode, inputfile, outputfile, bitstring;
13         Boolean err = false;                                                    // Boolean to tell
               if the arguments were passed correctly or not
14
15         if (args.length > 1) {                                                  // checking that at
               least one argument was passed to main
16             // assigning the mode & inputfile arguments
17             mode = args[0];
18             inputfile = args[1];
19
20             if (inputfile.equals("")) {                                         // checking that an
                   inputfile was provided (String was not empty)
21                 err = true;
22             }
23             else if ((mode.equals("A")) && (args.length > 3)){                  // checking if the
                   mode is "Add" & that the number of arguments provided was greater than 3
24                 // assigning the outputfile & bitstring arguments
25                 outputfile = args[2];
26                 bitstring = args[3];
27
28                 if (outputfile.equals("") || bitstring.equals("")) {            // checking that
                       neither the outputfile nor bitstring were empty strings
29                     err = true;
30                 }
31                 else {
32                     // hiding the bitstring
33                     hide(inputfile, outputfile, bitstring);
34                 }
35             }
36             else if (mode.equals("E")){                                         // checking if the
                   mode is "Extract"
37                 // retrieving (extracting) the bitstring from text
38                 retrieve(inputfile);
39             }
40             else {
41                 err = true;
42             }
43         }
44         else {
45             err = true;
46         }
47
48         if (err) {
49             System.out.println();
50             System.out.println("Use: Stegano1 <A:E><Input File><OutputFile><Bitstring>");
51             System.out.println("Example to add a bitvector to a file: Stegano1 A inp.txt out.txt
                   0010101");
52             System.out.println("Example to extract a bitvector from a file: Stegano1 E inp.txt");
53
54         }
55     }
56
57     // method to hide a bitstring in a copy the input file provided
58     static void hide(String inpFile, String outFile, String bitString) {
59         BufferedReader reader;                                                  // declaring a
               BufferedReader for the input file
60         BufferedWriter writer;                                                  // declaring a
               BufferedWriter for the output file
61
62         try {
63             // initialising the reader & writer to FileReaders of their respective files (inpFile &
                   outFile)
64             reader = new BufferedReader(new FileReader(inpFile));
65             writer = new BufferedWriter(new FileWriter(outFile));
66
```

```
            String line = reader.readLine();                                    // reading in the first line from the input file

            // will loop until there are no more lines to be read in from the input file (inpFile)
            while (line != null) {

                // if the bitString is not (yet) an empty String
                if (!bitString.equals("")) {

                    // if the first bit (char) of the bitString is 0
                    if (bitString.charAt(0) == '0') {                           // note: must use '' instead of "" for char literals
                        line = line.concat(" ");                                // concatenating a space to the end of the line (one space represents a 0)
                    }

                    // if the first bit of the bitString is 1
                    if (bitString.charAt(0) == '1') {
                        line = line.concat("  ");                               // concatenating two spaces to the end of the line (two spaces represents a 1)
                    }

                    // removing the first bit from the bitString now that it has been used
                    bitString = bitString.substring(1, bitString.length());    // replacing bitString with it's substring that goes from the second character to the last character
                }

                // writing the amended line to the output file
                writer.write(line);
                writer.newLine();

                // reading the next line
                line = reader.readLine();
            }

            // closing the reader & the writer
            reader.close();
            writer.close();
        }

        // catching any IOExceptions
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    // method to retrieve a hidden string from the input file provided
    static void retrieve(String inpFile) {
        BufferedReader reader;                                                  // declaring a BufferedReader for the input file (inpFile)
        String message = "";

        try {
            reader = new BufferedReader(new FileReader(inpFile));               // initialising the reader to a FileReader of the input file (inpFile)

            String line = reader.readLine();                                    // reading in the first line from the input file

            // will loop until there are no more lines to be read in from the input file
            while (line != null) {

                // checking if the line ends in a space using a regular expression
                if (Pattern.matches(".* $", line)) {                            // (checking if the String line contains any amount of any characters, followed by a space followed by the end of a line)

                    // checking if the line ends in two spaces using a regular expression
                    if (Pattern.matches(".*  $", line)) {                       // (checking if the String line contains any amount of any characters, followed by two spaces followed by the end of a line)
                        message = message.concat("1");                          // concatenating a "1" onto the message String (two spaces represent a "1")
                    }
                    else {                                                      // essentially, this "else" means "if the line ends with one space but not two"
                        message = message.concat("0");                          // concatenating a "0" onto the message String (one space represents a "0")
                    }
                }
                else {                                                          // if the String does not end in a space, then there is no (more) message to read
                    break;
                }

                // reading the next line
                line = reader.readLine();
            }

            // closing in the reader
            reader.close();
```

```
142
143             // checking if the message String is empty so that an error message can be printed if
                    no hidden message was found
144             if (message.equals("")) {
145                 message = "Error: No hidden message found!";
146             }
147
148             // printing out the message
149             System.out.println(message);
150         }
151         // catching any IOExceptions
152         catch (IOException e) {
153             e.printStackTrace();
154         }
155     }
156 }
```

1.2 Screenshot of Compilation & Output

```
[andrew@inspiron3501 CT255-Assignment-3]$ javac Stegano1.java && java Stegano1 A wby1.txt output.txt 101010
[andrew@inspiron3501 CT255-Assignment-3]$ java Stegano1 E output.txt
101010
[andrew@inspiron3501 CT255-Assignment-3]$
```

2 Problem 2

2.1 Code

```
1  import java.io.BufferedReader;
2  import java.io.BufferedWriter;
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.util.regex.Pattern;
7
8  public class Stegano1
9  {
10     public static void main(String[] args) {
11         // Strings to hold the arguments. mode is either A or E for "Add" or "Extract".
12         String mode, inputfile, outputfile, bitstring;
13         Boolean err = false;                                                    // Boolean to tell
                if the arguments were passed correctly or not
14
15         if (args.length > 1) {                                                  // checking that at
               least one argument was passed to main
16             // assigning the mode & inputfile arguments
17             mode = args[0];
18             inputfile = args[1];
19
20             if (inputfile.equals("")) {                                         // checking that an
                   inputfile was provided (String was not empty)
21                 err = true;
22             }
23             else if ((mode.equals("A")) && (args.length > 3)){                  // checking if the
                    mode is "Add" & that the number of arguments provided was greater than 3
24                 // assigning the outputfile & bitstring arguments
25                 outputfile = args[2];
26                 bitstring = args[3];
27
28                 if (outputfile.equals("") || bitstring.equals("")) {            // checking that
                        neither the outputfile nor bitstring were empty strings
29                     err = true;
30                 }
31                 else {
32                     // hiding the bitstring
33                     hide(inputfile, outputfile, bitstring);
34                 }
35             }
36             else if (mode.equals("E")){                                         // checking if the
                    mode is "Extract"
37                 // retrieving (extracting) the bitstring from text
38                 retrieve(inputfile);
39             }
40             else {
41                 err = true;
42             }
43         }
44         else {
45             err = true;
46         }
47
48         if (err) {
```

```
            System.out.println();
            System.out.println("Use: Stegano1 <A:E><Input File><OutputFile><Bitstring>");
            System.out.println("Example to add a bitvector to a file: Stegano1 A inp.txt out.txt 0010101");
            System.out.println("Example to extract a bitvector from a file: Stegano1 E inp.txt");

        }
    }

    // method to hide a bitstring in a copy the input file provided
    static void hide(String inpFile, String outFile, String bitString) {
        // to encode 2 bits with just one symbol, i'm going to represent the binary digits as an analog represenation of the number it represents plus one
        // e.g., 00 will be represented as " " (1 space), 01 as "  " (2 spaces), 10 as "   " (3 spaces), and 11 as "    " (4 spaces)
        // the two bits are treated as a binary number, and then i add one to said binary number to get the number of spaces that will represent that number

        BufferedReader reader;                                                  // declaring a BufferedReader for the input file
        BufferedWriter writer;                                                  // declaring a BufferedWriter for the output file

        try {
            // initialising the reader & writer to FileReaders of their respective files (inpFile & outFile)
            reader = new BufferedReader(new FileReader(inpFile));
            writer = new BufferedWriter(new FileWriter(outFile));

            String line = reader.readLine();                                    // reading in the first line from the input file

            // checking if the number of bits in the bitstring is uneven, and if so, adding a '0' onto the end
            if (bitString.length() % 2 != 0) { bitString = bitString.concat("0"); }

            // will loop until there are no more lines to be read in from the input file (inpFile)
            while (line != null) {

                // if the bitString is not (yet) an empty String
                if (!bitString.equals("")) {

                    // if the first 2-bit substring is 00, adding one space to the end of the line
                    if (bitString.substring(0,2).equals("00")) {
                        line = line.concat(" ");
                    }
                    // if the first 2-bit substring is 01, adding two spaces to the end of the line
                    else if (bitString.substring(0,2).equals("01")) {
                        line = line.concat("  ");
                    }
                    // if the first 2-bit substring is 10, adding three spaces to the end of the line
                    else if (bitString.substring(0,2).equals("10")) {
                        line = line.concat("   ");
                    }
                    // if the first 2-bit substring is 11, adding four spaces to the end of the line
                    else if (bitString.substring(0,2).equals("11")) {
                        line = line.concat("    ");
                    }
                    // removing the first two bits from the bitString now that they have been used
                    bitString = bitString.substring(2, bitString.length());     // replacing bitString with it's substring that goes from the third character to the last character
                }

                // writing the amended line to the output file
                writer.write(line);
                writer.newLine();

                // reading the next line
                line = reader.readLine();
            }

            // closing the reader & the writer
            reader.close();
            writer.close();
        }

        // catching any IOExceptions
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    // method to retrieve a hidden string from the input file provided
    static void retrieve(String inpFile) {
        BufferedReader reader;                                                  // declaring a BufferedReader for the input file (inpFile)
        String message = "";

        try {
```

```
127             reader = new BufferedReader(new FileReader(inpFile));                // initialising the
                      reader to a FileReader of the input file (inpFile)
128
129             String line = reader.readLine();                                     // reading in the
                     first line from the input file
130
131             // will loop until there are no more lines to be read in from the input file
132             while (line != null) {
133                 // checking if the line ends in a space using a regular expression
134                 if (Pattern.matches(".* $", line)) {                             // (checking if the
                       String line contains any amount of any characters, followed by a space
                      followed by the end of a line)
135
136                     if (Pattern.matches(".*    $", line)) {                     // checking if the
                         line ends in four spaces using a regular expression
137                         message = message.concat("11");                          // concatenating
                             "11" onto the end of the message String (four spaces represents "11")
138                     }
139                     else if (Pattern.matches(".*   $", line)) {                  // checking if the
                         line ends in three spaces using a regular expression
140                         message = message.concat("10");                          // concatenating
                             "10" onto the end of the message String (three spaces represents "10")
141                     }
142                     else if (Pattern.matches(".*  $", line)) {                   // (checking if the
                          String line contains any amount of any characters, followed by two spaces
                          followed by the end of a line)
143                         message = message.concat("01");                          // concatenating a
                             "1" onto the message String (two spaces represent a "1")
144                     }
145                     else {                                                       // essentially,
                         this "else" means "if the line ends with one space but not two"
146                         message = message.concat("00");                          // concatenating a
                             "0" onto the message String (one space represents a "0")
147                     }
148                 }
149                 else {                                                           // if the String
                      does not end in a space, then there is no (more) message to read
150                     break;
151                 }
152
153                 // reading the next line
154                 line = reader.readLine();
155             }
156
157             // closing in the reader
158             reader.close();
159
160             // checking if the message String is empty so that an error message can be printed if
                   no hidden message was found
161             if (message.equals("")) {
162                 message = "Error: No hidden message found!";
163             }
164
165             // printing out the message
166             System.out.println(message);
167         }
168         // catching any IOExceptions
169         catch (IOException e) {
170             e.printStackTrace();
171         }
172     }
173 }
```

2.2 Output

```
[andrew@inspiron3501 CT255-Assignment-3]$ javac Stegano1.java
[andrew@inspiron3501 CT255-Assignment-3]$ java Stegano1 A wby1.txt output.txt 00011011
[andrew@inspiron3501 CT255-Assignment-3]$ java Stegano1 E output.txt
00011011
```