CT255 Assignment 3

Steganography

Problem 1 Source Code

Hide Method

```
//binString index is declared before loop
  int i = 0;
  //While loop ensures that no more bits than the available lines are written
  while (line != null) {
      // Your code starts here
      //Ensures that the index variable doesnt exceed length of bit message
      if(i<binString.length()) {
          if (binString.charAt(i) == '0') {
              //Single space is concatenated at the end of the line
              line = line.concat(" ");
          } else if (binString.charAt(i) == '1') {
              //Two spaces are concatenated at the end of the line
              line = line.concat("  ");
          }
          i++;
      }
      // Store amended line in output file
      writer.write(line);
// read next line
line = reader.readLine();
      //If the next line is not null, print a newline "\n" to file for the next line to be amended
      if (line != null) {
          writer.newLine();
      }
  }
```

Retrieve Method

```
  while (line != null) {
      // Your code starts here
      if(line.length()>0) {
          //The last character in the line is checked
          if(line.charAt((line.length() - 1))==' ') {
              //The second last character in the line is checked
 //If the size of the line is one, there is no way there will be a second space present
//line.length() !=1 is used to ensure charAt doesn't check the position of a negative index
               if(line.length() !=1 && line.charAt((line.length() - 2))==' ') {
                  //If the last two characters are spaces, print a 1 to the screen
                  line = "1";
              }
              else
                  //Else if only the last character is a space, print a 0 to the screen
                  line = "0";
              //Line is changed to the bit in question and printed to the screen if there is a change.
              System.out.print(line);
          }
      }
// read next line
line = reader.readLine();
  }
```

<u>Problem 2 Source Code</u>

Hide Method

```
//binString index is declared before loop
  int i = 0;
  //Appending a 0 bit to the bitvector if the input bitvector length is odd
  if(binString.length()%2==1){
      binString = binString.concat("0");
  }
  //While loop ensures that no more bits than the available lines are written
  while (line != null) {
      //Ensures that the index variable doesnt exceed length of bit message
      if(i<binString.length()) {
          if (binString.charAt(i) == '0') {
              //Single space is concatenated at the end of the line
              line = line.concat(" ");
          }
          else if (binString.charAt(i) == '1') {
              //Two spaces are concatenated at the end of the line
              line = line.concat("  ");
          }
          //Move to the second bit
          i++;
          //First bit was hidden, now the second bit
          //The second bit will be hidden using the tab escape character "\t"
          if (binString.charAt(i) == '0') {
              //A hidden escape character tab will be appended
              line = line.concat("\t");
          }
          else if (binString.charAt(i) == '1') {
              //Two tabs are concatenated at the end of the line
              line = line.concat("\t\t");
          }
          //Proceed to the next pair of bits
          i++;
      }
      // Store amended line in output file
      writer.write(line);
// read next line
line = reader.readLine();
      //If the next line is not null, print a newline "\n" to file for the next line to
be amended
      if (line != null) {
          writer.newLine();
      }
```

Retrieve Method

```
//Keep looping until all lines in text file have been looped through
  while (line != null) {
      //First, the line is checked to see if it contains anything
      //and if the final character in the current line is a tab escape character
      //If there are secret bits hidden at the end of the line, the last character will
always be a tab character
      //This is done to prevent errors associated with using charAt()
      //If a tab can't be found, it continues looping to the next line without doing
any checks
      if(line.length()>0 && line.charAt((line.length())-1) == '\t') {
          for(int i = 4; i>1; i--){
              //This loop is used to search for space characters (which represent the
first secret bit)
              //A space character can exist within the 4th last character and the 2nd
last character
              //i.e. between two spaces followed by two tabs or one space followed by
one tab
if(line.charAt((line.length()) - i) == ' ') {
                  //If a space character is found, the character before it is also
checked
                  if (line.charAt((line.length()) - (i-1)) == ' '){
                      //If two spaces side by side are found, a 1 is printed to the
console
                      System.out.print("1");
                      break;
                  }
                  else
                      //Else, if only a single space is present, a 0 is printed and the
loop finishes
                      System.out.print("0");
                  break;
              }
          }
      //Looking for tab escape characters
          //This loop works similarly to the previous loop
          //Instead, only the last two characters of the line are checked
          //This is where the tabs will be present
          for(int i = 2; i>0; i--){
              if(line.charAt((line.length()) - i) == '\t') {
                  //If a tab is found, and is not located at the very end of the line,
                  //the next last character is checked to see if its also a tab escape
character.
                  if (i != 1 && line.charAt((line.length()) - (i-1)) == '\t'){
                      //If two tabs are side by side print 1 to the console
                      System.out.print("1");
                      break;
                  }
                  else
                      //Else, if only one tab is on its lonesome, it must be a 0
                      System.out.print("0");
                  break;
              }
          }
      }
// read next line
line = reader.readLine();
  }
```