/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Bailey Thompson

\* Encryption (1.1)

\* 16 September 2016

\* Info: This encryption program takes a user’s text, tells the user the text when it is encrypted, then returns the

\* Info: encrypted text to plain text. The plain text obviously has one part: the text. However, the encrypted text

\* Info: has four parts. The first part is the main encryption characters. The second part is the encryption code.

\* Info: The encryption code is randomly generated, but it is the same amount of characters as the plain text that

\* Info: the user inputs. When these numbers are generated, they are then added to the plain text to form the main

\* Info: encryption characters. The third part is the buffer, it is a set of random characters, and the buffer is

\* Info: of a random size. However, the buffer is determined by the fourth part: the buffer number. The buffer

\* Info: number is one character long, but it determines the amount of buffer characters. The whole point of the

\* Info: buffer is to slow down the process of determining the code, the buffer number is needed in order to

\* Info: determine the amount of buffer characters there are. The main encryption characters and encryption code

\* Info: work together just as the buffer and the buffer number work together. To keep adding security to the

\* Info: encryption code, more buffers and encryption codes can be added; however, it would add too much to the

\* Info: length of the encrypted text, and not offer as much protection as to justify the extra length.

\* Extra: Also, the reason why the encryption and decryption are separate private voids, and the only global variable

\* Extra: is the String word is to make it more realistic, as if the encryption and decryption were separate programs

\* Extra: operated by different people sending information back and forth.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

//declaring package

package encryption;

//declaring imports

import java.util.Arrays;

import javax.swing.JOptionPane;

//declaring public class

public class Encryption {

//declaring global variable

String word;

//declaring main method

public static void main(String[] args) {

//sending to method Encryption

Encryption Encryption = new Encryption();

Encryption.Encryption();

}

//declaring private void method

private void Encryption() {

//executes Encrypt method

Encrypt();

//executes Decrypt method

Decrypt();

}

//declaring private void method used for encryption

private void Encrypt() {

//declaring variable that checks if string is empty

int check;

//getting word that user wants to encrypt

word = JOptionPane.showInputDialog(null, "Write in text to be encrypted.", "Encryption Program", JOptionPane.PLAIN\_MESSAGE);

//if user inputs nothing and hits enter, user is notified to re-write the message

while ("".equals(word)) {

word = JOptionPane.showInputDialog(null, "I didn't quite get that!\nWrite in text to be encrypted.", "Encryption Program", JOptionPane.PLAIN\_MESSAGE);

}

//checking if string is empty

if (word == null) {

System.exit(0);

}

//declaring encryption variables

int length = word.length(), random;

//randomly generating a buffer number

int buffer = ((int) (Math.random() \* ((length - 0) + 0))) + 0;

//determining how big the array must be

char letter[] = new char[((length \* 2) + buffer + 1)];

//setting the buffer number

letter[((length \* 2) + buffer)] = (char) buffer;

//filling the word section and the code section

for (int i = 0; i < length; i++) {

random = ((int) (Math.random() \* ((500 - 20) + 20))) + 20;

letter[(i + length)] = (char) random;

letter[i] = (char) (word.charAt(i) + letter[(i + length)]);

}

//filling the buffer

for (int i = 0; i < buffer; i++) {

random = ((int) (Math.random() \* ((500 - 20) + 20))) + 20;

letter[(i + (length \* 2))] = (char) random;

}

//set char variables to string

word = String.valueOf(letter);

//set size of JOptionPane if text gets too long

if (length > 40) {

check = JOptionPane.showConfirmDialog(null, "This is the encrypted message:\n" + "<html><body width='1000'>" + word + "\nTo decrypt the message, press ok.", "Encryption Program", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

} else {

check = JOptionPane.showConfirmDialog(null, "This is the encrypted message:\n" + word + "\nTo decrypt the message, press ok.", "Encryption Program", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

}

//exits the game if cancel or exit is pressed

if (check != 0) {

System.exit(0);

}

}

//declaring private void method used for decryption

private void Decrypt() {

//declaring variable that checks if string is empty

int check;

//finding lenght of string

int length = word.length();

//redefine lenght after buffer and buffer number are taken out

length = length - ((int) word.charAt((length) - 1)) - 1;

//create array to size of string with buffer and buffer number taken out

char letter[] = new char[length];

//fill array with main characters and encryption characters

for (int i = 0; i < (length); i++) {

letter[i] = word.charAt(i);

}

//determine plain text by using main and encryption characters

for (int i = 0; i < (length / 2); i++) {

letter[i] -= letter[(i + (length / 2))];

}

//resize array in order to easier manipulate

letter = Arrays.copyOf(letter, (length / 2));

//set array to string

word = String.valueOf(letter);

//set size of JOptionPane if text gets too long

if (length > 400) {

check = JOptionPane.showConfirmDialog(null, "This is the decrypted message:\n" + "<html><body width='1000'>" + word, "Encryption Program", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

} else {

check = JOptionPane.showConfirmDialog(null, "This is the decrypted message:\n" + word, "Encryption Program", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

}

//exits the game if cancel or exit is pressed

if (check != 0) {

System.exit(0);

}

}

}