جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم هندسة البرمجيات

SWE314 PROJECT

Fall 2019

Cipher program

Section: *56739*	
Name	ID
Majd Bandar Bin Musibeh	438200829
Ghada Abdurrahman AlGhurairi	438201865
Shahad Omar AlHarbi	438201706
Lujain Abdullah Almuhaitheef	438201099
Rahaf Mohammed Al-Omar	438202048

Introduction:

Information nowadays become more valuable than ever, and as everything that is valuable it can be targeted. Information – sensitive information to be more specific- need to be secured in order to transmit it without it being exposed to unauthorized parties.

In our system (Whisper), we are encrypting texts to a ciphertext using our own modern block cipher and decrypt it to the original plaintext.

Whisper consist of two rounds, each have a s-box, p-box and rail fence techniques, after these two rounds it will use the result text from round 2 and convert it again then it will swap the converted text.

How can Whisper Encrypt a plaintext?

The plaintext should be either letters a-z (capital, small or both) , digits 0-9 and spaces . If the user entered some characters like (!@#$%^*(){} etc.) the system will not work (instead it will ask the user to enter a valid input consists of letters and digits). After that, the plaintext will go through the first round of encryption.

1- First, every letter and number in the plaintext should be converted to another letter or digit based on a special substitution boxes.

For letters:

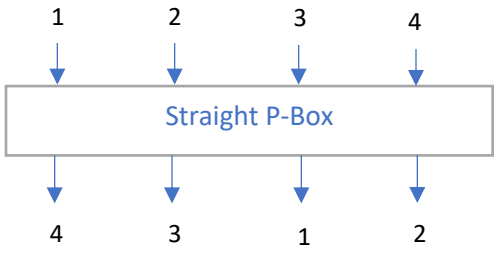
A → N	B → W	C → E	D → C
E → D	F → V	G → B	H → O
I → A	J → X	K → M	L → T
M → S	N → Z	O → J	P → K
Q → U	R → R	S → F	T → L
U → I	V → H	W → Q	X → P
Y → G	Z → Y		

For digits:

0 → 4	1 → 3	2 → 0	3 → 6	4 → 9
5 → 5	6 → 8	7 → 1	8 → 7	9 → 2

In case the user entered a space, the system will convert it to (~) and works with it as if its work with any letter.

2- Second, we will use a special straight p-box. We will divide the result of the substitution to four parts, if the input was just a one letter, we will add ~|} to make it four, if it two letters we will add ~| and if it three we will use ~. After the division, we will use the following p-box.



3- Finally in the round, we will do a simple rail fence and the width is 2.

Then, the second round is started by the same steps.

After the 2 rounds we have a more two steps.

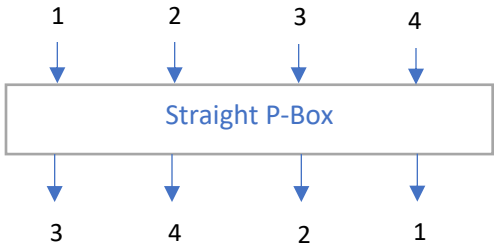
1- Convert the result of the second round to ASCII code (every character will take 3 digits after the converting).
2- The resulting ASCII codes, we will do a simple swap between each 2 digits.

Now we got the ciphertext!

<u>Now, how can Whisper Decrypt a ciphertext?</u>

First, we will check if the ciphertext is digits only, the number of digits is multiple of 12 (since we use 3 digits ASCII codes).

1- we will do a simple swap between each 2 digits.
2- Take every three digits and convert it to characters based on the ASCII code.
3- The round id started:
   a- Simple rail fence with width 2.
   b- The result is used in the next straight p-box after dividing the ciphertext to 4 parts as we mentioned.



   c- The last par in the round will be a substitution based on:
   For letters:

A → I	B → G	C → D	D → E
E → C	F → S	G → Y	H → V
I → U	J → O	K → P	L → T
M → K	N → A	O → H	P → X
Q → W	R → R	S → M	T → L
U → Q	V → F	W → B	X → J
Y → Z	Z → N		

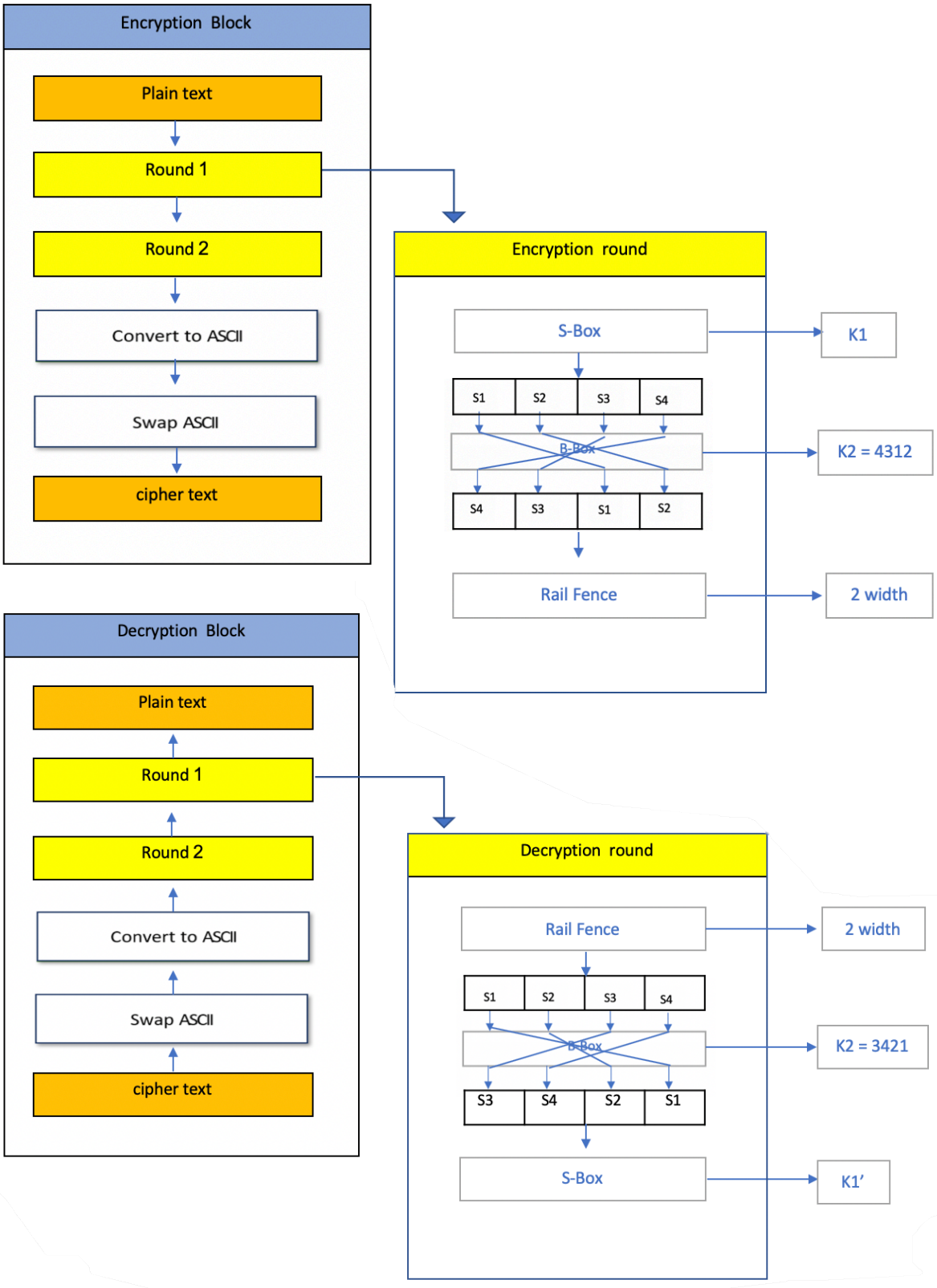
   For digits:

0 → 2	1 → 7	2 → 9	3 → 1	4 → 0
5 → 5	6 → 3	7 → 8	8 → 6	9 → 4

   If the character is (~) the system will convert it to a space.

The we will do another round by steps a-c, after finish this round we will get the plaintext!

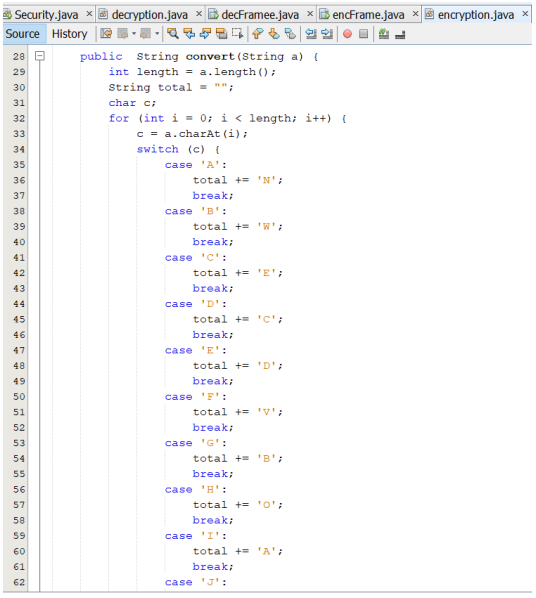
Whisper Model:



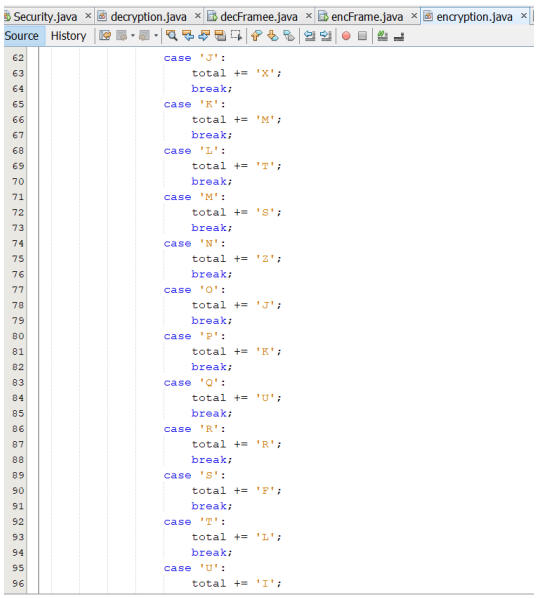
snaps of the source code:

Class encryption:

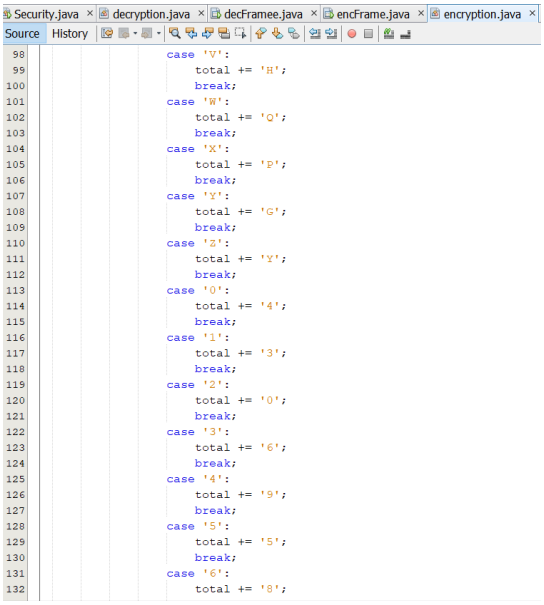
- The substitution method (convert):



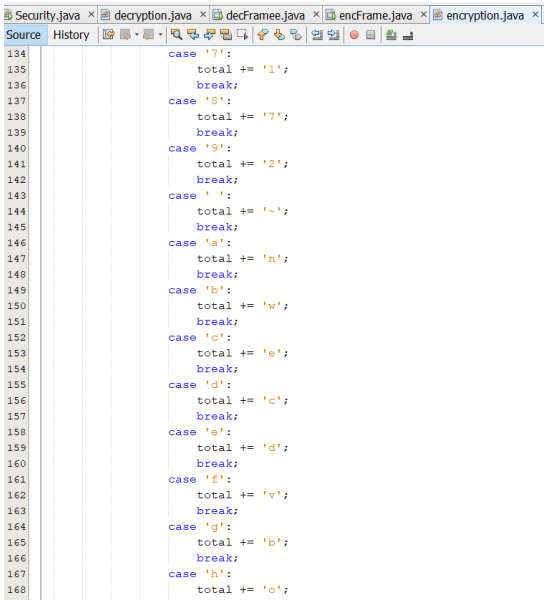

```
28      public  String convert(String a) {
29          int length = a.length();
30          String total = "";
31          char c;
32          for (int i = 0; i < length; i++) {
33              c = a.charAt(i);
34              switch (c) {
35                  case 'A':
36                      total += 'N';
37                      break;
38                  case 'B':
39                      total += 'W';
40                      break;
41                  case 'C':
42                      total += 'E';
43                      break;
44                  case 'D':
45                      total += 'C';
46                      break;
47                  case 'E':
48                      total += 'D';
49                      break;
50                  case 'F':
51                      total += 'V';
52                      break;
53                  case 'G':
54                      total += 'B';
55                      break;
56                  case 'H':
57                      total += 'O';
58                      break;
59                  case 'I':
60                      total += 'A';
61                      break;
62                  case 'J':
```




```
62                  case 'J':
63                      total += 'X';
64                      break;
65                  case 'K':
66                      total += 'M';
67                      break;
68                  case 'L':
69                      total += 'T';
70                      break;
71                  case 'M':
72                      total += 'S';
73                      break;
74                  case 'N':
75                      total += 'Z';
76                      break;
77                  case 'O':
78                      total += 'J';
79                      break;
80                  case 'P':
81                      total += 'K';
82                      break;
83                  case 'Q':
84                      total += 'U';
85                      break;
86                  case 'R':
87                      total += 'R';
88                      break;
89                  case 'S':
90                      total += 'F';
91                      break;
92                  case 'T':
93                      total += 'L';
94                      break;
95                  case 'U':
96                      total += 'I';
```

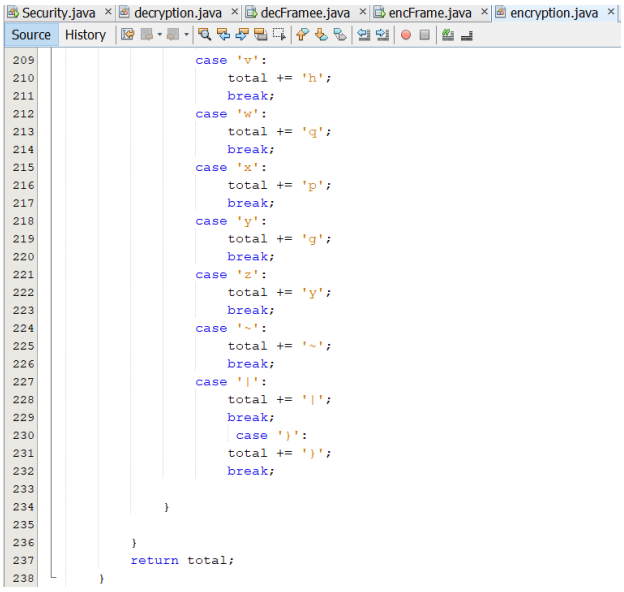



```
98                  case 'V':
99                      total += 'H';
100                     break;
101                 case 'W':
102                     total += 'Q';
103                     break;
104                 case 'X':
105                     total += 'P';
106                     break;
107                 case 'Y':
108                     total += 'G';
109                     break;
110                 case 'Z':
111                     total += 'Y';
112                     break;
113                 case '0':
114                     total += '4';
115                     break;
116                 case '1':
117                     total += '3';
118                     break;
119                 case '2':
120                     total += '0';
121                     break;
122                 case '3':
123                     total += '6';
124                     break;
125                 case '4':
126                     total += '9';
127                     break;
128                 case '5':
129                     total += '5';
130                     break;
131                 case '6':
132                     total += '8';
```



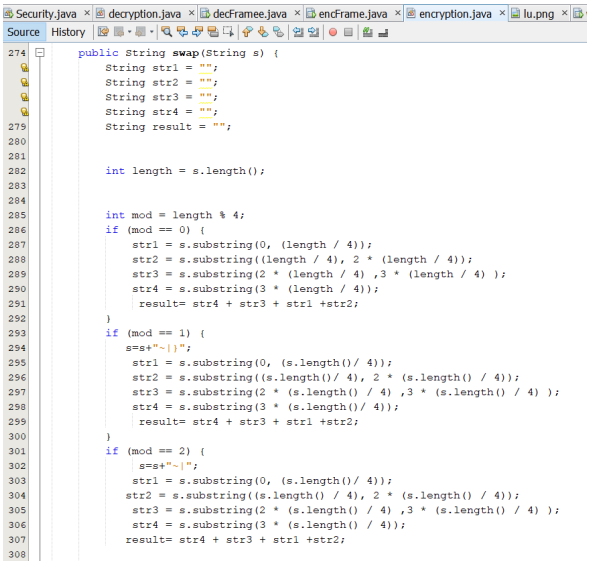

```
134                 case '7':
135                     total += '1';
136                     break;
137                 case '8':
138                     total += '7';
139                     break;
140                 case '9':
141                     total += '2';
142                     break;
143                 case ' ':
144                     total += '~';
145                     break;
146                 case 'a':
147                     total += 'n';
148                     break;
149                 case 'b':
150                     total += 'w';
151                     break;
152                 case 'c':
153                     total += 'e';
154                     break;
155                 case 'd':
156                     total += 'c';
157                     break;
158                 case 'e':
159                     total += 'd';
160                     break;
161                 case 'f':
162                     total += 'v';
163                     break;
164                 case 'g':
165                     total += 'b';
166                     break;
167                 case 'h':
168                     total += 'o';
```


```
                case 'j':
                    total += 'x';
                    break;
                case 'k':
                    total += 'm';
                    break;
                case 'l':
                    total += 't';
                    break;
                case 'm':
                    total += 's';
                    break;
                case 'n':
                    total += 'z';
                    break;
                case 'o':
                    total += 'j';
                    break;
                case 'p':
                    total += 'k';
                    break;
                case 'q':
                    total += 'u';
                    break;
                case 'r':
                    total += 'r';
                    break;
                case 's':
                    total += 'f';
                    break;
                case 't':
                    total += 'l';
                    break;
                case 'u':
                    total += 'i';
```

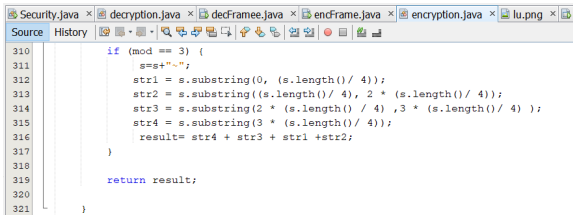



```
                case 'v':
                    total += 'h';
                    break;
                case 'w':
                    total += 'q';
                    break;
                case 'x':
                    total += 'p';
                    break;
                case 'y':
                    total += 'g';
                    break;
                case 'z':
                    total += 'y';
                    break;
                case '~':
                    total += '~';
                    break;
                case '|':
                    total += '|';
                    break;
                    case '}':
                    total += ')';
                    break;

            }

        }
        return total;
    }
```

- The permutation method (swap):

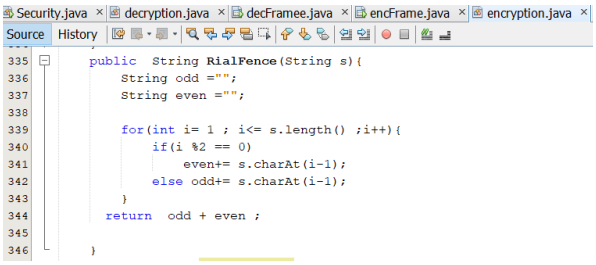



```
    public String swap(String s) {
        String str1 = "";
        String str2 = "";
        String str3 = "";
        String str4 = "";
        String result = "";


        int length = s.length();


        int mod = length % 4;
        if (mod == 0) {
            str1 = s.substring(0, (length / 4));
            str2 = s.substring((length / 4), 2 * (length / 4));
            str3 = s.substring(2 * (length / 4) ,3 * (length / 4) );
            str4 = s.substring(3 * (length / 4));
             result= str4 + str3 + str1 +str2;
        }
        if (mod == 1) {
           s=s+"~|)";
            str1 = s.substring(0, (s.length()/ 4));
            str2 = s.substring((s.length()/ 4), 2 * (s.length() / 4));
            str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length() / 4) );
            str4 = s.substring(3 * (s.length()/ 4));
             result= str4 + str3 + str1 +str2;
        }
        if (mod == 2) {
             s=s+"~|";
            str1 = s.substring(0, (s.length()/ 4));
           str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
            str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length() / 4) );
            str4 = s.substring(3 * (s.length() / 4));
           result= str4 + str3 + str1 +str2;
```



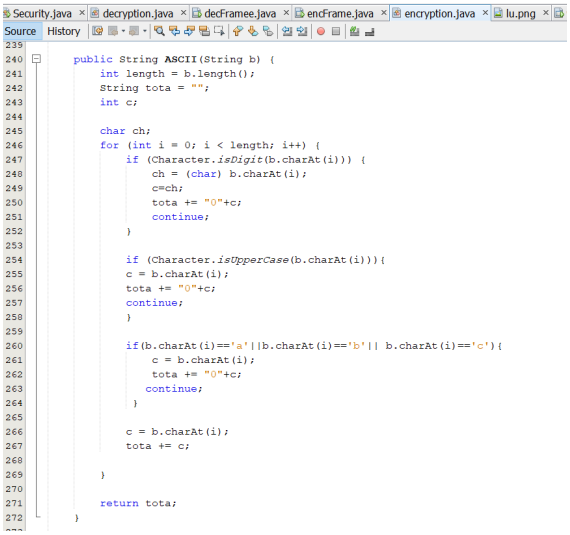

```
        if (mod == 3) {
             s=s+"~";
            str1 = s.substring(0, (s.length()/ 4));
            str2 = s.substring((s.length()/ 4), 2 * (s.length()/ 4));
            str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length()/ 4) );
            str4 = s.substring(3 * (s.length()/ 4));
             result= str4 + str3 + str1 +str2;
        }

        return result;

    }
```

- Rail Fence method (Railfence):




```
335     public  String RialFence(String s){
336         String odd ="";
337         String even ="";
338
339         for(int i= 1 ; i<= s.length() ;i++){
340             if(i %2 == 0)
341                 even+= s.charAt(i-1);
342             else odd+= s.charAt(i-1);
343         }
344        return  odd + even ;
345
346     }
```

- Converting to ASCII method (ASCII):

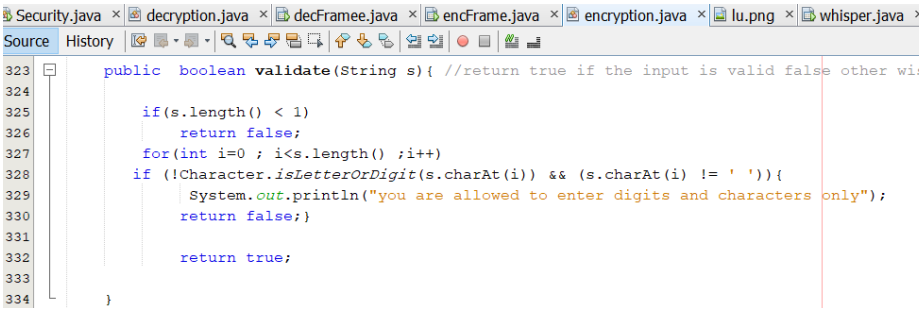



```
239
240     public String ASCII(String b) {
241         int length = b.length();
242         String tota = "";
243         int c;
244
245         char ch;
246         for (int i = 0; i < length; i++) {
247             if (Character.isDigit(b.charAt(i))) {
248                 ch = (char) b.charAt(i);
249                 c=ch;
250                 tota += "0"+c;
251                 continue;
252             }
253
254             if (Character.isUpperCase(b.charAt(i))){
255             c = b.charAt(i);
256             tota += "0"+c;
257             continue;
258             }
259
260             if(b.charAt(i)=='a'||b.charAt(i)=='b'|| b.charAt(i)=='c'){
261                 c = b.charAt(i);
262                 tota += "0"+c;
263                continue;
264              }
265
266             c = b.charAt(i);
267             tota += c;
268
269         }
270
271         return tota;
272     }
```

- Swap ASCII method (convert Ascii):

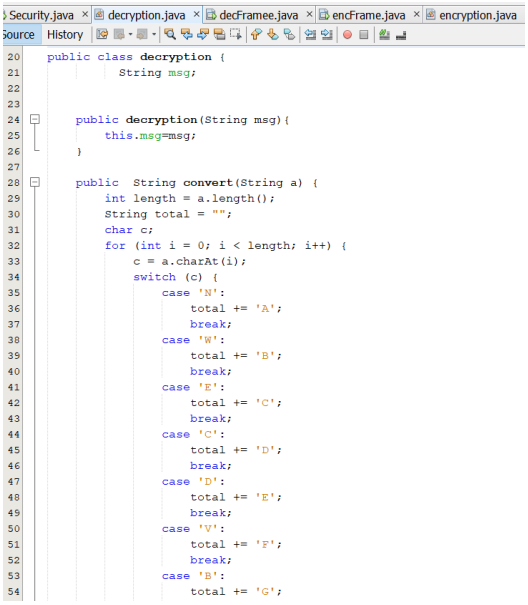
```
public String convertAscii(String s){

    int length = s.length();
    char[] array = s.toCharArray();
    String total="";
    int j=1;
    int k = 0;
        for(; j+2<=length && k+2<=length;j+=2,k+=2){
        total=total+""+ array[j]+""+array[k];
       }
        total=total+""+ array[j]+""+array[k];

return total;
}
```

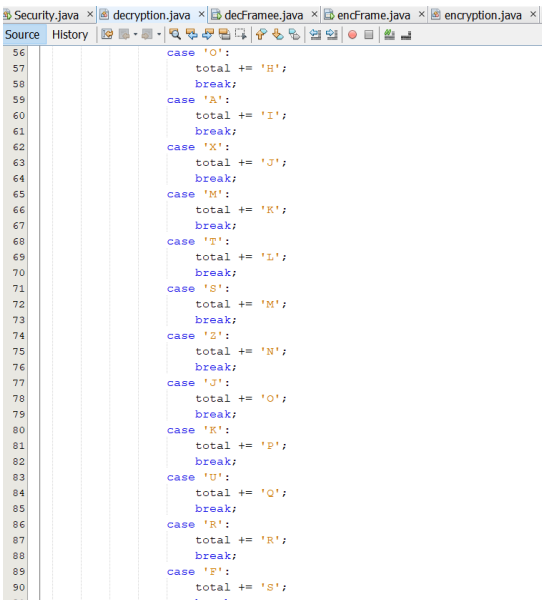
- Method (validate):

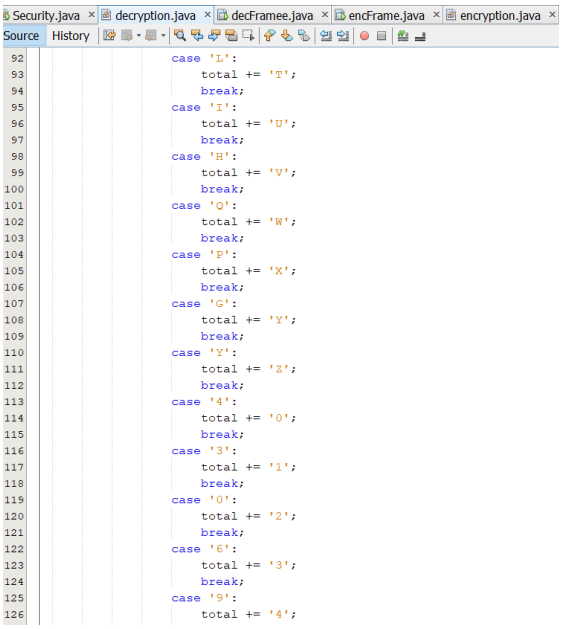


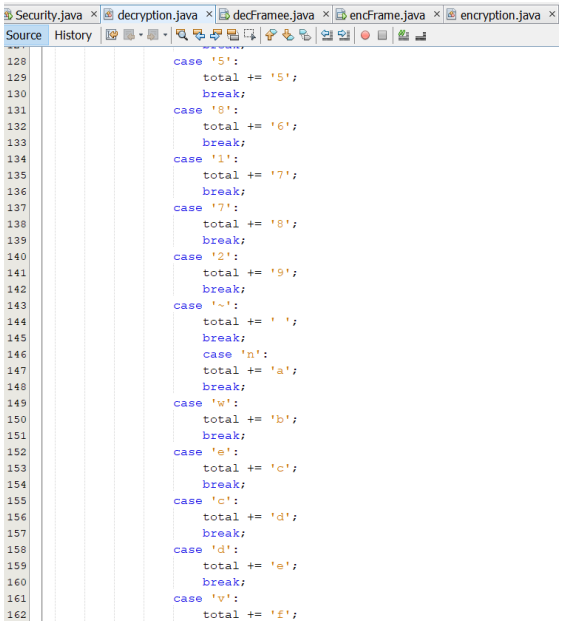
Class decryption:

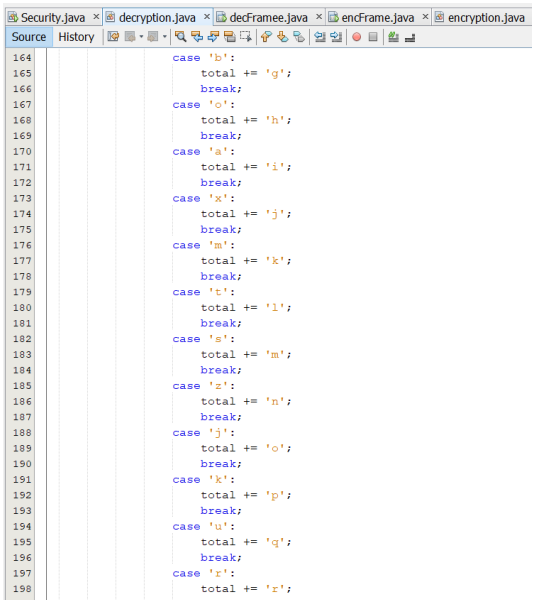
- The substitution method (convert):

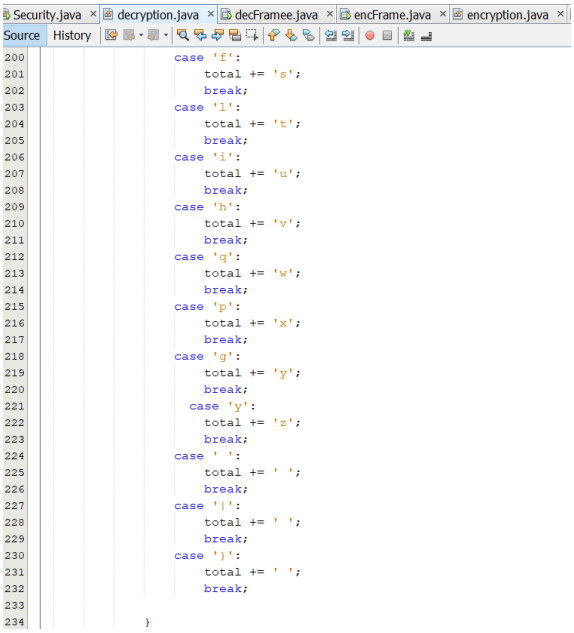




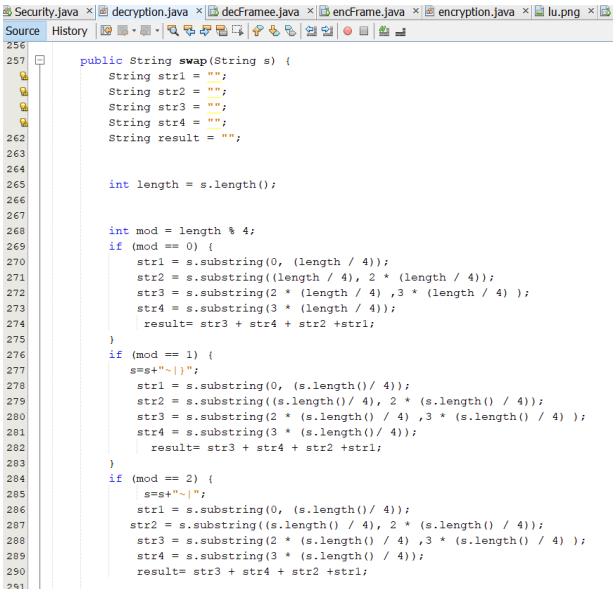




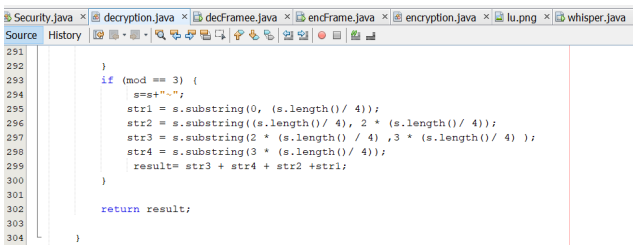




- The permutation method (swap):

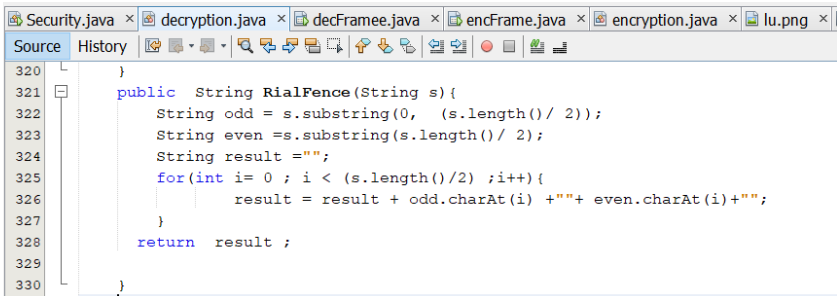


```
Security.java | decryption.java | decFramee.java | encFrame.java | encryption.java | lu.png
Source | History
256
257     public String swap(String s) {
            String str1 = "";
            String str2 = "";
            String str3 = "";
            String str4 = "";
            String result = "";


            int length = s.length();


            int mod = length % 4;
            if (mod == 0) {
                str1 = s.substring(0, (length / 4));
                str2 = s.substring((length / 4), 2 * (length / 4));
                str3 = s.substring(2 * (length / 4) ,3 * (length / 4) );
                str4 = s.substring(3 * (length / 4));
                 result= str3 + str4 + str2 +str1;
            }
            if (mod == 1) {
               s=s+"~|";
                str1 = s.substring(0, (s.length()/ 4));
                str2 = s.substring((s.length()/ 4), 2 * (s.length() / 4));
                str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length() / 4) );
                str4 = s.substring(3 * (s.length()/ 4));
                  result= str3 + str4 + str2 +str1;
            }
            if (mod == 2) {
                 s=s+"~|";
                str1 = s.substring(0, (s.length()/ 4));
               str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
                str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length() / 4) );
                str4 = s.substring(3 * (s.length() / 4));
                result= str3 + str4 + str2 +str1;
```



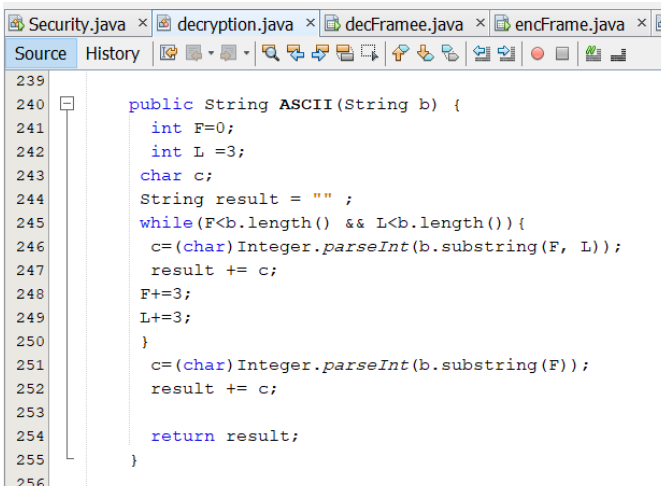
```
Security.java | decryption.java | decFramee.java | encFrame.java | encryption.java | lu.png | whisper.java
Source | History
251
252             }
293             if (mod == 3) {
294                 s=s+"~";
295                 str1 = s.substring(0, (s.length()/ 4));
296                 str2 = s.substring((s.length()/ 4), 2 * (s.length()/ 4));
297                 str3 = s.substring(2 * (s.length() / 4) ,3 * (s.length()/ 4) );
298                 str4 = s.substring(3 * (s.length()/ 4));
299                  result= str3 + str4 + str2 +str1;
300             }
301
302             return result;
303
304         }
```

- Rail Fence method (Railfence):



```
Security.java | decryption.java | decFramee.java | encFrame.java | encryption.java | lu.png
Source | History
320     }
321     public  String RialFence(String s){
322         String odd = s.substring(0,  (s.length()/ 2));
323         String even =s.substring(s.length()/ 2);
324         String result ="";
325         for(int i= 0 ; i < (s.length()/2) ;i++){
326                 result = result + odd.charAt(i) +""+ even.charAt(i)+"";
327         }
328      return  result ;
329
330     }
```

- Converting to ASCII method (ASCII):



```
Security.java | decryption.java | decFramee.java | encFrame.java
Source | History
239
240     public String ASCII(String b) {
241       int F=0;
242       int L =3;
243      char c;
244      String result = "" ;
245      while(F<b.length() && L<b.length()){
246       c=(char)Integer.parseInt(b.substring(F, L));
247       result += c;
248      F+=3;
249      L+=3;
250      }
251       c=(char)Integer.parseInt(b.substring(F));
252       result += c;
253
254       return result;
255     }
256
```

- Swap ASCII method (convert Ascii):

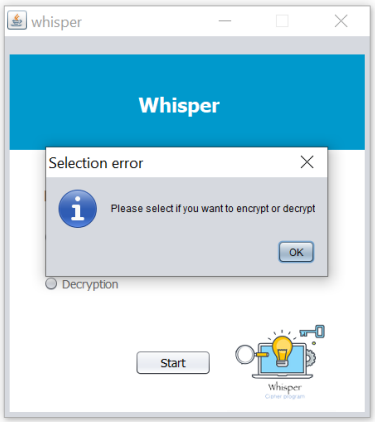
```
330     }
331
332     public String convertAscii(String s){
333
334         int length = s.length();
335         char[] array = s.toCharArray();
336         String total="";
337         int j=1;
338         int k = 0;
339             for(; j+2<=length && k+2<=length;j+=2,k+=2){
340            total=total+""+ array[j]+""+array[k];
341
342            }
343                total=total+""+ array[j]+""+array[k];
344
345     return total;
346     }
347
348   }
```

- Method (validate):

```
305
306     public  boolean validate(String s){    //return true if the input is valid false other wise
307       if(s.length()<1)
308            return false;
309        if (s.length() % 12 !=0){
310            System.out.println("The length has to be multiplier of 12");
311            return false;}
312
313         for(int i=0 ; i<s.length() ;i++)
314        if (!Character.isDigit(s.charAt(i)) ){
315            System.out.println("you are allowed to enter digits only");
316             return false;}
317
318             return true;
319
320     }
```

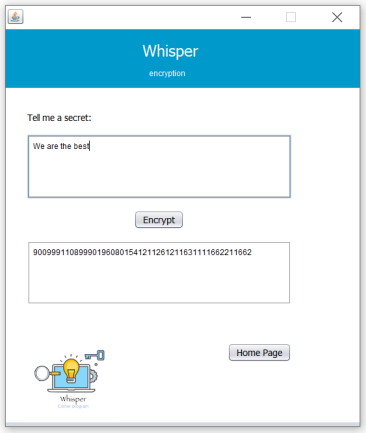
sample of the tests:

1- The user selects Start without selecting an option.

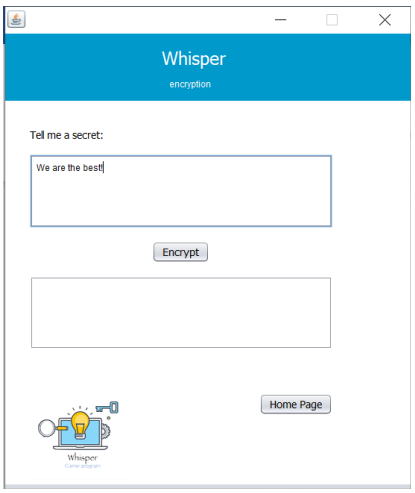


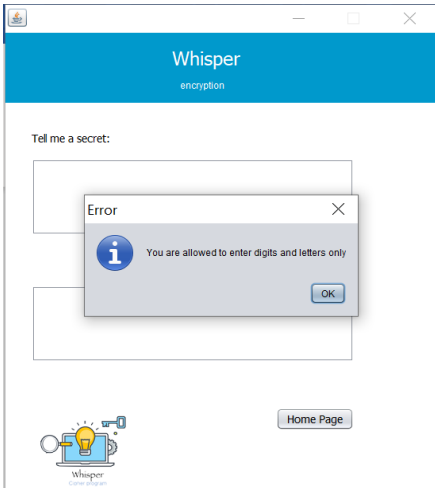
2-Using encryption:

2A – When the user enters a valid input(letters and spaces) and results with a ciphertext.

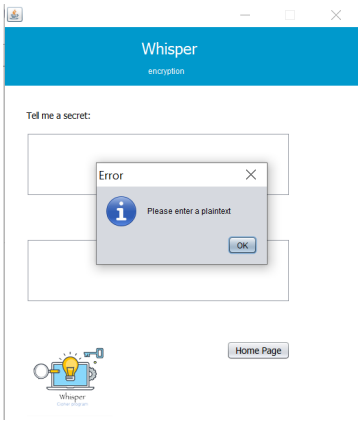


2B- User enters an invalid input (letters with symbols).

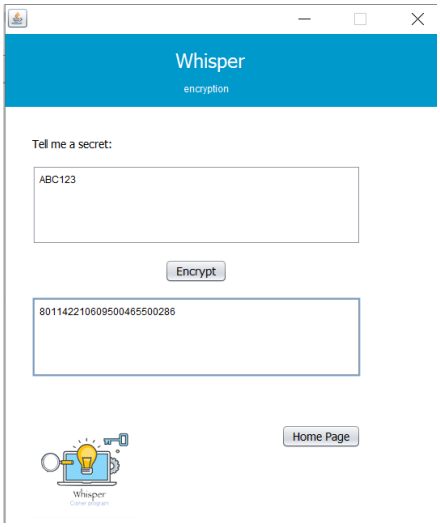




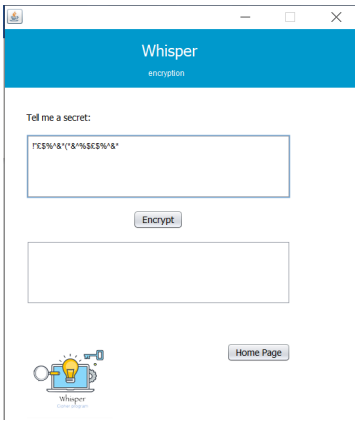
2C- When the user selects on Encrypt without typing a plaintext.

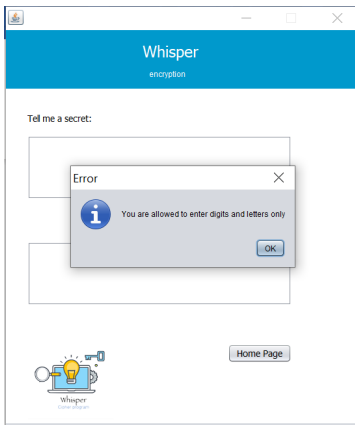


2D- When the user enters a valid input (letters and numbers) and results with a ciphertext.



2E- The user enters an invalid input (symbols).

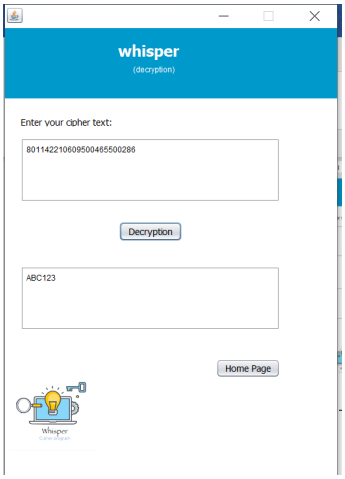




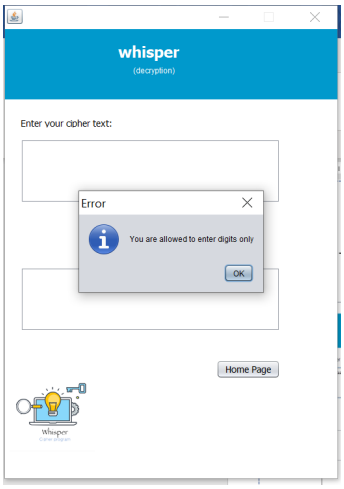
3-Using decryption:

3A – The user selects decrypt without entering a ciphertext.

3B – The user enters a valid ciphertext and results with a plaintext.

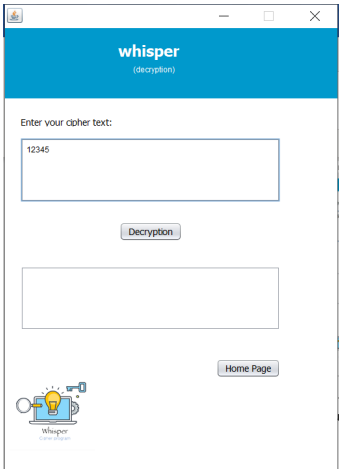


3C – The user enters an invalid input as ciphertext (user enters letters).

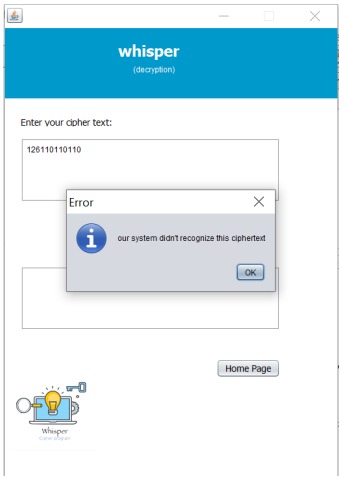


3D- The user enters an invalid input as cipherext (user enters symbols).

3E – The user enters an invalid input as ciphertext (user enters ciphertext where its length isnt a multipule of 12).



3F- The user enters an invalid input as ciphertext (user enters a ciphertext that the system does not recognize).



At the end, as a team, we have a great experience to work together on this project. And we saw that Whisper has more than three components and it can encrypt and decrypt both letters (dealing with capital and small letters) and numbers which make it a strong model, the two rounds influence the strength of the model. Because of these reasons, Whisper system achieved the goal of security and protected the valuable information.

References :

[1] Richardson, T. and Thies, C. (2013). *Secure software design*. Burlington, Mass.: Jones & Bartlett Learning.

[2] Wu, C. (2010). *An introduction to object-oriented programming with Java*. 5th ed. New York: McGraw-Hill Higher Education.

[3] C. Baziotis, "How to go back to a JFrames in Java," Stack Overflow, 01-Mar-1962. [Online]. Available: https://stackoverflow.com/questions/8771558/how-to-go-back-to-a-jframes-in-java/8771756. [Accessed: 18-Nov-2019].

[4] P. K. P. Kharbuja, "How to add radio buttons in button group?," Stack Overflow, 01-Feb-1963. [Online]. Available: https://stackoverflow.com/questions/13832209/how-to-add-radio-buttons-in-button-group. [Accessed: 18-Nov-2019].

[5] Java #N3 - Making a GUI in NetBeans. [Online]. Available: https://www.youtube.com/watch?v=LFr06ZKIpSM. [Accessed: 18-Nov-2019].

[6] "GUI Programming - Java Programming Tutorial", *Ntu.edu.sg*, 2019. [Online]. Available: https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html. [Accessed: 19- Nov- 2019].

[7] J. resizing?, Z. Junejo, I. Gaur, B. Vijay, C. Latte and D. Gavkar, "JFrame: How to disable window resizing?", *Stack Overflow*, 2019. [Online]. Available: https://stackoverflow.com/questions/18031704/jframe-how-to-disable-window-resizing. [Accessed: 19- Nov- 2019].

[8] J. NetBeans, A. Thompson and Y. K., "JFrame Resizing in Desktop Application - NetBeans", *Stack Overflow*, 2019. [Online]. Available: https://stackoverflow.com/questions/8265548/jframe-resizing-in-desktop-application-netbeans. [Accessed: 19- Nov- 2019].