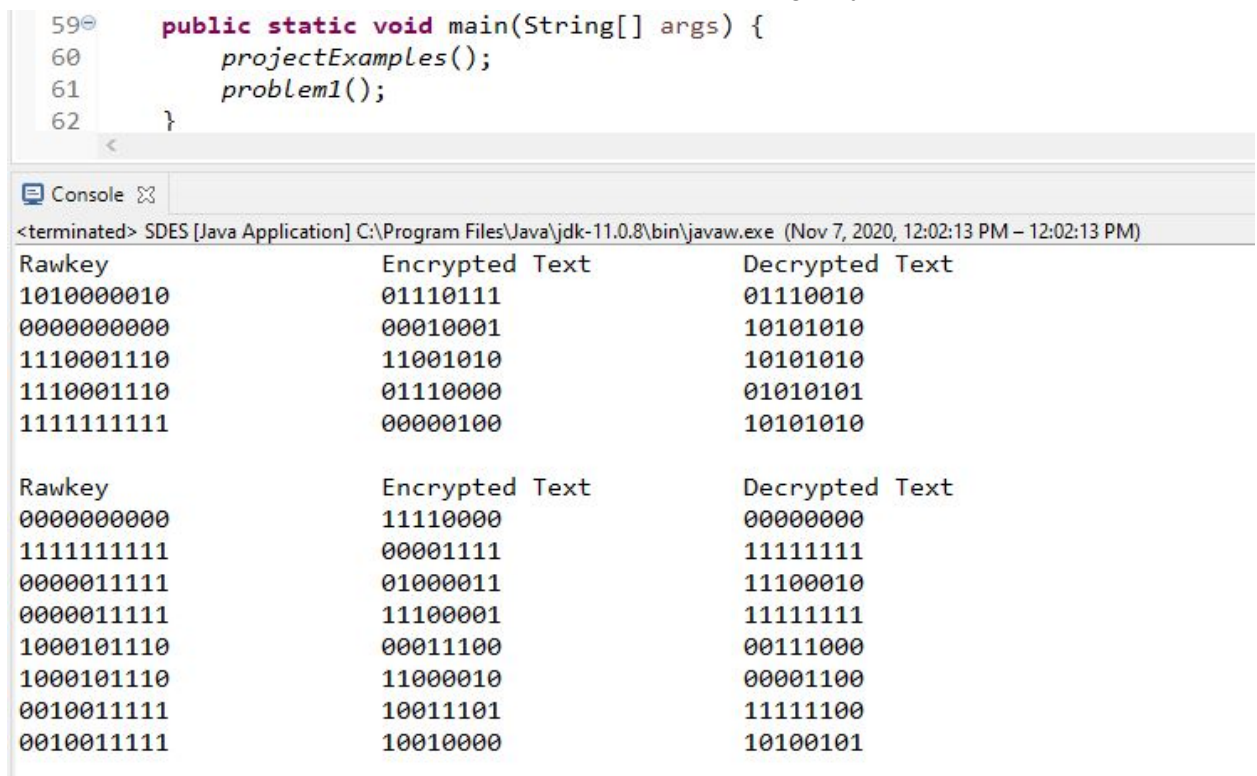


In order to run the project, make sure all files (CASCII.java, Cracking.java, SDES.java, and TripleSDES.java) are in the same package in your IDE. Each problem is separated in its own java file, but Cracking.java utilizes some other files, so all java files are needed in order to work correctly.

## 1. SDES

To run SDES, open the SDES file in the IDE and run the file. In the console, it should print out the examples we were given for the project, as well as the questions we were given to solve (as shown below in pic 1). In order to run the Encryption/Decryption by itself, you would just need to run the appropriate function (i.e. Encrypt(rawkey, plaintext); ) and it will return an array of bytes. Our class turns those byte arrays to strings and prints it out so the user can see the result in a more appealing way.

```
59 public static void main(String[] args) {
60     projectExamples();
61     problem1();
62 }
```



Rawkey	Encrypted Text	Decrypted Text
1010000010	01110111	01110010
0000000000	00010001	10101010
1110001110	11001010	10101010
1110001110	01110000	01010101
1111111111	00000100	10101010

Rawkey	Encrypted Text	Decrypted Text
0000000000	11110000	00000000
1111111111	00001111	11111111
0000011111	01000011	11100010
0000011111	11100001	11111111
1000101110	00011100	00111000
1000101110	11000010	00001100
0010011111	10011101	11111100
0010011111	10010000	10101010

(Pic 1 - SDES.java running and showing the results of the examples and questions from the project)

Rawkey	Plaintext	Cipherkey
00000 00000	0000 0000	1111 0000
11111 11111	1111 1111	0000 1111
00000 11111	0000 0000	0100 0011
00000 11111	1111 1111	1110 0001
10001 01110	0011 1000	0001 1100
10001 01110	0000 1100	1100 0010
00100 11111	1111 1100	1001 1101
00100 11111	1010 0101	1001 0000

(Table 1 - Results for the problems given in the project highlighted in green)

Also attached is a table with the results for the problems given in the project.

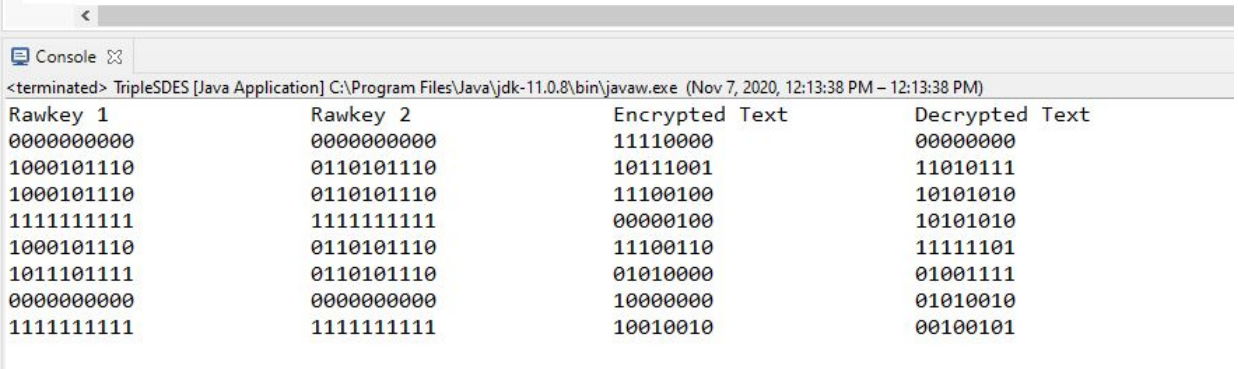
## 2. TripleSDES

TripleSDES works just like SDES. Open the file and run the file in order to see the results. The only real difference for TripleSDES was that we had to have 2 keys, and run the specific algorithm - Encrypt(Decrypt(Encrypt) ) or vice versa in order to obtain the results we wanted. So in this class, we added a EncryptMini and DecryptMini which are the same methods as Encrypt and Decrypt from SDES so we can run the triple in the main methods for this class.

```

52 public static void main(String[] args) {
53     problem2();
54 }
55

```



Rawkey 1	Rawkey 2	Encrypted Text	Decrypted Text
0000000000	0000000000	11110000	00000000
1000101110	0110101110	10111001	11010111
1000101110	0110101110	11100100	10101010
1111111111	1111111111	00000100	10101010
1000101110	0110101110	11100110	11111101
1011101111	0110101110	01010000	01001111
0000000000	0000000000	10000000	01010010
1111111111	1111111111	10010010	00100101

(Pic 2 - TripleSDES.java running displaying the results of the problems from the project)

Raw Key 1	Raw Key 2	Plaintext	Ciphertext
00000 00000	00000 00000	0000 0000	1111 0000
10001 01110	01101 01110	1101 0111	1011 1001
10001 01110	01101 01110	1010 1010	1110 0100
11111 11111	11111 11111	1010 1010	0000 0100
10001 01110	01101 01110	1111 1101	1110 0110
10111 01111	01101 01110	0100 1111	0101 0000
00000 00000	00000 00000	0101 0010	1000 0000
11111 11111	11111 11111	0010 0101	1001 0010

(Table 2 - Results for the problems given in the project highlighted in green)

### 3. Cracking SDES and Triple SDES

The Cracking java file contains 3 parts from the project and are separated into 3 methods in the class file. As a reminder, this class utilizes other java files, so having all java files are needed for this class to properly work.

#### 3.1. Encode the CASCII plaintext CRYPTOGRAPHY using SDES, key of 01110 01101.

For this problem, we converted the "CRYPTOGRAPHY" into a byte array using the CASCII class, then encrypted the text using our SDES class.

```

29 public static void part1() {
30     String rawkey = "0111001101";
31
32     // Convert message to CASCII byte array
33     byte[] plaintext = CASCII.Convert(string);
34
35     // Encrypt the CASCII message with the key
36     byte[] ciphertext = SDES.Encrypt(toByteArray(rawkey), plaintext);
37
38     System.out.println("Encrypted Message: ");
39     printArray(ciphertext);
40
41 }

```

Console

<terminated> Cracking [Java Application] C:\Program Files\Java\jdk-11.0.8\bin\javaw.exe (Nov 7, 2020, 12:23:02 PM – 12:23:05 PM)

Encrypted Message:

011000011100101010110110111010011000101011011011111000111011

(Pic 3-1 - Conversion and encryption for the message and the encrypted message result)

### 3.2. Decrypt msg1.txt using SDES and provide the 10-bit key

For this problem, we ran the function multiple times to make sure it was right; and we ended by making it simple for the user to see the end result by manually finding a key phrase for the decrypted message. **The final message ended up being “WHOEVER THINKS HIS PROBLEM CAN BE SOLVED USING CRYPTOGRAPHY, DOESN'T UNDERSTAND HIS PROBLEM AND DOESN'T UNDERSTAND CRYPTOGRAPHY. ATTRIBUTED BY ROGER NEEDHAM AND BUTLER LAMPSON TO EACH OTHER”** from a key of 10111 10101.

```
43 public static void part2() {
44     String currentIter = "0000000000";
45     byte[] plaintext;
46     byte[] ciphertext = toByteArray(msg1);
47
48     // Do 2^10 iterations
49     for (int i=0; i<1024; i++) {
50         plaintext = SDES.Decrypt(toByteArray(currentIter), ciphertext);
51         currentIter = addBit(currentIter);
52         String decrypted = CASCII.toString(plaintext);
53
54         // After manually checking, the decryption seems to have a "WHOEVER", "AND", and "HIS" for some reference
55         if (decrypted.contains("WHOEVER")) {
56             System.out.println("Key: " + currentIter);
57             System.out.println(decrypted);
58         }
59     }
60 }
61 }
```

Console

<terminated> Cracking [Java Application] C:\Program Files\Java\jdk-11.0.8\bin\javaw.exe (Nov 7, 2020, 12:25:35 PM – 12:25:36 PM)

Key: 1011110101

WHOEVER THINKS HIS PROBLEM CAN BE SOLVED USING CRYPTOGRAPHY, DOESN'T UNDERSTAND HIS PROBLEM AND DOESN'T UNDERSTAND CRYPTOGRAPHY.

(Pic 3-2 - SDES Decryption of the message and the result.

### 3.3. Decrypt msg2.txt using TripleSDES and provide the 2 10-bit keys

This problem was tackled the same way as the one with SDES; we just had to wait longer and manually search more since there were 2 keys and a lot more text to go through. This one, we also made a File reader to export the text so that it would be easier to search through the results. **The final message ended up being “THERE ARE NO SECRETS BETTER KEPT THAN THE SECRETS THAT EVERYBODY GUESSES.”** with key1 as 11100 00101 and key2 as 01011 00011.

```

63 public static void part3() {
64     String key1 = "0000000000";
65     String key2 = "0000000000";
66
67     byte[] plaintext;
68     byte[] ciphertext = toByteArray(msg2);
69
70     try {
71         File file = new File("part3.txt");
72         if (file.createNewFile()) {
73             System.out.println("created File");
74         } else {
75             System.out.println("File exists");
76         }
77
78         FileWriter writer = new FileWriter("part3.txt");
79
80         // Do 2^10 ^ 2^10 iterations
81         for (int i=0; i<1024; i++) {
82             for (int j=0; j<1024; j++) {
83                 plaintext = TripleDES.Decrypt(toByteArray(key1), toByteArray(key2), ciphertext);
84                 String decrypted = ASCII.toString(plaintext);
85
86                 // Manually checking, the decrypted message has a "THERE"
87                 if (decrypted.contains("THERE")) {
88                     System.out.println("Key 1: " + key1 + "\tKey 2: " + key2);
89                     System.out.println(decrypted);
90                     writer.write("Key 1: " + key1 + "\nKey 2: " + key2 + "\n");
91                     writer.write(decrypted);
92                 }
93                 key1 = addBit(key1);
94             }
95             key2 = addBit(key2);
96         }
97     }
98 }
99

```

Console

<terminated> Cracking [Java Application] C:\Program Files\Java\jdk-11.0.8\bin\javaw.exe (Nov 7, 2020, 12:32:59 PM – 12:33:31 PM)

File exists

Key 1: 1110000101      Key 2: 0101100011

THERE ARE NO SECRETS BETTER KEPT THAN THE SECRETS THAT EVERYBODY GUESSES.

(Pic 3-3 - TripleDES decoding of the message and the result)