



MARMARA
UNIVERSITY

EE1004 / CSE2062

Object-Oriented Programming

Project #2

Submission: 12.05.2024

Due: 26.05.2024 Sunday at 23:59

Demo: 27.05.2024 Monday 11:00 – 18:00

There are **two** parts in this project.

This project must be done by a group of **4 (four) (dört)** students. Team members must be indicated [in this spreadsheet](#) until **15.05.2024, 23:59**. Otherwise, you will not have the opportunity to present your work in DEMO (i.e. - 60%).

Report (team grade)	40 %
Demo (individual grade)	60 %
Total	100 %

Aim of the Project

You are expected to develop a program by using JAVA language.

In this project you will first encrypt a message on an image by using the fact that the pixels of the image carry some specific data that is ordinarily a color data stored as RGB values. You will manipulate this data to hide another information in them (in this case you are expected to hide characters as ASCII values).

You will be given clear instructions on how to store these data in a way that you can encrypt the message into an image and get back the hidden message from it.

On the second part of the project you are expected to decrypt a given image and get the hidden secret message back. You will also be given clear instructions on how to get the message properly.

Choosing an Image

Please select proper images for encryption, i.e. a flower, a car, an animal or a mountain. The size of the image should not exceed 50x50 pixels. You can downsize any image by using a third-part programs such as Paint[®]. Improper images will be penalized.

Attention! We will also try different images and messages in the demo. Also encrypted messages will be changed in the demo for the second part.

Libraries

You are expected to solve this project using any kind of built-in or third party library **to read** an image pixel by pixel. (Only for *image read* part.)

Coding Rules

- You must use at least one multi-dimensional array to store RGB values in an image.
- There should be at least two class definitions.
- You should use **Object-Oriented Programming Concepts** in your program. (Inheritance, Encapsulation, Abstraction etc.)
- Error handling should be done.
- A log file should be arranged. A sample log info at the end of this document. You can insert more meaningful log info into the text file.
- (Optional!) Even though we will not be able to cover the topic of GUI Design in JAVA, you can enhance the appearance of your project with an interface by using *Java Swing*.
- Do **NOT** use variable names such as x, y, z, t and so on.
- Do **NOT** forget that your main function should be as short as possible.
- Your methods should **NOT** be too long and verbose, if necessary create a new method with a logical name.
- Establish a standard size for an indent, such as one tab, and use it consistently. Align sections of code using the prescribed indentation.
- Source code should be cleverly commented (Do **NOT** abuse!).
- Late uploads will **NOT** be considered.
- You can discuss the project with your friends, but code sharing is **NOT** allowed.
- Your source codes will be tested through the measure of software similarity. So, do **NOT** cheat!
- E-mail submissions will **NOT** be accepted!!!
- You will present your work as a group in the face-to-face DEMO. DEMO scheduling will be announced **2 days before the DEMO day**.

Report

Report format can be downloaded under the Projects section in the Main Menu in the following days. You must prepare your report as the format implies. Otherwise, you will get 0 for the Report Grade. Also you must convert your report to PDF before submission. Other file formats will also be penalized. Reports can be uploaded by only **1 (one) (bir)** student from each group.

Code Submission

All students from each groups must submit the code under the related part!

Project Tasks

PART-1: Encryption

Getting User Input and Sample Run of the Program

You're expected to ask for a secret message with a length between 1-255 and print the length of the message. We know that RGB values are at range 0-255. If the user enters nothing or a message longer than 255 characters, print the length of the message and ask for a new one until the user enters a message with a proper length.

Please understand the relationship between line length restriction and RGB range before moving on to the next part. A sample run is given in the Figure 1.

```
Enter the message you want to hide (Must be 1-255 characters long):
Message Length:0
Enter a String with length 1-255!: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae augue ultricies,
efficitur ex ac, blandit arcu. Vestibulum imperdiet accumsan mi, id viverra nulla faucibus et. In ut nibh posuere, maximus
leo nec, tempus leo. Mauris tincidunt eu odio id feugiat. Nulla facilisis erat eu velit dignissim aliquam. Quisque interdum
diam at dui imperdiet commodo. Nunc gravida enim sit amet fringilla congue.
Message Length:408
Enter a String with length 1-255!: Meet me at our ordinary place. 4 pm sharp!
Message Length:42
Your message is successfully encrypted.
```

After encryption is completed, a message is printed to signal that fact:” ***Message has been successfully encrypted.***”

Encrypting an Image

Encryption Method: There are infinitely many ways for encoding a set of information on an image. However, for this project you are expected to encrypt the message with a way we specifically asked.

In this project you will choose a target **blue** value randomly. Then, you will hide your secret message to the pixels with **blue** values equal to this target **blue** value.

Choosing a Random **Blue** Value

First of all, you need to choose a random **blue** value before you start hiding your secret message. This **blue** value will be your target **blue** value. The only constraint is that it must be different than the message length. Also this value surely does not violate RGB value boundary. If these rules violated by the chosen **blue** value then give an error message of “**Invalid entry.**”, and generate a new target **blue** value.

Creating a Random Value Array

Create an array with the size of the message length, i.e. `randomArray[i]`. Each element in this array must be an integer in the range between 0-255. The integer values must be unique, i.e., each integer must be different from each other. This array will be essential for your decryption algorithm.

Then sort the random array in ascending order. For example, the array [8, 2, 15, 19, 1, 67] will become [1, 2, 8, 15, 19, 67]. You can use any sorting algorithm you want, however, you need to understand the logic behind it.

Changing the Pixels with Target Blue Value

Only the pixels that carry the secret message must have blue values equal to the target blue value. To make sure that, you need to find pixels that have already had the target blue value and change their blue value to another value. You can make this change by increasing the blue value by 1. Aim of this step is preventing them from corrupting your message. After you complete this part, there shouldn't be any pixel with target blue value.

Example: Let's say your target blue value is 55 and one of the pixel in your image has the following rgb values: {63, 98, 55}. As you see, the blue value of this pixel and your target blue value are the same. Therefore, you must change the blue value of this pixel by increasing the value by 1. This will lead to given pixel's rgb values to be {63, 98, 56}.

Hiding the Message Length and Target Blue Value in Pixel[0,0]

You will use the [0,0] pixel of the image as a key and keep the message length and target blue value in this pixel. To do so, the following assignments are needed:

- Red Value of the [0,0] pixel = Message Length.
- Green Value of the [0,0] pixel = Target blue Value.
- Blue Value of the [0,0] pixel = Do not change.

Choosing Random Pixels and Hiding Information on them

Choose random pixels and set each random pixel's RGB values to:

- Red Value = randomArray[i]
- Green Value = ASCII code of the i'th character in the secret message
- Blue Value = Target blue Value

Hints:

1. Do not choose (0,0) pixel as one of the random pixels. Remember, it is your key-pixel!
2. Make sure that you use a pixel only once!

Sample Messages to be Encrypted

Hello, World! This is me :)	Meet me at our ordinary place. 4 pm sharp!
Hostile attack will be happened at mid-night.	No homework for tomorrow but stay tuned!

PART-2: Decryption

After you finished the Part 1 (encryption), you can now develop a program to reveal the message hidden in an image.

Getting Message Length and Target **blue** Value

In the Part 1 (encryption), you have hidden the message length and target **blue** value into pixel at the location [0,0]. Since we will use these values, you need to extract them from the encrypted image.

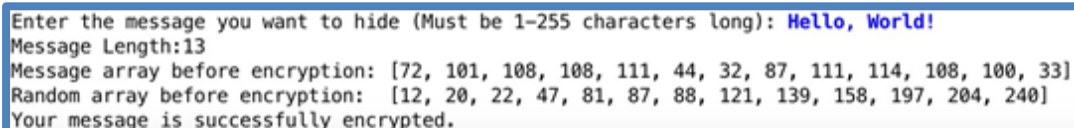
Getting the Message Array and Random Array from Image

Since you know the message length and which **blue** value you need to look for, you can start to extract hidden information, i.e. decrypt the image and get the message. You need to create two arrays and fill the first array with the *randomArray* and the second array with the ASCII values of characters of the hidden message.

Hint: You must put the information you get from one particular pixel into the same index of both arrays.

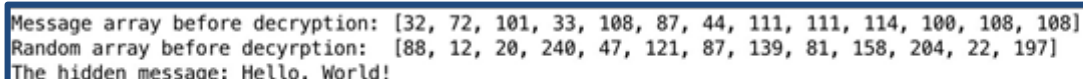
Sorting Two Arrays Simultaneously

In Part 1 (encryption), ASCII values of the message characters are assigned to random pixels together with a sorted random array. However, while you're looking for these pixels, you may not find them in the same order. Because, the order of the characters will be mixed up. Fortunately, the random array will be mixed up with exactly the same way. In other words, the values, that share the same index location before the encryption, will still share the same index location. Screenshots that demonstrates this change are given in Figure 2.



```
Enter the message you want to hide (Must be 1-255 characters long): Hello, World!
Message Length:13
Message array before encryption: [72, 101, 108, 108, 111, 44, 32, 87, 111, 114, 108, 100, 33]
Random array before encryption: [12, 20, 22, 47, 81, 87, 88, 121, 139, 158, 197, 204, 240]
Your message is successfully encrypted.
```

a)



```
Message array before decryption: [32, 72, 101, 33, 108, 87, 44, 111, 111, 114, 100, 108, 108]
Random array before decryption: [88, 12, 20, 240, 47, 121, 87, 139, 81, 158, 204, 22, 197]
The hidden message: Hello, World!
```

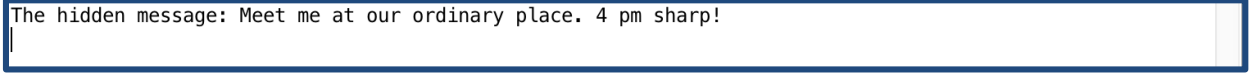
b)

Figure 2: A sample run to demonstrate the change of the message array (ASCII values) and random array before encryption (a) and before decryption (b).

So, if we sort the mixed random array back in ascending order and re-arrange the message array accordingly, we can get ASCII values of the message characters in correct order. You're expected to write a method that will perform this process. You are not restricted to which sorting algorithm to use, however, you are expected to write your own method.

Printing the Message

Congratulations! You have successfully decrypted the image and acquire the ASCII values of characters in correct order. Your final task is to print the secret message to Console as an example given in Figure 3.



```
The hidden message: Meet me at our ordinary place. 4 pm sharp!
```

Figure 3: A sample run for printing the decrypted hidden message.

Sample log File Info

The following entries are just an example. Do NOT directly insert these into your code. Be original!

A user entered the system.	User_ID	123456	Sun May 12 11:48:15 GMT 2024
Mode Selection.	Mode	1	Sun May 12 11:49:25 GMT 2024
A message entered.	Size	522	Sun May 12 11:49:45 GMT 2024
Fault in message.	Fault_Code	3	Sun May 12 11:49:46 GMT 2024
A message entered.	Size	125	Sun May 12 11:50:45 GMT 2024
An image uploaded.	Im_Type	4(png)	Sun May 12 11:52:16 GMT 2024
Encryption process.	Success	true	Sun May 12 11:52:56 GMT 2024
Mode Selection.	Mode	2	Sun May 12 11:54:25 GMT 2024
An image uploaded.	Im_Type	2(bmp)	Sun May 12 11:55:16 GMT 2024
Decryption process.	Success	true	Sun May 12 11:55:56 GMT 2024
Message revealed.	Size	113	Sun May 12 11:55:57 GMT 2024
A user exited the system.	User_ID	123456	Sun May 12 11:58:19 GMT 2024

END OF THE PROJECT ASSIGNMENT