

<https://github.com/sauravhathi>



**SIX WEEKS SUMMER TRAINING  
REPORT**

On

**DSA Self-paced**

Submitted by

Your Full Name

Registration No. 12000000

Program Name: Bachelor of Technology

Under the Guidance of

**Mr. Sandeep Jain**

School of Computer Science & Engineering

Lovely professional University, Phagwara

(May-July, 2022)

## DECLARATION

I hereby declare that I have completed my six weeks summer training at DSA Self paced from 25<sup>th</sup> May 2022 to 10<sup>th</sup> July 2022 under the guidance of Mr. Sandeep Jain. I have declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfil the requirements of training for the award of degree of Bachelor of Technology, Lovely Professional university, Phagwara.

(Signature of student)

Name of Student: Your Full Name

Registration no: 1200000

Date: 23th Aug, 2022

## **Acknowledgement**

It is with sense of gratitude; I acknowledge the efforts of entire hosts of well-wishers who have in some way or other contributed in their own special ways to the success and completion of the Summer Training.

Successfully completion of any type technology requires helps from a number of people. I have also taken help from different people for the preparation of the report. Now, there is little efforts to show my deep gratitude to those helpful people.

I would like to also thank my own college Lovely Professional University for offering such a course which not only improve my programming skill but also taught me other new technology.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance for choosing this course.

Last but not least I would like to thank my all classmates who have helped me a lot.

## Training certificate from organization



# CERTIFICATE

## OF COURSE COMPLETION

THIS IS TO CERTIFY THAT

has successfully completed the course on DSA Self paced of duration 8 weeks.

*Sandeep Jain*

**Mr. Sandeep Jain**  
Founder & CEO, GeeksforGeeks

<https://media.geeksforgeeks.org/courses/certificates/f53b0844011051170fa88c2c3b6cb686.pdf>

## Table Of Contents

1. Introduction
2. Technology Learnt
3. Reason for choosing this technology.
4. Profile of the Problem
5. Existing System
6. Problem Analysis
  - Product definition
  - Feasibility Analysis
7. Software Requirement Analysis
8. Design
  - Tables and their relationships
  - Flowcharts/Pseudo code
9. Implementation
10. Learning Outcome from training/technology learnt
11. Gantt chart
12. Project Legacy
  - Technical and Managerial learnt.
13. Bibliography

## Introduction

The course name DSA stands for “Data Structures and Algorithms” and Self-paced means, one can join the course anytime. All of the content will be available once one gets enrolled. One can finish it at his own decided speed.

### 1. What is Data Structure?

Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's name "Virat" and age 26. Here "Virat" is of String data type and 26 is of integer data type.

### 2. What is Algorithm?

An algorithm is a finite set of instructions or logic, written in order, to accomplish a certain predefined task. Algorithm is not the complete code or program, it is just the core logic(solution) of a problem, which can be expressed either as an informal high-level description as pseudocode or using a flowchart.

This course is a complete package that helped me learn Data Structures and Algorithms from basic to an advanced level. The course curriculum has been divided into 8 weeks where one can practice questions & attempt the assessment tests according to his own pace. The course offers me a wealth of programming challenges that will help me to prepare for interviews with top-notch companies like Microsoft, Amazon, Adobe etc.

## Technology Learnt

- Learn Data Structures and Algorithms from basic to an advanced level like:
- Learn Topic-wise implementation of different Data Structures & Algorithms as follows

### 1. Introduction

- **Asymptotic Notations**

Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.

- **Worst, Average and Best-Case Time Complexities**

It is important to analyze an algorithm after writing it to find its efficiency in terms of time and space in order to improve it if possible.

When it comes to analyzing algorithms, the asymptotic analysis seems to be the best way possible to do so. This is because asymptotic analysis analyzes algorithms in terms of the input size. It checks how are the time and space growing in terms of the input size.

- **Analysis of Loops**

### 2. Mathematics

- **Finding number of Digits in a Number**
- **Arithmetic and Geometric Progressions**
- **Quadratic Equations**
- **Mean and Median**
- **Prime Numbers**
- **LCM and HCF**
- **Factorials**
- **Permutation and Combinations Basics**
- **Modular Arithmetic**

### 3. Bit Magic

- **Binary Representation**
- **Set and Unset**
- **Toggling**
- **Bitwise Operators**
- **Algorithms**

**Objective:** The objective of this track is to familiarize the learners with *Bitwise Algorithms* which can be used to solve problems efficiently and some interesting tips and tricks using Bit Algorithms.

#### 4. Recursion

- **Recursion Basics**

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are [Towers of Hanoi \(TOH\)](#), [Inorder/Preorder/Postorder Tree Traversals](#), [DFS of Graph](#), etc.

- **Basic Problems on Recursion**

- **Tail Recursion**

A recursive function is said to be following Tail Recursion if it invokes itself at the end of the function. That is, if all of the statements are executed before the function invokes itself then it is said to be following Tail Recursion.

```
void printN(int N)
{
    if(N==0)
        return;
    else
        cout<<N<<" ";

    printN(N-1);
}
```

The above function call for **N = 5** will print:

**5 4 3 2 1**



- **Explanation of Subset Generation Problem**
- **Explanation of Joesphus Problem**
- **Explanation of permutations of a string**

We iterate from first to last index. For every index  $i$ , we swap the  $i$ -th character with the first index. This is how we fix characters at the current first index, then we recursively generate all permutations beginning with fixed characters (by parent recursive calls). After we have recursively generated all permutations with the first character fixed, then we move the first character back to its original position so that we can get the original string back and fix the next character at first position.

## Arrays

- **Introduction to Arrays**
- **Insertion and Deletion in Arrays**
- **Array Rotation**
- **Reversing an Array**
- **Sliding Window Technique**
- **Prefix Sum Array**
- **Implementing Arrays in C++ using STL**
- **Iterators in C++ STL**
- **Implementing Arrays in Java**
- **Sample Problems on Array**
- **XOR Linked List - A Memory Efficient Doubly Linked List | Set 1**

## 5. Searching

- **Binary Search Iterative and Recursive**
- **Binary Search and various associated problems**
- **Two Pointer Approach Problems**

## 6. Sorting

- **Implementation of C++ STL sort() function in Arrays and Vectors**
- **Sorting in Java**
- **Arrays.sort() in Java**
- **Collection.sort() in Java**
- **Stability in Sorting Algorithms**
- **Insertion Sort**
- **Merge Sort**
- **Quick Sort**
- **Overview of Sorting Algorithms**

## 7. Matrix

- **Introduction to Matrix in C++ and Java**
- **Multidimensional Matrix**
- **Pass Matrix as Argument**
- **Printing matrix in a snake pattern**

- **Transposing a matrix**
- **Rotating a Matrix**
- **Check if the element is present in a row and column-wise sorted matrix.**
- **Boundary Traversal**
- **Spiral Traversal**
- **Matrix Multiplication**
- **Search in row-wise and column-wise Sorted Matrix**

## **8. Hashing**

- **Introduction and Time complexity analysis**
- **Application of Hashing**
- **Discussion on Direct Address Table**
- **Working and examples on various Hash Functions**
- **Introduction and Various techniques on Collision Handling**
- **Chaining and its implementation**
- **Open Addressing and its Implementation**
- **Chaining V/S Open Addressing**
- **Double Hashing**
- **C++**
  - **Unordered Set**
  - **Unordered Map**
- **Java**
  - **HashSet**
  - **HashMap**

## **9. Strings**

- **Discussion of String DS**
- **Strings in CPP**
- **Strings in Java**
- **Rabin Karp Algorithm**
- **KMP Algorithm**

## **10. Linked List**

- **Introduction Implementation in CPP**
  - Implementation in Java
  - Comparison with Array DS
- **Doubly Linked List**
- **Circular Linked List**
- **Loop Problems**
  - Detecting Loops
  - Detecting loops using Floyd cycle detection
  - Detecting and Removing Loops in Linked List

## 11. Stack

- **Understanding the Stack data structure**
- **Applications of Stack**
- **Implementation of Stack in Array and Linked List**
  - In C++
  - In Java

## 12. Queue

- **Introduction and Application**
- **Implementation of the queue using array and LinkedList**
  - In C++ STL
  - In Java
  - Stack using queue

## 13. Deque

- **Introduction and Application**
- **Implementation**
  - In C++ STL 14
  - In Java
- **Problems (With Video Solutions)**
  - Maximums of all subarrays of size k
  - ArrayDeque in Java
  - Design a DS with min max operations

## 14. Tree

- **Introduction**
  - Tree
  - Application
  - Binary Tree
  - Tree Traversal
- **Implementation of:**
  - Inorder Traversal
  - Preorder Traversal
  - Postorder Traversal
  - Level Order Traversal (Line by Line)
  - Tree Traversal in Spiral Form

## 15. Binary Search Tree

- **Background, Introduction and Application**
- **Implementation of Search in BST**
- **Insertion in BST**
- **Deletion in BST**
- **Floor in BST**
- **Self Balancing BST**
- **AVL Tree**

## 16. Heap

- **Introduction & Implementation**
- **Binary Heap**
  - Insertion
  - Heapify and Extract
  - Decrease Key, Delete and Build Heap
- **Heap Sort**
- **Priority Queue in C++**
- **PriorityQueue in Java**

## 17. Graph

- **Introduction to Graph**
- **Graph Representation**

- Adjacency Matrix
- Adjacency List in CPP and Java
- Adjacency Matrix VS List
- **Breadth-First Search**
  - Applications
- **Depth First Search**
  - Applications
- **Shortest Path in Directed Acyclic Graph**
- **Prim's Algorithm/Minimum Spanning Tree**
  - Implementation in CPP
  - Implementation in Java
- **Dijkstra's Shortest Path Algorithm**
  - Implementation in CPP 16
  - Implementation in Java
- **Bellman-Ford Shortest Path Algorithm**
- **Kosaraju's Algorithm**
- **Articulation Point**
- **Bridges in Graph**
- **Tarjan's Algorithm**

## 18. Greedy

- **Introduction**
- **Activity Selection Problem**
- **Fractional Knapsack**
- **Job Sequencing Problem**

## 19. Backtracking

- **Concepts of Backtracking**
- **Rat In a Maze**
- **N Queen Problem**

## 20. Dynamic Programming

- **Introduction**
- **Dynamic Programming**

- Memoization
- Tabulation

## 21. Tree

- **Introduction**
  - Representation
  - Search
  - Insert
  - Delete
- **Count Distinct Rows in a Binary Matrix**

## 23. Segment Tree

- **Introduction**
- **Construction**
- **Range Query**
- **Update Query**

## 24. Disjoint Set

- **Introduction**
- **Find and Union Operations**
- **Union by Rank**
- **Path Compression**
- **Kruskal's Algorithm**

1. Improved my problem-solving skills by practicing problems to become a stronger developer
2. Developed my analytical skills on Data Structures to use them efficiently
3. Solved problems asked in product-based companies' interviews
4. Solved problems in contests similar to coding round for SDE role

## Reason for choosing this technology

With advancement and innovation in technology, programming is becoming a highly in-demand skill for Software Developers. Everything you see around yourself from Smart TVs, ACs, Lights, Traffic Signals uses some kind of programming for executing user commands.

Data Structures and Algorithms are the identity of a good Software Developer. The interviews for technical roles in some of the tech giants like Google, Facebook, Amazon, Flipkart is more focused on measuring the knowledge of Data Structures and Algorithms of the candidates. The main reason behind this is Data Structures and Algorithms improves the problem-solving ability of a candidate to a great extent.

1. This course has video lectures of all the topics from which one can easily learn. I prefer learning from video rather than books and notes. I know books and notes and thesis have their own significance but still video lecture or face to face lectures make it easy to understand faster as we are involved Practically.
2. It has 200+ algorithmic coding problems with video explained solutions.
3. It has track based learning and weekly assessment to test my skills.
4. It was a great opportunity for me to invest my time in learning instead of wasting it here and there during my summer break in this Covid-19 pandemic.
5. This was a life time accessible course which I can use to learn even after my training whenever I want to revise.



## Profile of the Problem

One such practical scenario is the contract between the Banks to perform fund/message transfer. The information has to be confidential and each bank might use its own encryption & decryption techniques. The Reserve Bank might setup an Interface giving the method names and not worrying about the implementation.

1. Create an interface BankTransfers
2. Add two methods with the following prototype

```
--- public String encrypt(String a);
```

```
--- public String decrypt(String a);
```

3. Create class ICICI which implements the BankTransfers Interface & implements a simple encryption technique.

4. Create class HDFC which implements the BankTransfers Interface & implements a simple encryption technique.

5. Encrypt technique followed by both banks:

ICICI - add 1 with the ASCII value of the character and insert number '1' after every character.

HDFC - add 1 with the ASCII value of the character in the even Index and subtract 1 with the ASCII value of the character in the odd Index. It does not encrypt the space.

The reverse of both will be decrypted the message (i.e original text)

## Problem Analysis

### 1. Product definition

One such practical scenario is the contract between the Banks to perform fund/message transfer. The information has to be confidential and each bank might use its own encryption & decryption techniques.

### 2. Feasibility Analysis

The feasibility of this project is dependent on the security measures that each bank has in place. The goal of this project is to assess the feasibility of using blockchain technology in order to store and share confidential information between banks. If the banks are not using encryption and decryption techniques, then it would be impossible for them to use a blockchain for storing and sharing confidential information.

The feasibility analysis of this project would be to find out how much the banks are willing to invest in this project. The cost for each bank will be different, depending on their own encryption and decryption techniques. There are three major factors that need to be considered before proceeding with this project: the financial feasibility, the technical feasibility, and the legal compliance.

The current state of banking is a mess. There are too many banks, systems and data sources that are not connected. This makes it very difficult to provide a seamless experience for the customers. The information has to be confidential and each bank might use its own encryption & decryption techniques. The paper will try to find out if it is possible to make a system that can handle all these things in an efficient way.

## Software Requirement Analysis

### 1. Java

It is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is used to develop mobile apps, web apps, desktop apps, games and much more.

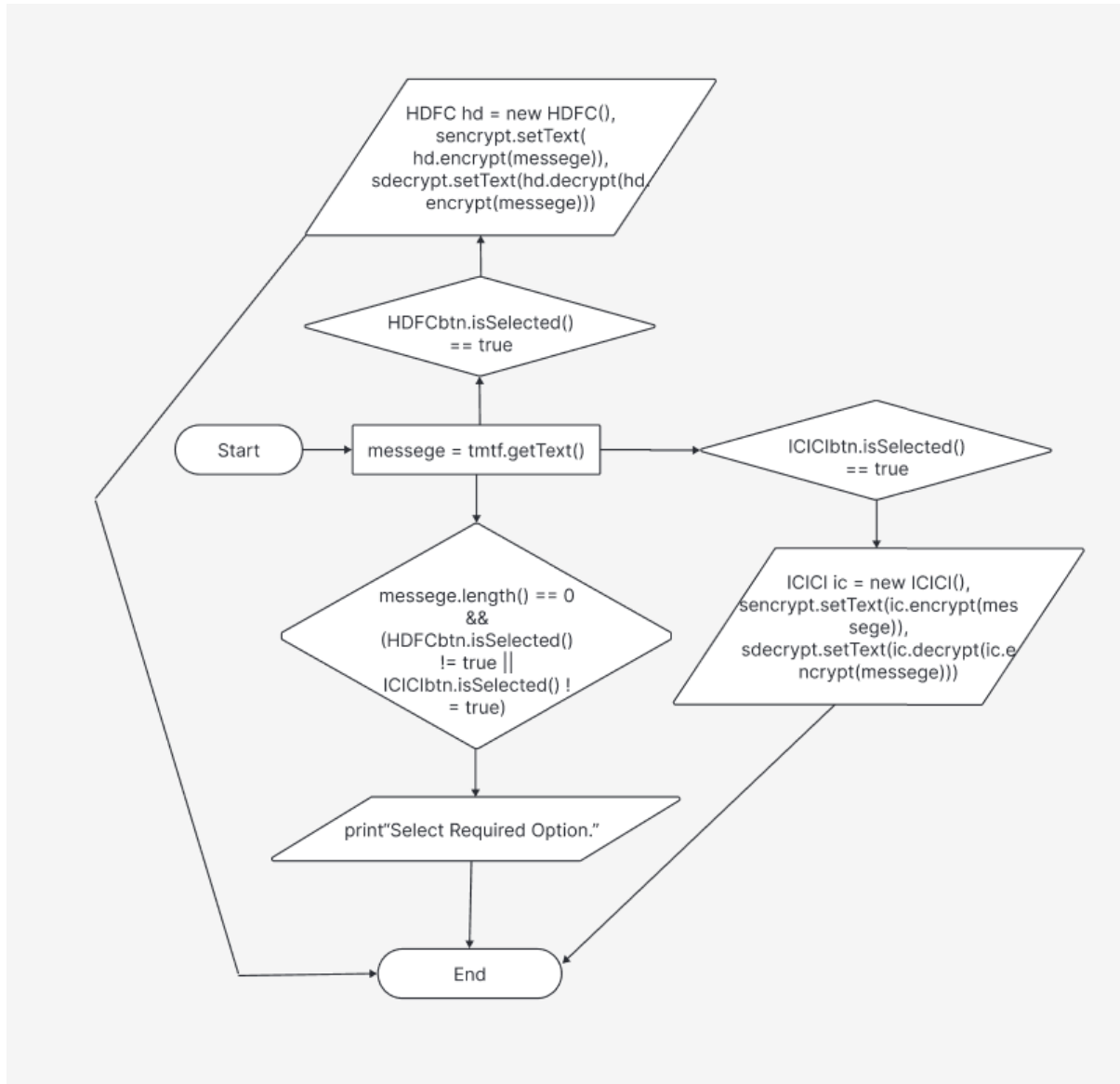
### 2. NetBeans IDE

It is a free and open-source integrated development environment for application development on Windows, Mac, Linux, and Solaris operating systems. The IDE simplifies the development of web, enterprise, desktop, and mobile applications that use the Java and HTML5 platforms.

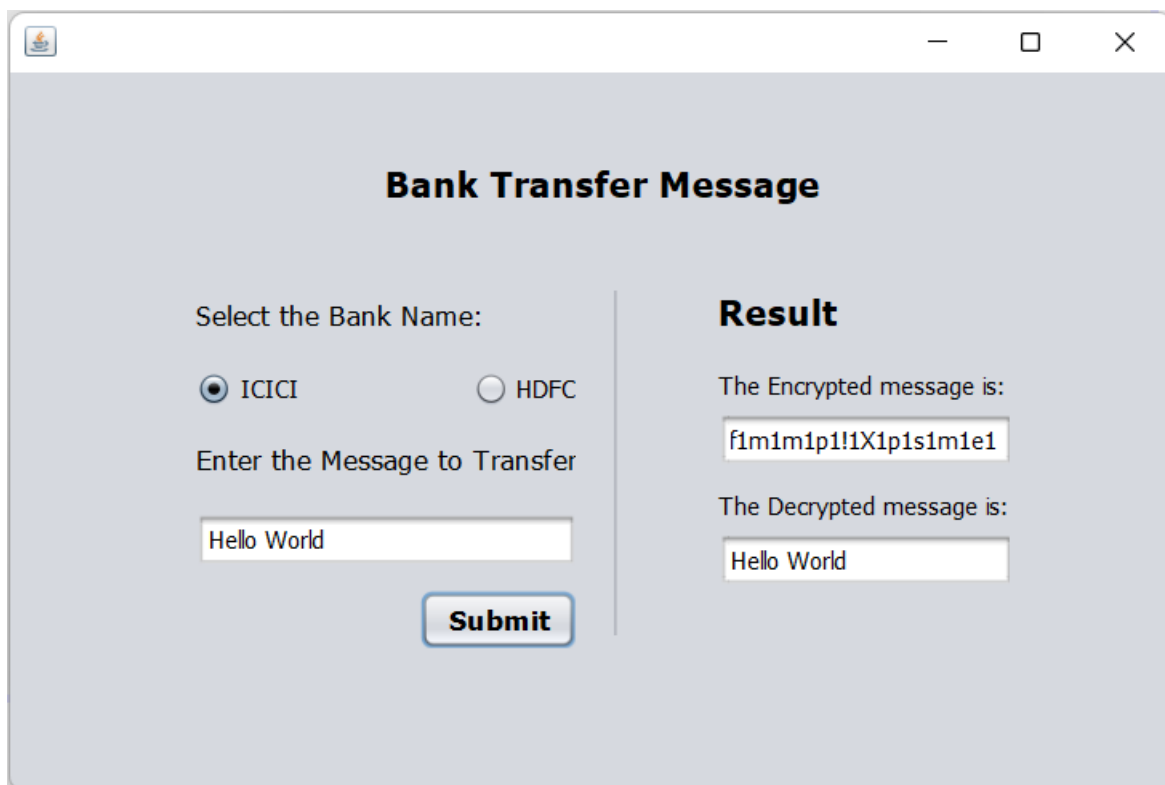
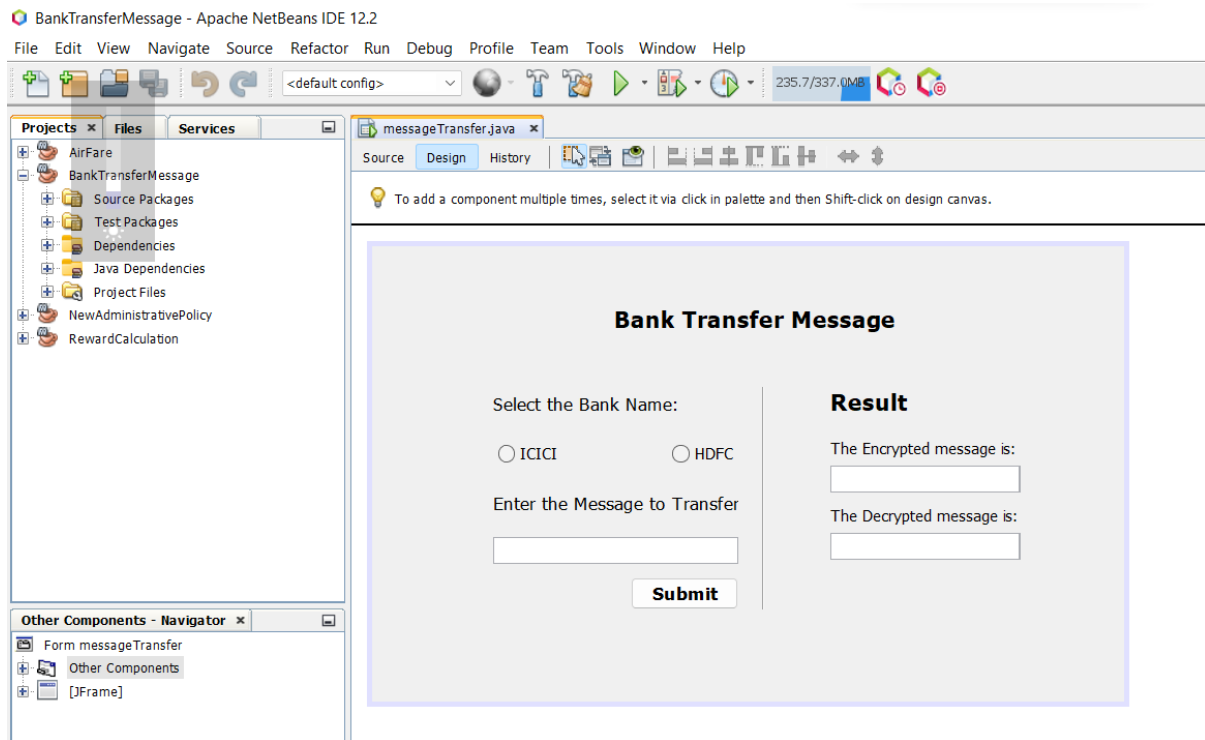
### 3. Operating System

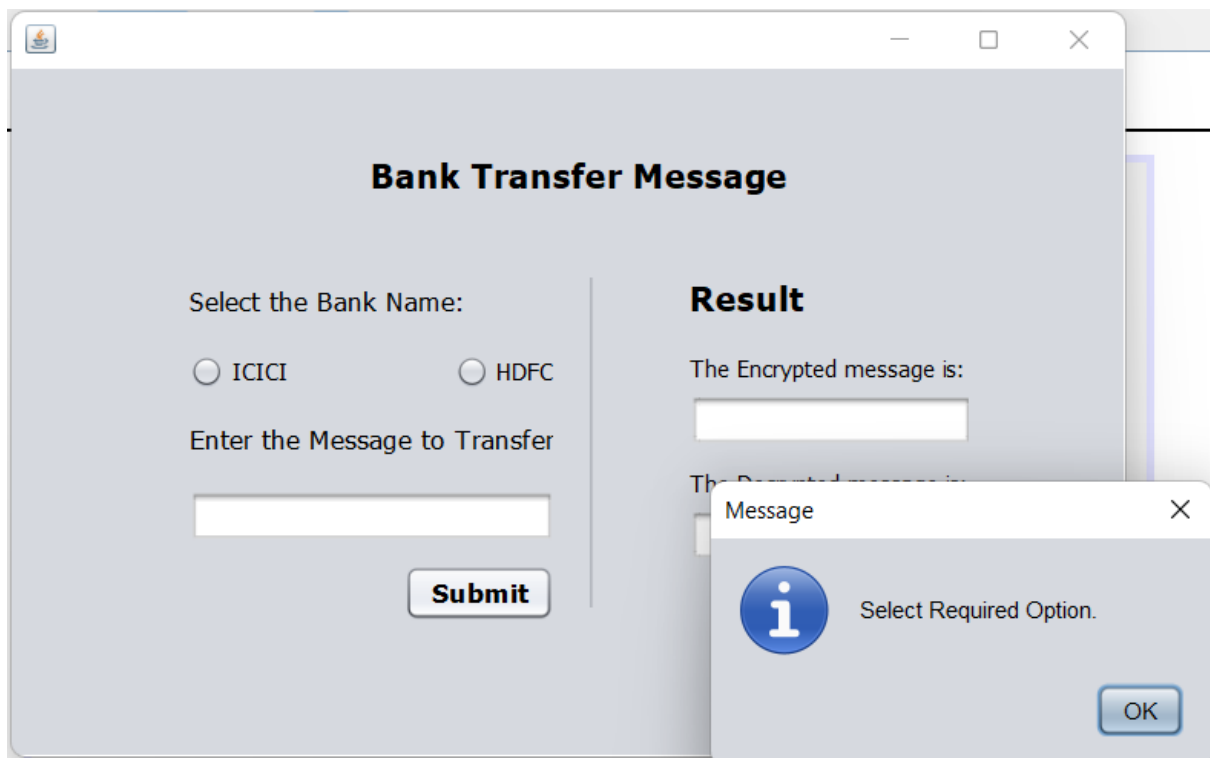
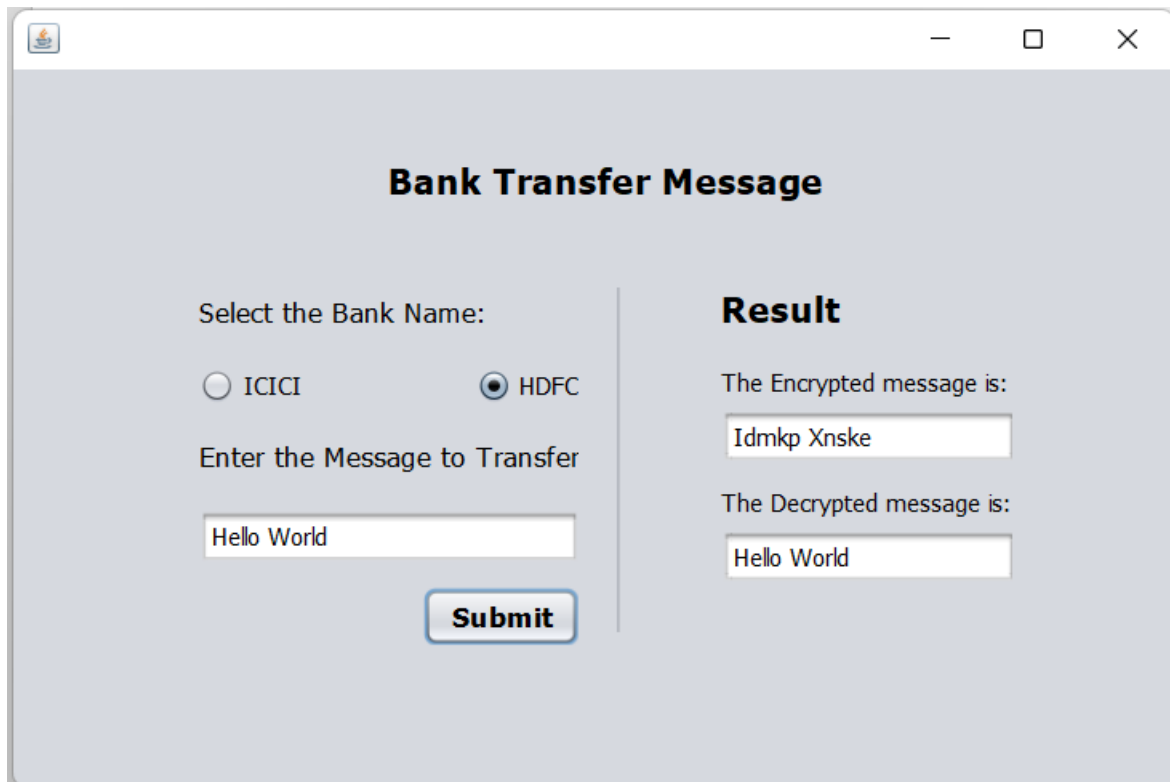
## Design

- **Flowcharts/Pseudo code**



# Implementation





## Learning Outcomes

Programming is all about data structures and algorithms. Data structures are used to hold data while algorithms are used to solve the problem using that data.

Data structures and algorithms (DSA) goes through solutions to standard problems in detail and gives you an insight into how efficient it is to use each one of them. It also teaches you the science of evaluating the efficiency of an algorithm. This enables you to choose the best of various choices.

For example, you want to search your roll number in 30000 pages of documents, for that you have choices like Linear search, Binary search, etc.

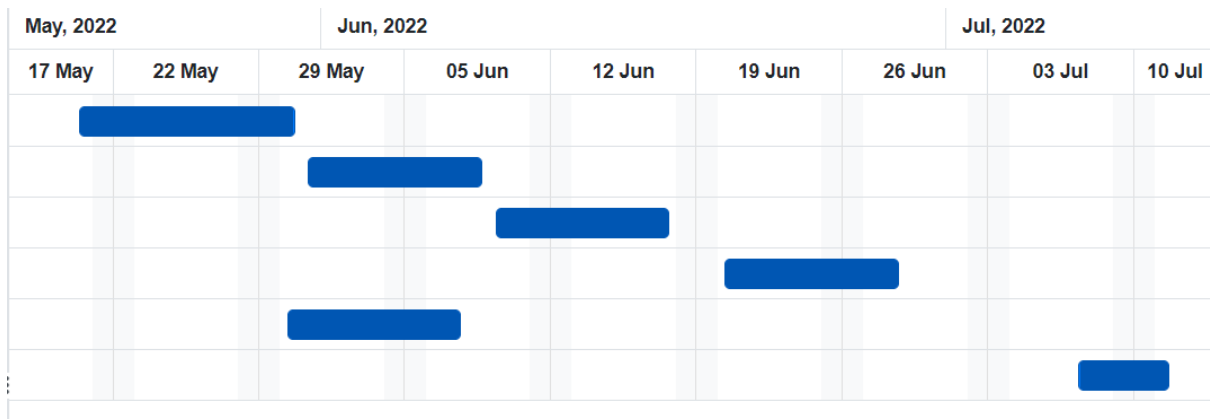
So, the more efficient way will be Binary search for searching something in a huge number of data. So, if you know the DSA, you can solve any problem efficiently.

The main use of DSA is to make your code scalable because

- Time is precious
- Memory is expensive

## Gantt Chart

Name	Start Date	End Date	Duration	Progress %
Introduction, Mathematics	May 20, 2022	May 30, 2022	7 days	100
Bit Magic, Recursion, Arrays	May 31, 2022	Jun 08, 2022	7 days	100
Searching, Sorting, Matrix	Jun 09, 2022	Jun 17, 2022	7 days	100
Hashing, Strings, Linked List	Jun 20, 2022	Jun 28, 2022	7 days	100
Stack, Queue, Deque, Tree, Binary Search Tree	May 30, 2022	Jun 07, 2022	7 days	100
Heap, Graph, Greedy	Jul 07, 2022	Jul 11, 2022	3 days	100





## **Bibliography**

- Google
- GeeksforGeeks
- Figma
- java.com
- YouTube

<https://github.com/sauravhathi>

<https://github.com/sauravhathi>