DATA STRUCTURE AND ALGORITHM

GROUP ASSIGNMENT

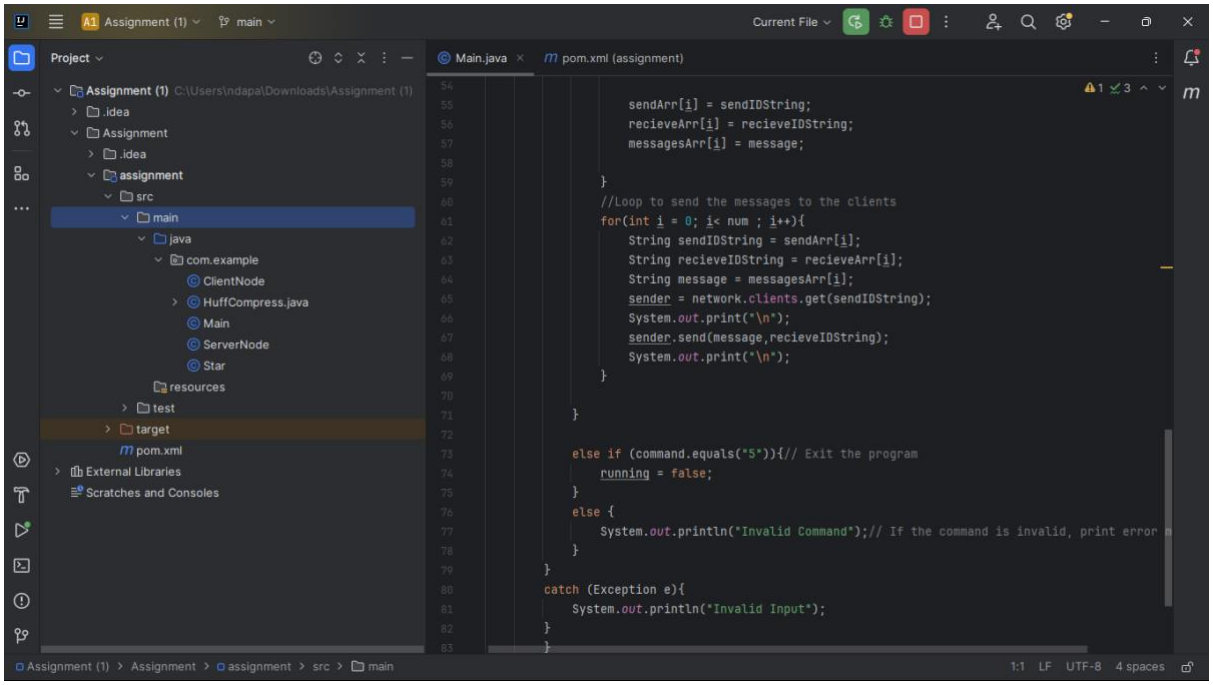
PT

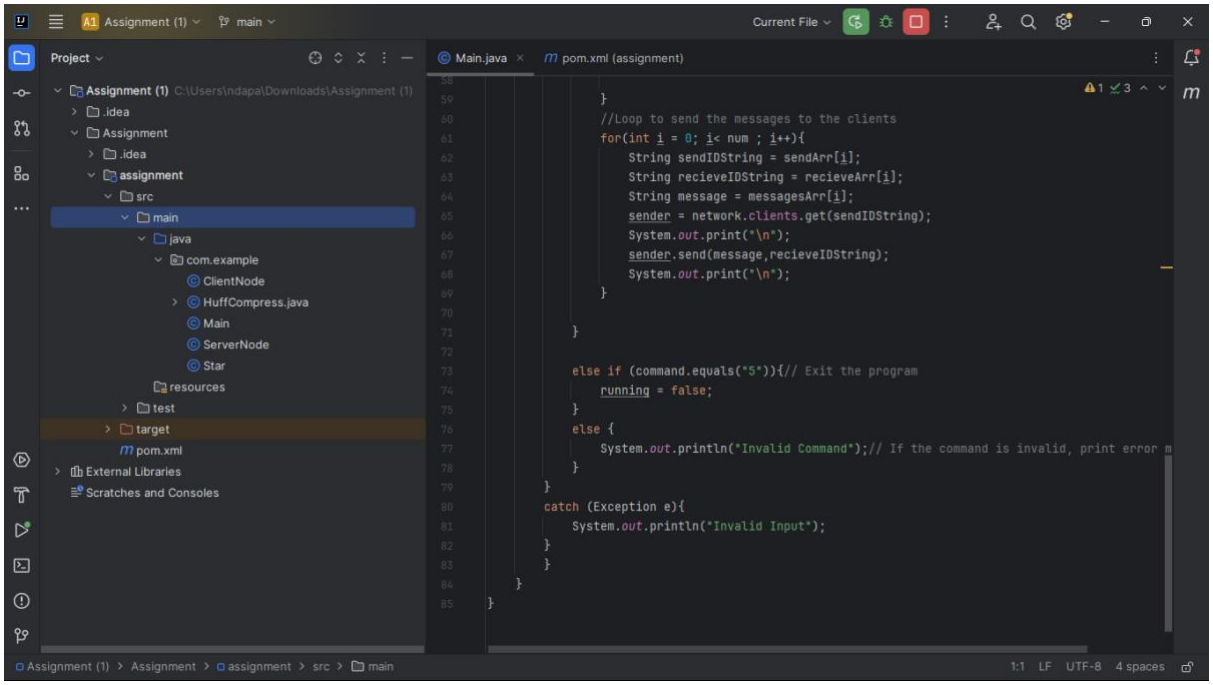
NANES	**STUDENT NO**
Nelumbu Paskaris	221061010
Ndapanda Andrade	220022410
David Ndantsi	219068119
kufanga Gerald	22046465

Main

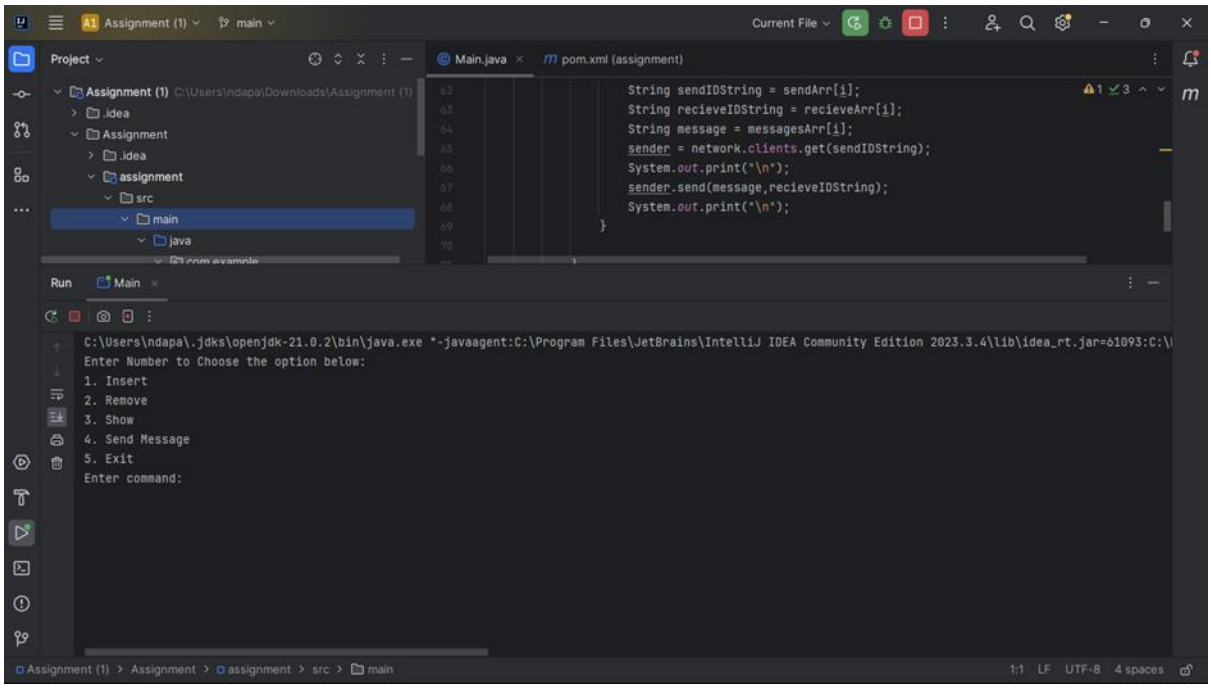
```java
package com.example;

import java.util.*;

Ndapanda Andrade 220022410
public class Main {

    Ndapanda Andrade 220022410
    public static void main(String[] args) {
        Star network = new Star();
        boolean running = true;
        Scanner input = new Scanner(System.in);
        ClientNode sender;

        //Arrays to store the users input
        String[] sendArr = new String[10];
        String[] recieveArr = new String[10];
        String[] messagesArr = new String[10];
        //Will keep the program in a loop until the user exits
        while (running){
            //Menu
            System.out.println("Enter Number to Choose the option below:");
            System.out.println("1. Insert \n2. Remove \n3. Show \n4. Send Message \n5. Exit");
            System.out.println("Enter command: ");
            try{
            String command = input.next();

            //If statement to execute the command
            if (command.equals("1")){// Insert a new client to the network
                System.out.println("Enter ID: ");
```

```java
                System.out.println("Enter ID: ");
                    String id = input.next();
                    network.insert(id);
            }
            else if (command.equals("2")){// Remove a client from the network
                System.out.println("Enter ID: ");
                    String id = input.next();
                    network.remove(id);
            }
            else if (command.equals("3")){// Show all the clients in the network
                System.out.println("Clients in the network: ");
                network.show();
            }
            else if (command.equals("4")){// Send a messages to a client
                System.out.println("Enter number of messages to send: ");
                int num = input.nextInt();
                //Loop to get the sender, reciever, and message
                for(int i = 0; i < num; i++){
                    System.out.println("Enter sender ID: ");
                    String sendIDString = input.next();

                    System.out.println("Enter recieve ID: ");
                    String recieveIDString = input.next();
                    System.out.println("Enter message: ");
                    input.nextLine();
                    String message = input.nextLine();

                    sendArr[i] = sendIDString;
                    recieveArr[i] = recieveIDString;
                    messagesArr[i] = message;
```


Assignment (1)
main
Current File
Project
Main.java
pom.xml (assignment)
Assignment (1) C:\Users\ndapa\Downloads\Assignment (1)
.idea
Assignment
.idea
assignment
src
main
java
com.example
ClientNode
HuffCompress.java
Main
ServerNode
Star
resources
test
target
pom.xml
External Libraries
Scratches and Consoles
sendArr[i] = sendIDString;
recieveArr[i] = recieveIDString;
messagesArr[i] = message;
}
//Loop to send the messages to the clients
for(int i = 0; i< num ; i++){
String sendIDString = sendArr[i];
String recieveIDString = recieveArr[i];
String message = messagesArr[i];
sender = network.clients.get(sendIDString);
System.out.print("\n");
sender.send(message,recieveIDString);
System.out.print("\n");
}
}
else if (command.equals("5")){// Exit the program
running = false;
}
else {
System.out.println("Invalid Command");// If the command is invalid, print error m
}
}
catch (Exception e){
System.out.println("Invalid Input");
}
}
Assignment (1) > Assignment > assignment > src > main
1:1 LF UTF-8 4 spaces

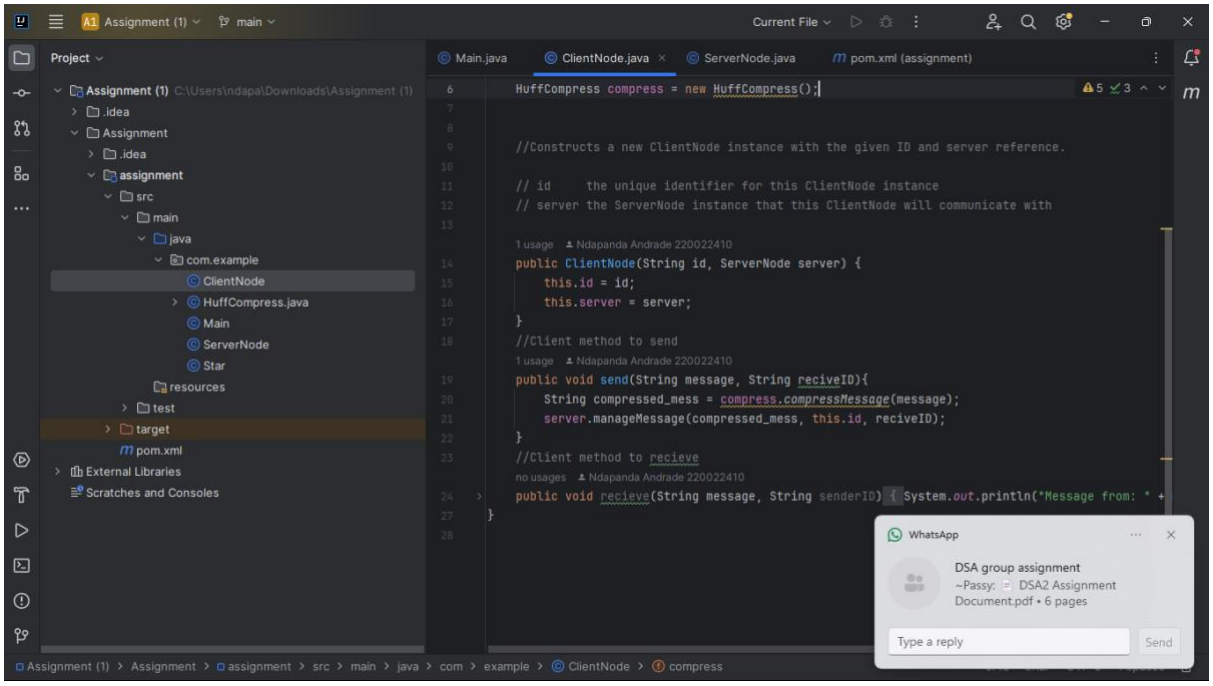

Assignment (1)
main
Current File
Project
Main.java
pom.xml (assignment)
Assignment (1) C:\Users\ndapa\Downloads\Assignment (1)
.idea
Assignment
.idea
assignment
src
main
java
com.example
ClientNode
HuffCompress.java
Main
ServerNode
Star
resources
test
target
pom.xml
External Libraries
Scratches and Consoles
}
//Loop to send the messages to the clients
for(int i = 0; i< num ; i++){
String sendIDString = sendArr[i];
String recieveIDString = recieveArr[i];
String message = messagesArr[i];
sender = network.clients.get(sendIDString);
System.out.print("\n");
sender.send(message,recieveIDString);
System.out.print("\n");
}
}
else if (command.equals("5")){// Exit the program
running = false;
}
else {
System.out.println("Invalid Command");// If the command is invalid, print error m
}
}
catch (Exception e){
System.out.println("Invalid Input");
}
}
}
}
Assignment (1) > Assignment > assignment > src > main
1:1 LF UTF-8 4 spaces

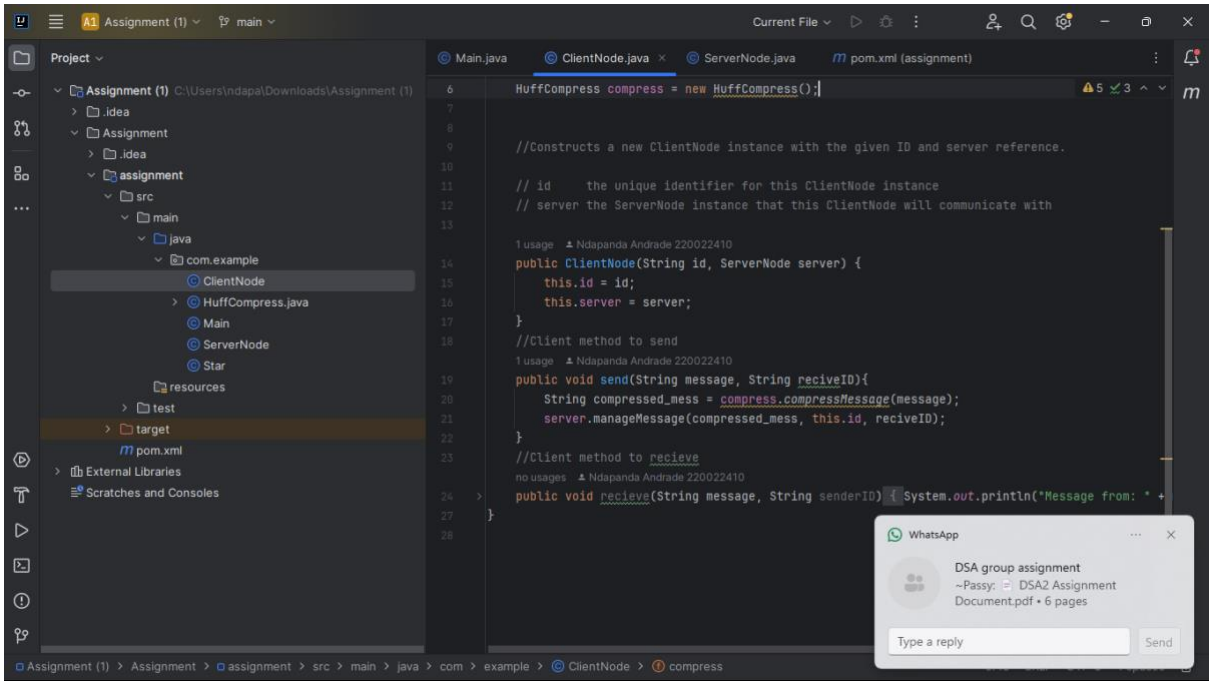
OUTPUT



JAVA CODE

ClientNode






```
package com.example;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Star network = new Star();
        boolean running = true;
        Scanner input = new Scanner(System.in);
```

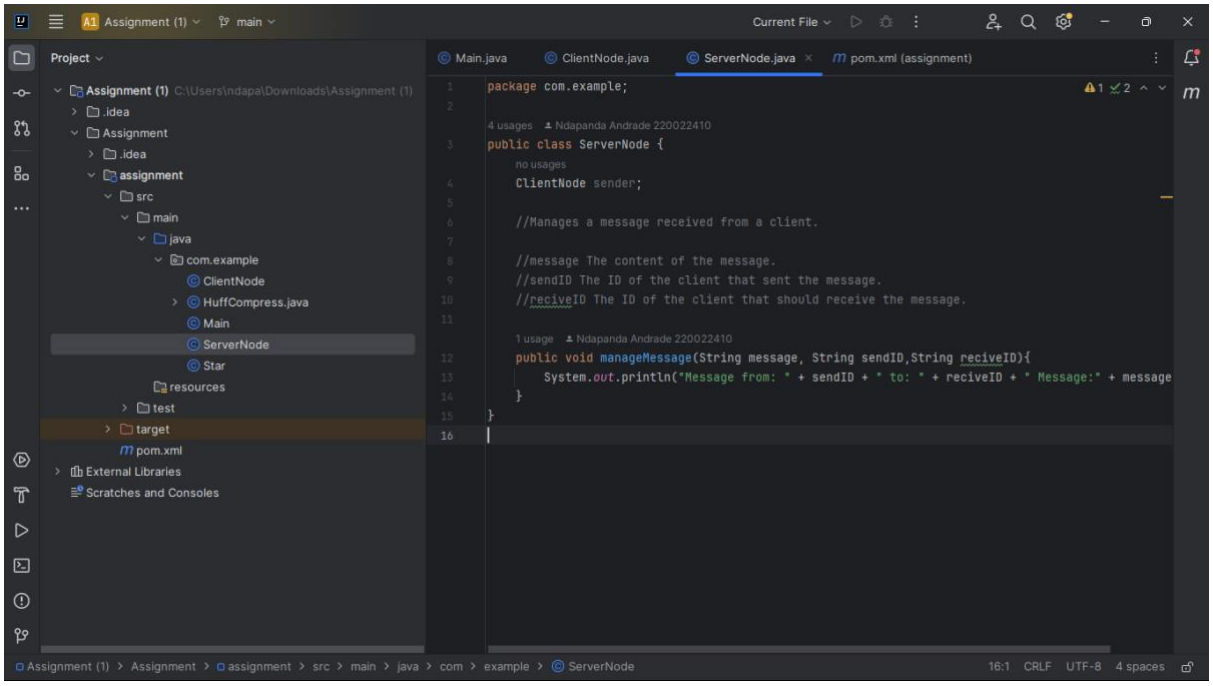
```
    ClientNode sender;

    //Arrays to store the users input
    String[] sendArr = new String[10];
    String[] recieveArr = new String[10];
    String[] messagesArr = new String[10];
    //Will keep the program in a loop until the user exits
    while (running){
      //Menu
      System.out.println("Enter Number to Choose the option below:");
      System.out.println("1. Insert \n2. Remove \n3. Show \n4. Send Message \n5.
Exit");
      System.out.println("Enter command: ");
      try{
      String command = input.next();

      //If statement to execute the command
      if (command.equals("1")){// Insert a new client to the network
        System.out.println("Enter ID: ");
          String id = input.next();
          network.insert(id);
      }
      else if (command.equals("2")){// Remove a client from the network
        System.out.println("Enter ID: ");
          String id = input.next();
          network.remove(id);
      }
      else if (command.equals("3")){// Show all the clients in the network
        System.out.println("Clients in the network: ");
        network.show();
      }
      else if (command.equals("4")){// Send a messages to a client
        System.out.println("Enter number of messages to send: ");
        int num = input.nextInt();
        //Loop to get the sender, reciever, and message
        for(int i = 0; i < num; i++){
          System.out.println("Enter sender ID: ");
          String sendIDString = input.next();

          System.out.println("Enter recieve ID: ");
          String recieveIDString = input.next();
```

```
            System.out.println("Enter message: ");
            input.nextLine();
            String message = input.nextLine();

            sendArr[i] = sendIDString;
            recieveArr[i] = recieveIDString;
            messagesArr[i] = message;

          }
          //Loop to send the messages to the clients
          for(int i = 0; i< num ; i++){
            String sendIDString = sendArr[i];
            String recieveIDString = recieveArr[i];
            String message = messagesArr[i];
            sender = network.clients.get(sendIDString);
            System.out.print("\n");
            sender.send(message,recieveIDString);
            System.out.print("\n");
          }

        }

        else if (command.equals("5")){// Exit the program
          running = false;
        }
        else {
          System.out.println("Invalid Command");// If the command is invalid, print error
message
        }
      }
      catch (Exception e){
        System.out.println("Invalid Input");
      }
      }
    }
}
```

SeverNodeClass

```
package com.example;

public class ClientNode {
   String id;
   ServerNode server;
   HuffCompress compress = new HuffCompress();


   //Constructs a new ClientNode instance with the given ID and server reference.

   // id     the unique identifier for this ClientNode instance
   // server the ServerNode instance that this ClientNode will communicate with

   public ClientNode(String id, ServerNode server) {
      this.id = id;
      this.server = server;
   }
   //Client method to send
   public void send(String message, String reciveID){
      String compressed_mess = compress.compressMessage(message);
      server.manageMessage(compressed_mess, this.id, reciveID);
   }
   //Client method to recieve
   public void recieve(String message, String senderID){
      System.out.println("Message from: " + message + "");
   }
}
```

SeverNode

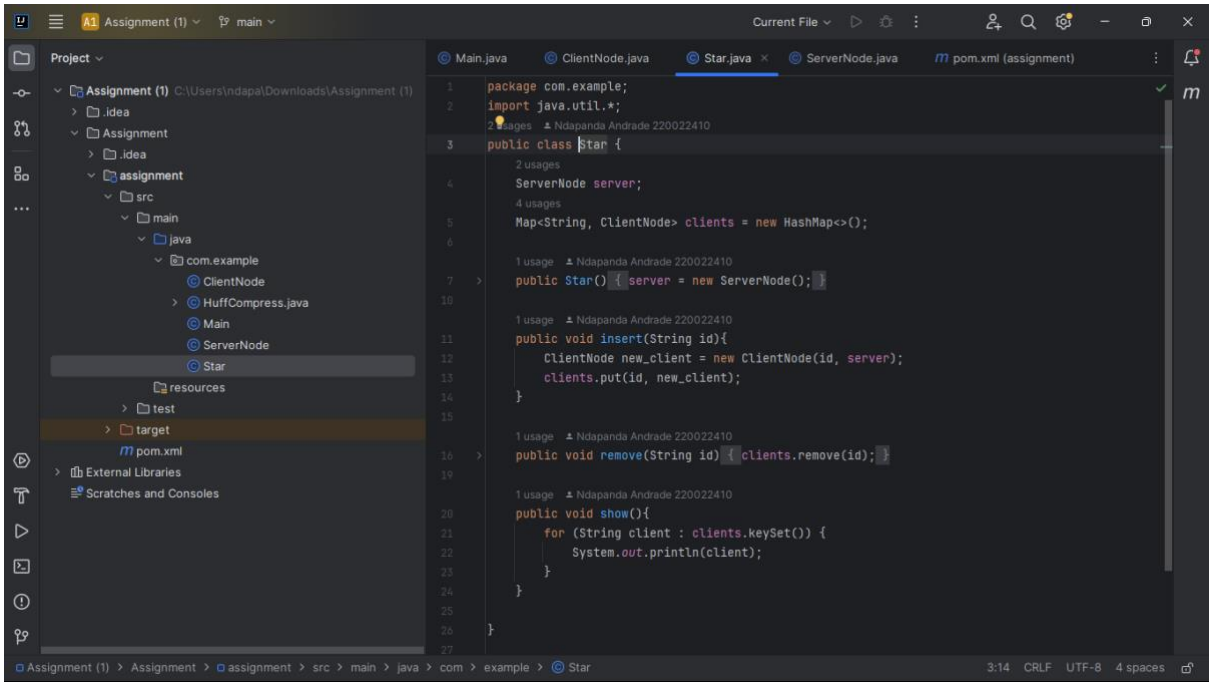



```
package com.example;

public class ServerNode {
   ClientNode sender;

   //Manages a message received from a client.

   //message The content of the message.
   //sendID The ID of the client that sent the message.
   //reciveID The ID of the client that should receive the message.

   public void manageMessage(String message, String sendID,String reciveID){
      System.out.println("Message from: " + sendID + " to: " + reciveID + " Message:" +
message);
   }
}
```

Star

```
package com.example;
import java.util.*;
public class Star {
   ServerNode server;
   Map<String, ClientNode> clients = new HashMap<>();
```

```
    public Star() {
        server = new ServerNode();
    }

    public void insert(String id){
        ClientNode new_client = new ClientNode(id, server);
        clients.put(id, new_client);
    }

    public void remove(String id){
        clients.remove(id);
    }

    public void show(){
        for (String client : clients.keySet()) {
            System.out.println(client);
        }
    }

}
```

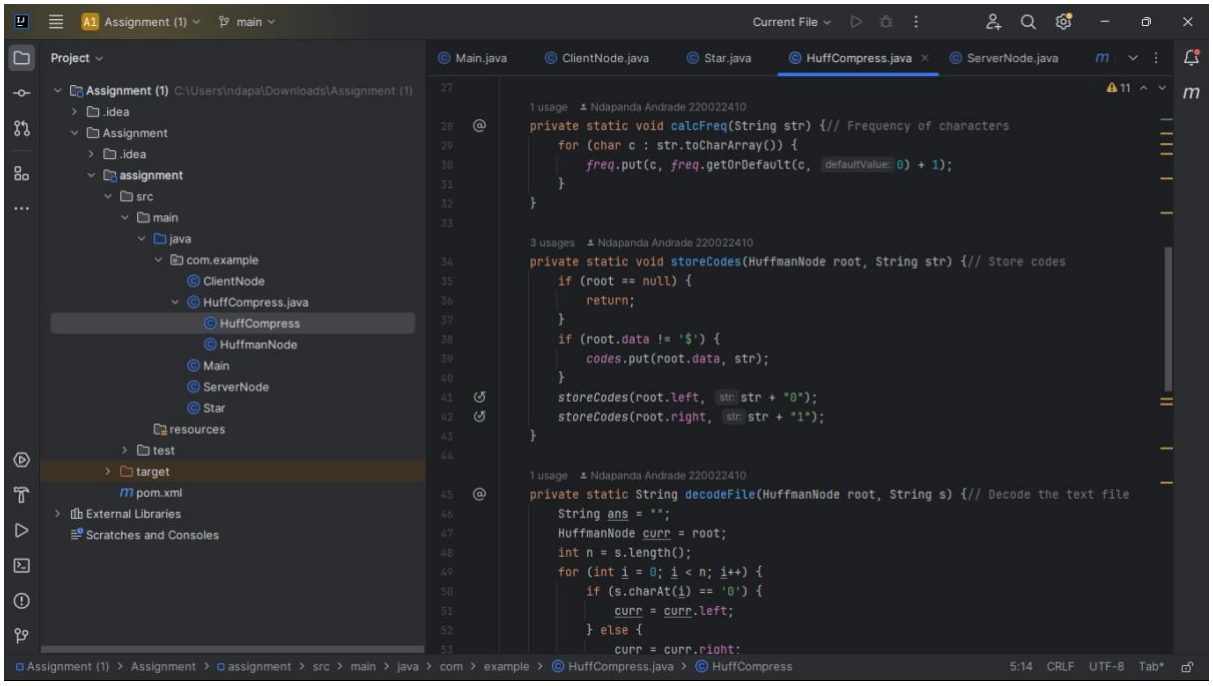


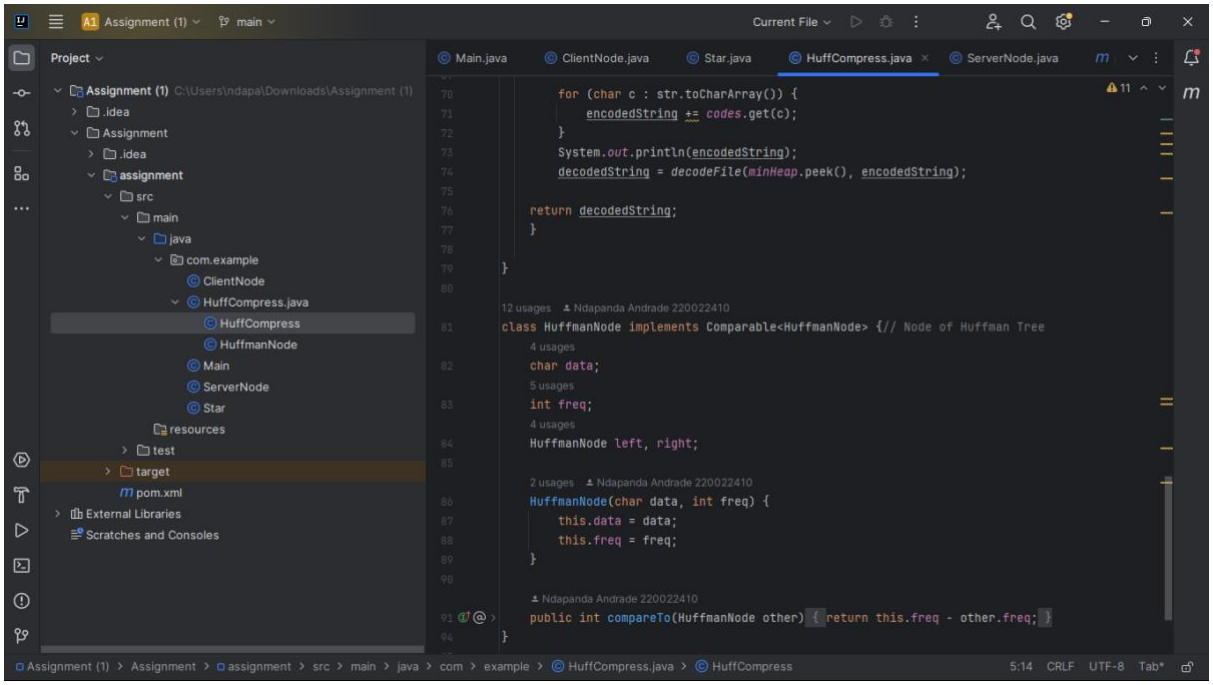
Data Compression

Compression

```java
package com.example;
import ...

2 usages  Ndapanda Andrade 220022410
public class HuffCompress {

    2 usages
    private static Map<Character, String> codes = new HashMap<>();
    3 usages
    private static Map<Character, Integer> freq = new HashMap<>();
    7 usages
    private static PriorityQueue<HuffmanNode> minHeap = new PriorityQueue<>();



    1 usage  Ndapanda Andrade 220022410
    private static void HuffmanCodes(int size) {// Huffman Codes
        for (Entry<Character, Integer> entry : freq.entrySet()) {
            minHeap.add(new HuffmanNode(entry.getKey(), entry.getValue()));
        }
        while (minHeap.size() != 1) {
            HuffmanNode left = minHeap.poll();
            HuffmanNode right = minHeap.poll();
            HuffmanNode top = new HuffmanNode( data: '$',  freq: left.freq + right.freq);
            top.left = left;
            top.right = right;
            minHeap.add(top);
        }
        storeCodes(minHeap.peek(),  str: "");
    }
```

```java
        storeCodes(minHeap.peek(),  str: "");
    }

    1 usage  Ndapanda Andrade 220022410
    private static void calcFreq(String str) {// Frequency of characters
        for (char c : str.toCharArray()) {
            freq.put(c, freq.getOrDefault(c,  defaultValue: 0) + 1);
        }
    }

    3 usages  Ndapanda Andrade 220022410
    private static void storeCodes(HuffmanNode root, String str) {// Store codes
        if (root == null) {
            return;
        }
        if (root.data != '$') {
            codes.put(root.data, str);
        }
        storeCodes(root.left,  str: str + "0");
        storeCodes(root.right,  str: str + "1");
    }

    1 usage  Ndapanda Andrade 220022410
    private static String decodeFile(HuffmanNode root, String s) {// Decode the text file
        String ans = "";
        HuffmanNode curr = root;
        int n = s.length();
        for (int i = 0; i < n; i++) {
            if (s.charAt(i) == '0') {
```





Java code

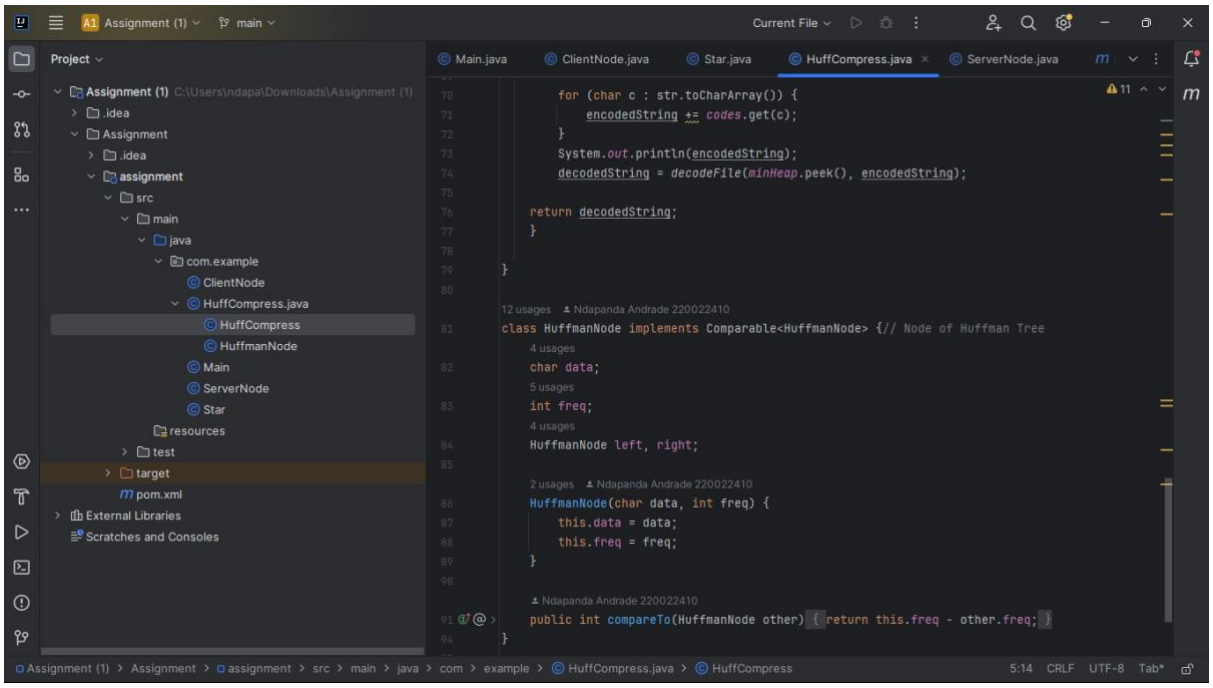
```
package com.example;
import java.util.*;
import java.util.Map.Entry;

public class HuffCompress {

    private static Map<Character, String> codes = new HashMap<>();
    private static Map<Character, Integer> freq = new HashMap<>();
    private static PriorityQueue<HuffmanNode> minHeap = new PriorityQueue<>();
```

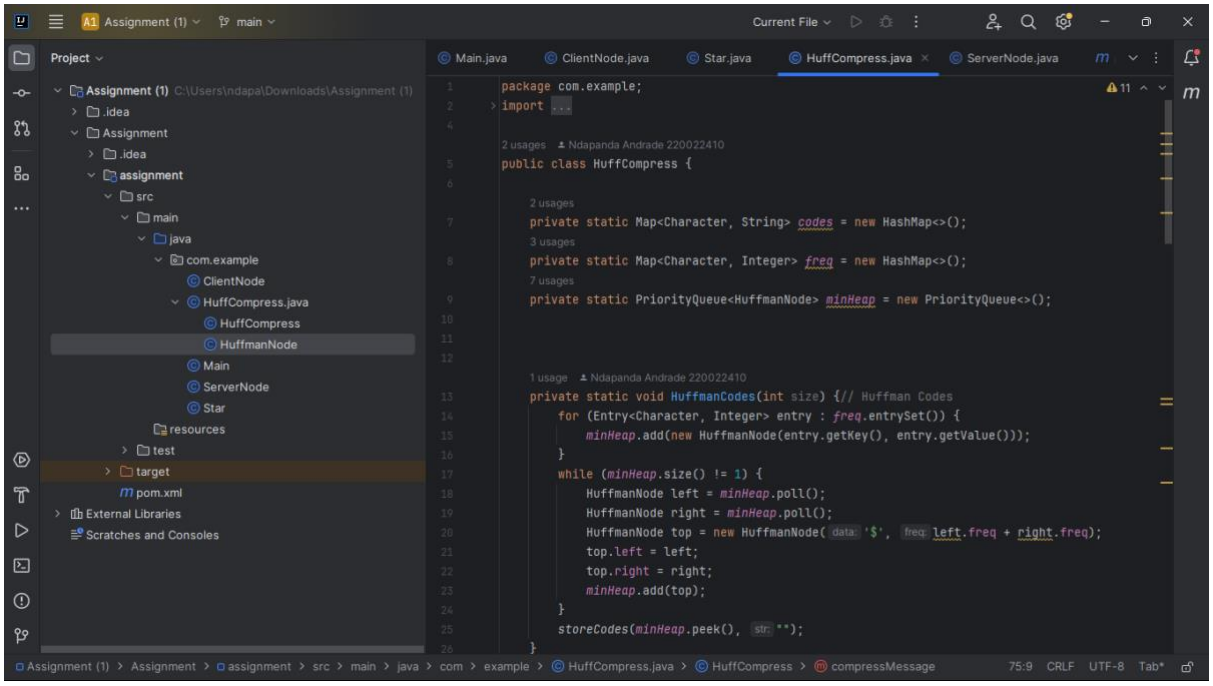
```
private static void HuffmanCodes(int size) {// Huffman Codes
  for (Entry<Character, Integer> entry : freq.entrySet()) {
    minHeap.add(new HuffmanNode(entry.getKey(), entry.getValue()));
  }
  while (minHeap.size() != 1) {
    HuffmanNode left = minHeap.poll();
    HuffmanNode right = minHeap.poll();
    HuffmanNode top = new HuffmanNode('$', left.freq + right.freq);
    top.left = left;
    top.right = right;
    minHeap.add(top);
  }
  storeCodes(minHeap.peek(), "");
}

private static void calcFreq(String str) {// Frequency of characters
  for (char c : str.toCharArray()) {
    freq.put(c, freq.getOrDefault(c, 0) + 1);
  }
}

private static void storeCodes(HuffmanNode root, String str) {// Store codes
  if (root == null) {
    return;
  }
  if (root.data != '$') {
    codes.put(root.data, str);
  }
  storeCodes(root.left, str + "0");
  storeCodes(root.right, str + "1");
}

private static String decodeFile(HuffmanNode root, String s) {// Decode the text file
  String ans = "";
  HuffmanNode curr = root;
  int n = s.length();
  for (int i = 0; i < n; i++) {
    if (s.charAt(i) == '0') {
```

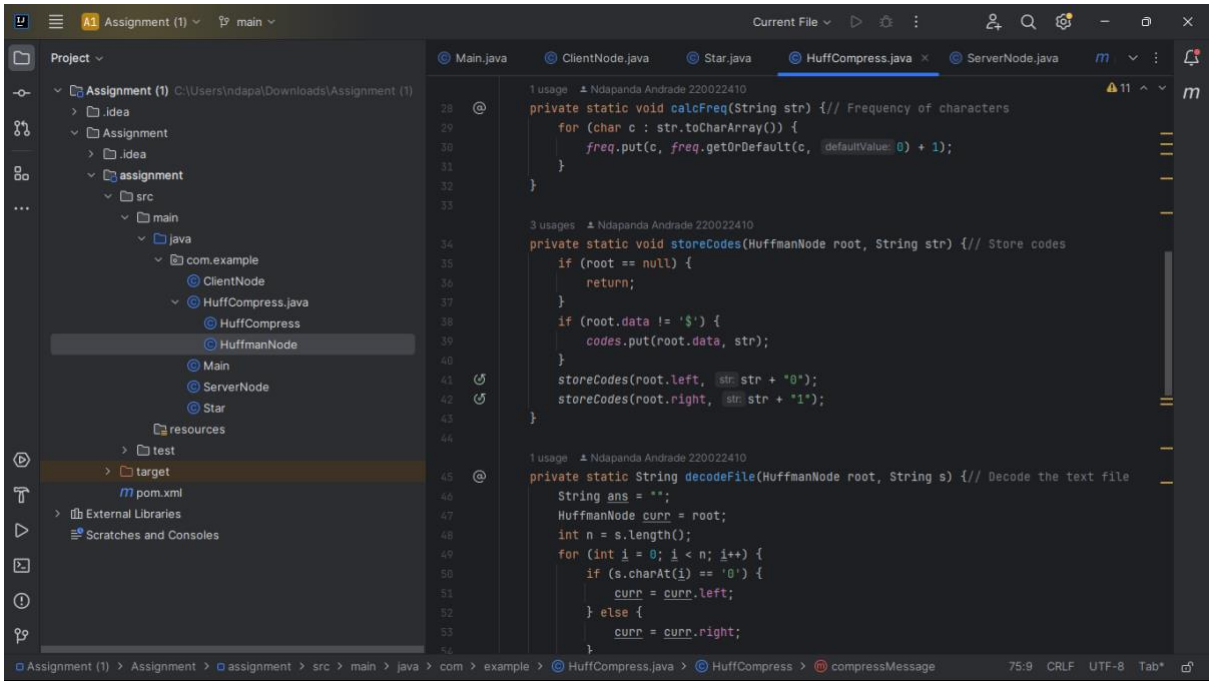
```
        curr = curr.left;
      } else {
        curr = curr.right;
      }
      if (curr.left == null && curr.right == null) {
        ans += curr.data;
        curr = root;
      }
    }
    return ans + '\0';
  }

  public static String compressMessage(String message) {// Compresses the message
    String str = message;
    String encodedString = "";
    String decodedString = "";
    calcFreq(str);
    HuffmanCodes(str.length());

    for (char c : str.toCharArray()) {
      encodedString += codes.get(c);
    }
    System.out.println(encodedString);
    decodedString = decodeFile(minHeap.peek(), encodedString);

  return decodedString;
  }

}

class HuffmanNode implements Comparable<HuffmanNode> {// Node of Huffman Tree
  char data;
  int freq;
  HuffmanNode left, right;

  HuffmanNode(char data, int freq) {
    this.data = data;
    this.freq = freq;
  }

  public int compareTo(HuffmanNode other) {
```

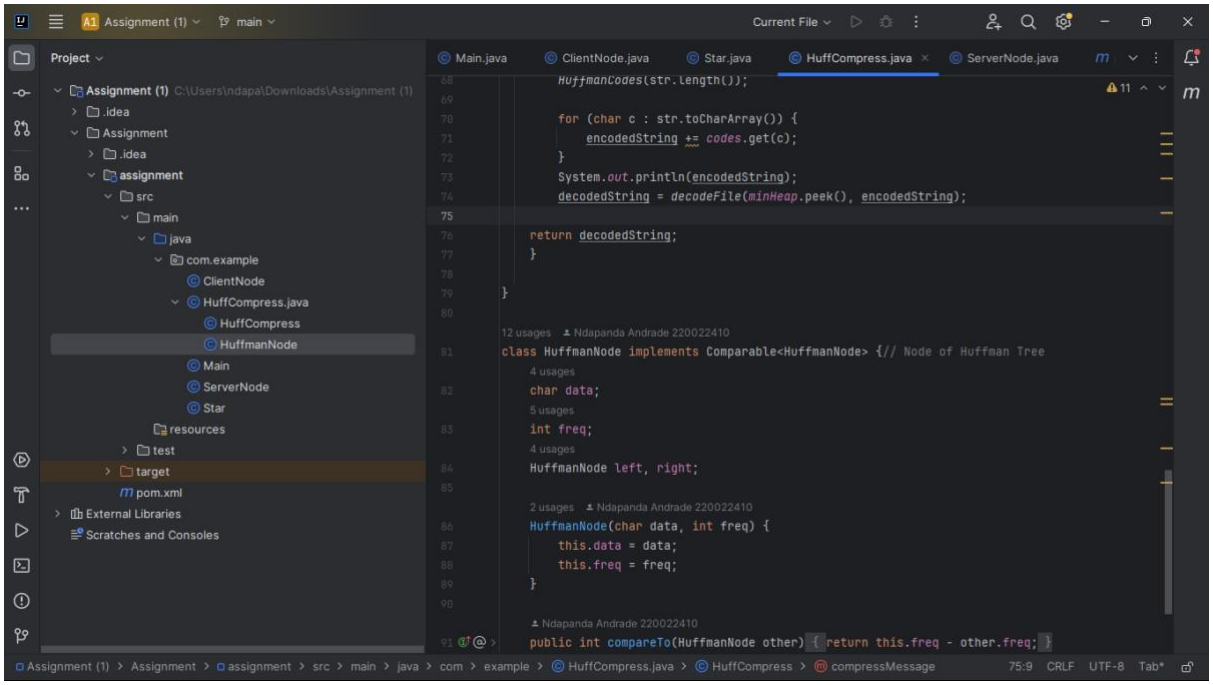
```
        return this.freq - other.freq;
    }
}
```

Node









Java code

```
package com.example;
import java.util.*;
import java.util.Map.Entry;

public class HuffCompress {

    private static Map<Character, String> codes = new HashMap<>();
```

```
private static Map<Character, Integer> freq = new HashMap<>();
private static PriorityQueue<HuffmanNode> minHeap = new PriorityQueue<>();



private static void HuffmanCodes(int size) {// Huffman Codes
  for (Entry<Character, Integer> entry : freq.entrySet()) {
    minHeap.add(new HuffmanNode(entry.getKey(), entry.getValue()));
  }
  while (minHeap.size() != 1) {
    HuffmanNode left = minHeap.poll();
    HuffmanNode right = minHeap.poll();
    HuffmanNode top = new HuffmanNode('$', left.freq + right.freq);
    top.left = left;
    top.right = right;
    minHeap.add(top);
  }
  storeCodes(minHeap.peek(), "");
}

private static void calcFreq(String str) {// Frequency of characters
  for (char c : str.toCharArray()) {
    freq.put(c, freq.getOrDefault(c, 0) + 1);
  }
}

private static void storeCodes(HuffmanNode root, String str) {// Store codes
  if (root == null) {
    return;
  }
  if (root.data != '$') {
    codes.put(root.data, str);
  }
  storeCodes(root.left, str + "0");
  storeCodes(root.right, str + "1");
}

private static String decodeFile(HuffmanNode root, String s) {// Decode the text file
  String ans = "";
  HuffmanNode curr = root;
  int n = s.length();
```

```
    for (int i = 0; i < n; i++) {
      if (s.charAt(i) == '0') {
        curr = curr.left;
      } else {
        curr = curr.right;
      }
      if (curr.left == null && curr.right == null) {
        ans += curr.data;
        curr = root;
      }
    }
    return ans + '\0';
  }

  public static String compressMessage(String message) {// Compresses the message
    String str = message;
    String encodedString = "";
    String decodedString = "";
    calcFreq(str);
    HuffmanCodes(str.length());

    for (char c : str.toCharArray()) {
      encodedString += codes.get(c);
    }
    System.out.println(encodedString);
    decodedString = decodeFile(minHeap.peek(), encodedString);

  return decodedString;
  }

}

class HuffmanNode implements Comparable<HuffmanNode> {// Node of Huffman Tree
  char data;
  int freq;
  HuffmanNode left, right;

  HuffmanNode(char data, int freq) {
    this.data = data;
    this.freq = freq;
  }
```

```
  public int compareTo(HuffmanNode other) {
    return this.freq - other.freq;
  }
}
```