

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

9/12/2024

# LAB O4 Exercise

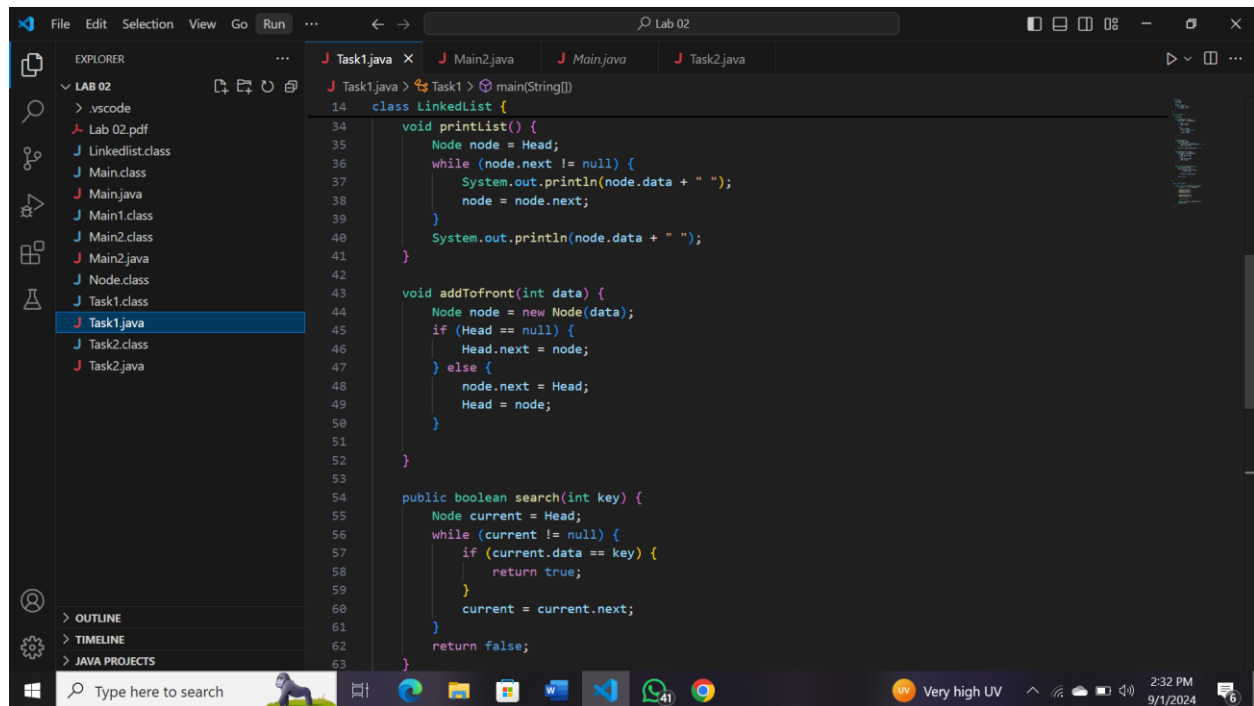
Faheem Akbar

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and curve upwards and to the right.

CMS ID: 023-23-0365

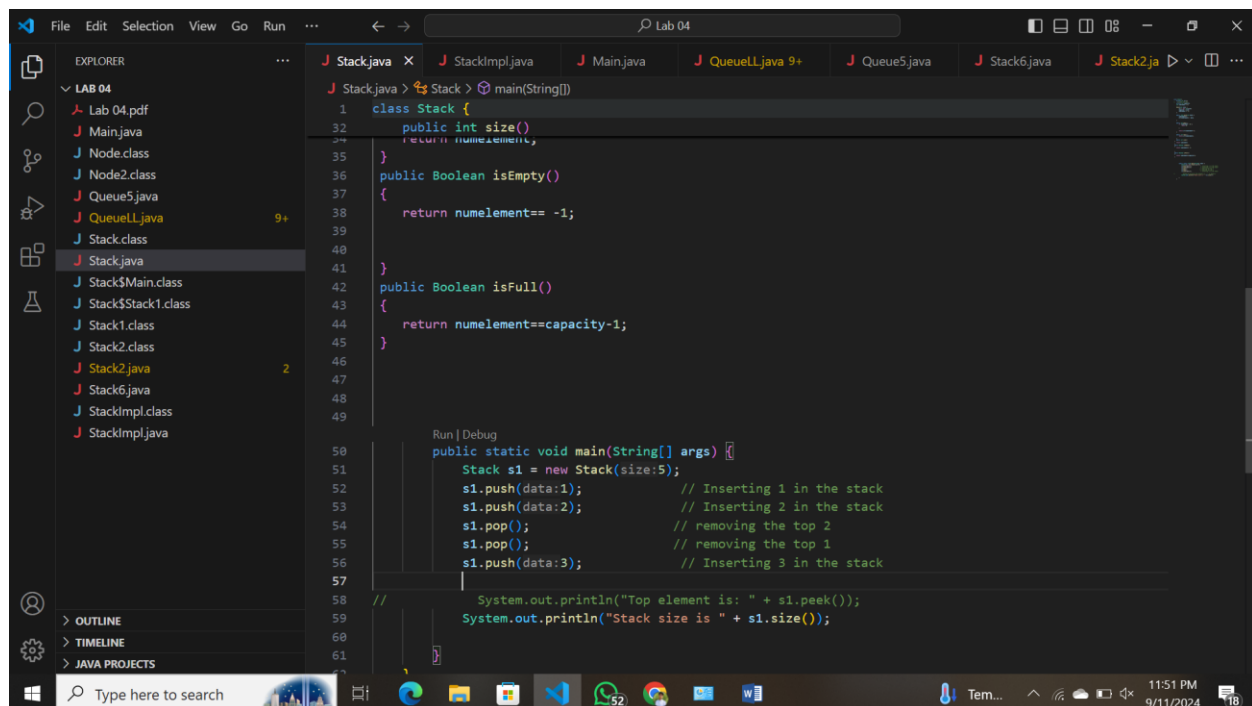
DSA LAB 04 || SUBMITTED TO MA'AM MARINA GUL

1. Stack using array: Understand provided code and implement all required methods in Stack.  
Stack Code is given below:



The screenshot shows a Visual Studio Code editor window with a project named 'Lab 02'. The Explorer sidebar on the left lists files including 'Task1.java', which is currently selected. The main editor displays the code for the 'LinkedList' class. The code includes methods for printing the list, adding a node to the front, and searching for a key. The taskbar at the bottom shows the system clock as 2:32 PM on 9/1/2024.

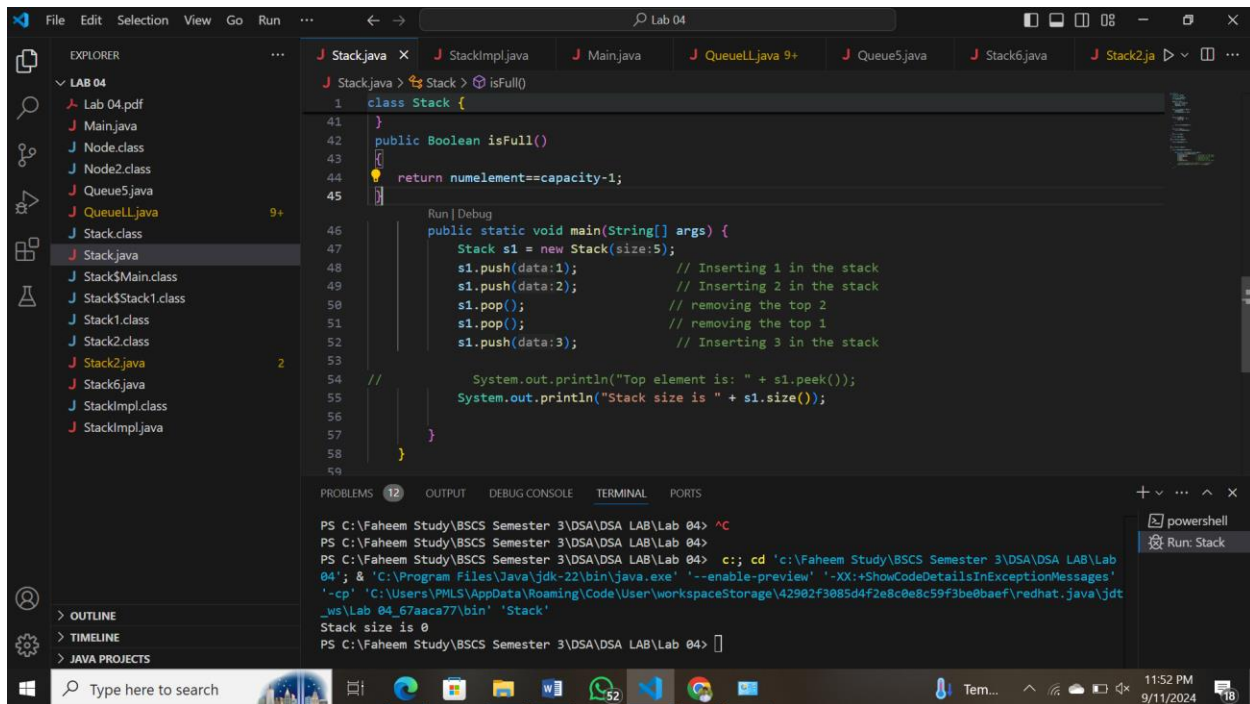
```
14 class LinkedList {
34     void printList() {
35         Node node = Head;
36         while (node.next != null) {
37             System.out.println(node.data + " ");
38             node = node.next;
39         }
40         System.out.println(node.data + " ");
41     }
42
43     void addToFront(int data) {
44         Node node = new Node(data);
45         if (Head == null) {
46             Head.next = node;
47         } else {
48             node.next = Head;
49             Head = node;
50         }
51     }
52
53     public boolean search(int key) {
54         Node current = Head;
55         while (current != null) {
56             if (current.data == key) {
57                 return true;
58             }
59             current = current.next;
60         }
61         return false;
62     }
63 }
```



The screenshot shows a Visual Studio Code editor window with a project named 'Lab 04'. The Explorer sidebar on the left lists files including 'Stack.java', which is currently selected. The main editor displays the code for the 'Stack' class. The code includes methods for checking if the stack is empty or full, and a main method demonstrating push and pop operations. The taskbar at the bottom shows the system clock as 11:51 PM on 9/11/2024.

```
1 class Stack {
32     public int size()
33     {
34         return numelement;
35     }
36
37     public Boolean isEmpty()
38     {
39         return numelement== -1;
40     }
41
42     public Boolean isFull()
43     {
44         return numelement==capacity-1;
45     }
46
47
48
49
50     Run | Debug
51     public static void main(String[] args) {
52         Stack s1 = new Stack(size:5);
53         s1.push(data:1); // Inserting 1 in the stack
54         s1.push(data:2); // Inserting 2 in the stack
55         s1.pop(); // removing the top 2
56         s1.pop(); // removing the top 1
57         s1.push(data:3); // Inserting 3 in the stack
58
59         System.out.println("Top element is: " + s1.peek());
60         System.out.println("Stack size is " + s1.size());
61     }
62 }
```

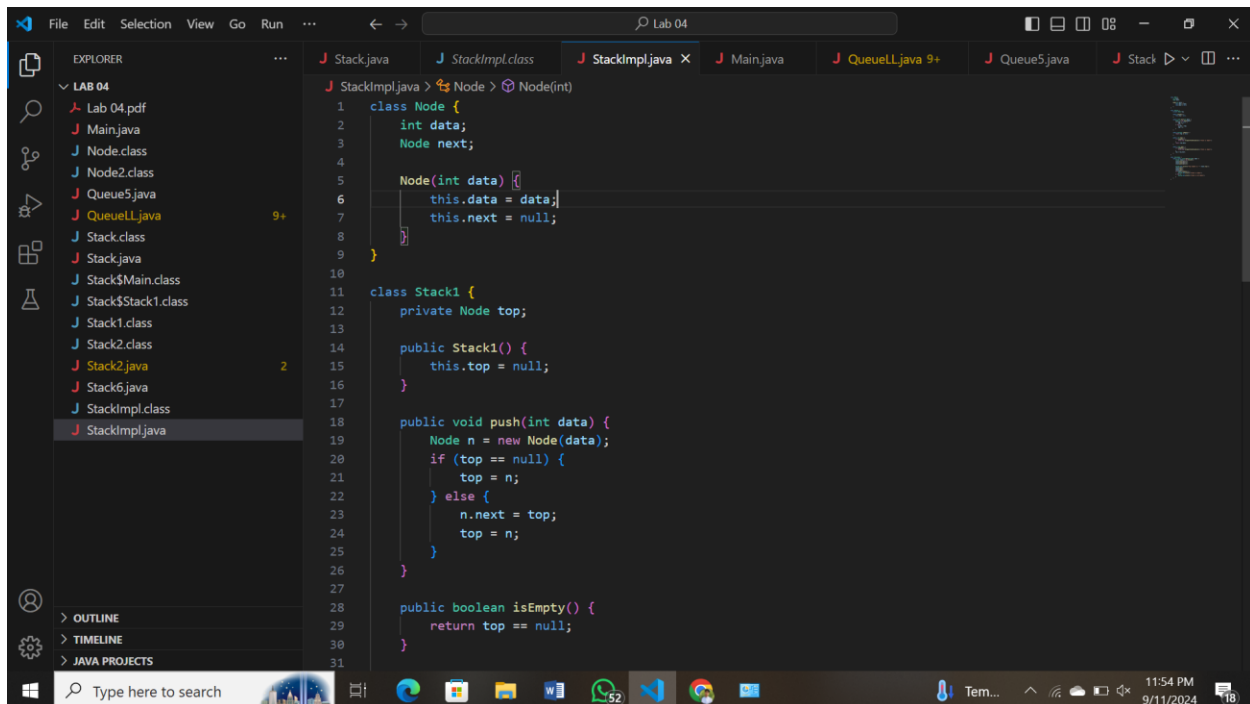
## OUTPUT 1:



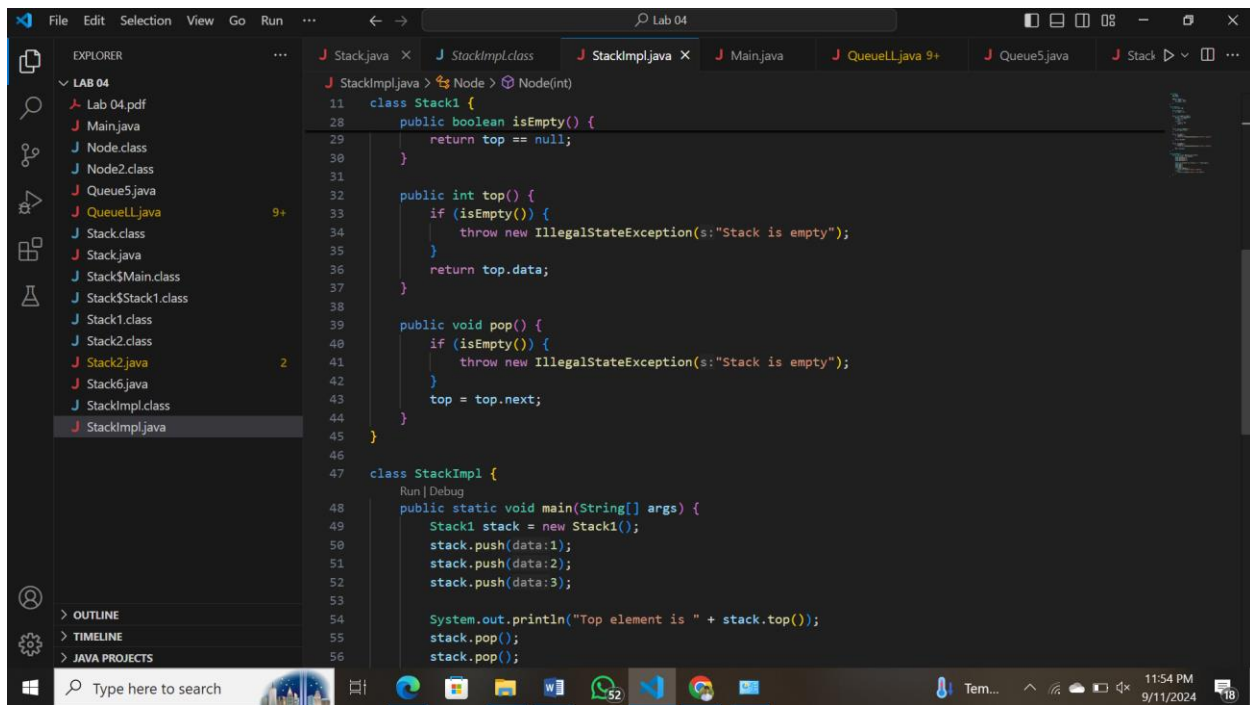
```
1 class Stack {
2     int[] arr;
3     int top;
4     int capacity;
5
6     Stack(int capacity) {
7         arr = new int[capacity];
8         top = -1;
9     }
10
11    public boolean isFull() {
12        return top == capacity - 1;
13    }
14
15    public void push(int data) {
16        if (!isFull()) {
17            arr[++top] = data;
18        }
19    }
20
21    public int pop() {
22        if (top >= 0) {
23            return arr[top--];
24        }
25        return -1;
26    }
27
28    public int peek() {
29        if (top >= 0) {
30            return arr[top];
31        }
32        return -1;
33    }
34
35    public void printStack() {
36        for (int i = 0; i <= top; i++) {
37            System.out.print(arr[i] + " ");
38        }
39        System.out.println();
40    }
41
42    public static void main(String[] args) {
43        Stack s1 = new Stack(5);
44        s1.push(1); // Inserting 1 in the stack
45        s1.push(2); // Inserting 2 in the stack
46        s1.pop();   // removing the top 2
47        s1.pop();   // removing the top 1
48        s1.push(3); // Inserting 3 in the stack
49
50        System.out.println("Top element is: " + s1.peek());
51        System.out.println("Stack size is " + s1.size());
52    }
53}
```

PS C:\Faheem Study\BSCS Semester 3\DSA\DSA LAB\Lab 04> ^C  
PS C:\Faheem Study\BSCS Semester 3\DSA\DSA LAB\Lab 04>  
PS C:\Faheem Study\BSCS Semester 3\DSA\DSA LAB\Lab 04> c:: cd 'c:\Faheem Study\BSCS Semester 3\DSA\DSA LAB\Lab 04'; & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\PMLS\AppData\Roaming\Code\User\workspaceStorage\42982f3085d4f2e8c0e8c59f3be0baef\redhat\_java\jdt\_ws\Lab\_04\_67aaca77\bin' 'Stack'  
Stack size is 0  
PS C:\Faheem Study\BSCS Semester 3\DSA\DSA LAB\Lab 04> |

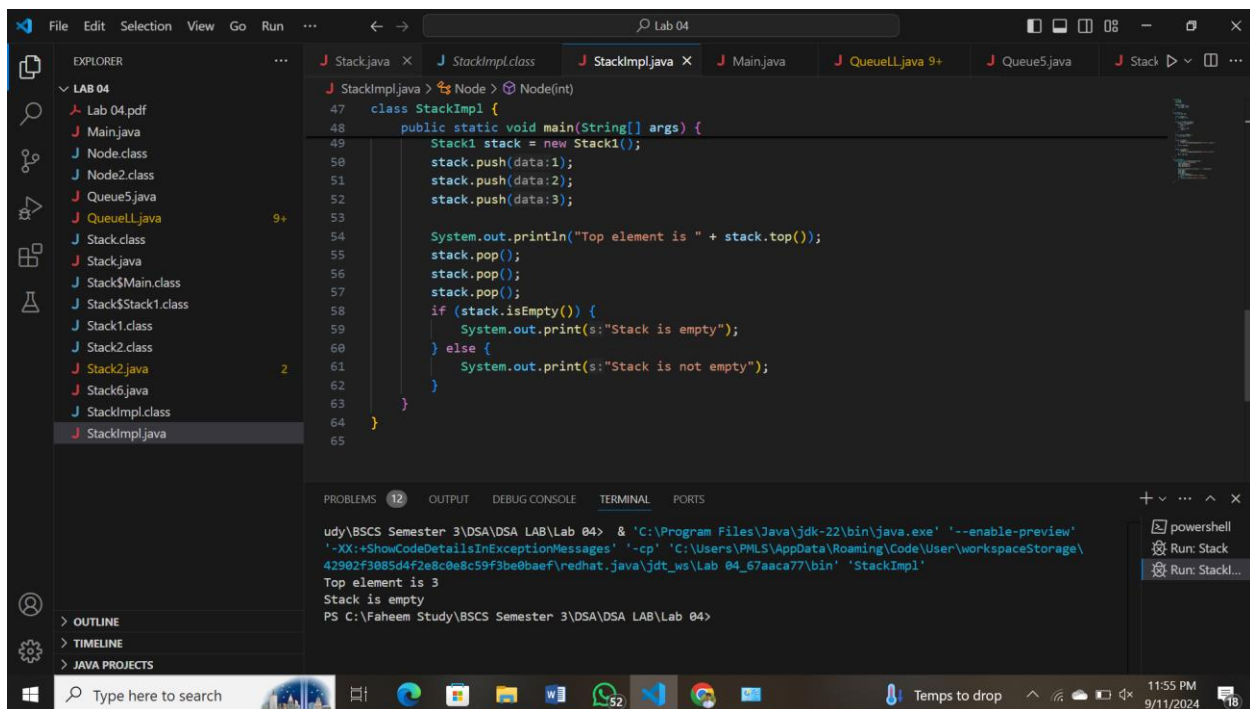
2. Stack using Linked list: Understand provided code and implement all required methods in Stack. Stack Code is given below:



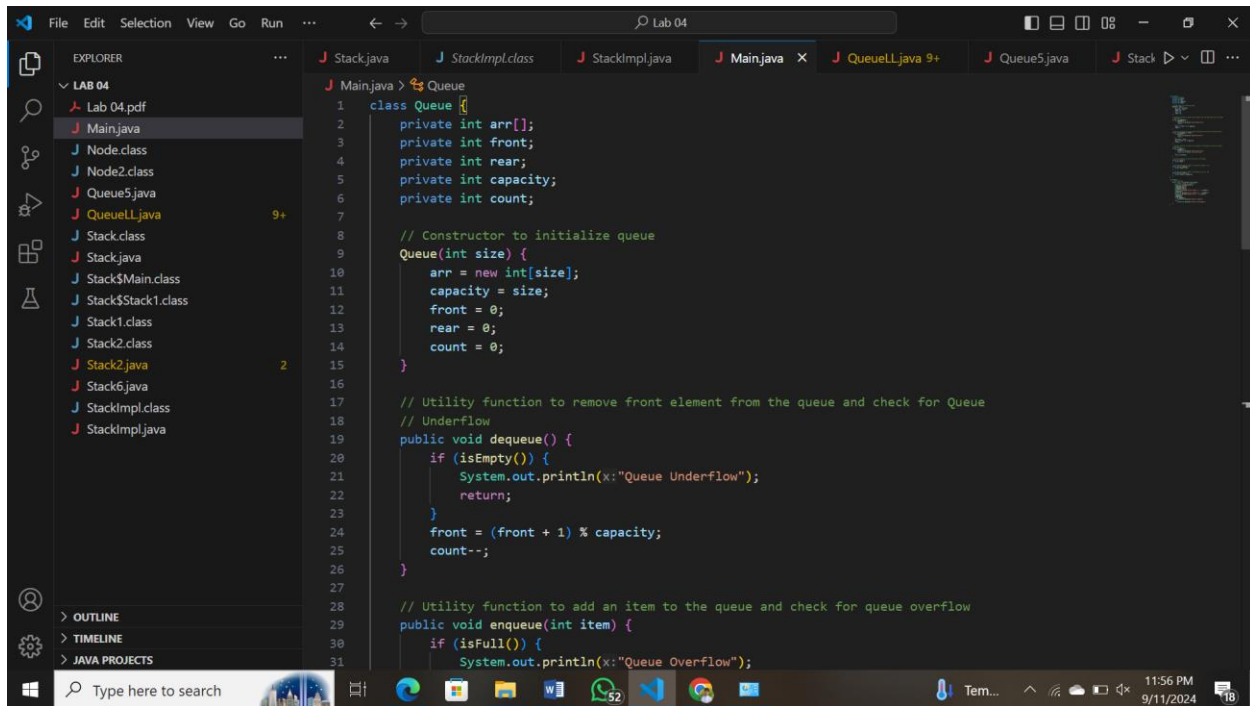
```
1 class Node {
2     int data;
3     Node next;
4
5     Node(int data) {
6         this.data = data;
7         this.next = null;
8     }
9 }
10
11 class Stack1 {
12     private Node top;
13
14     public Stack1() {
15         this.top = null;
16     }
17
18     public void push(int data) {
19         Node n = new Node(data);
20         if (top == null) {
21             top = n;
22         } else {
23             n.next = top;
24             top = n;
25         }
26     }
27
28     public boolean isEmpty() {
29         return top == null;
30     }
31 }
```



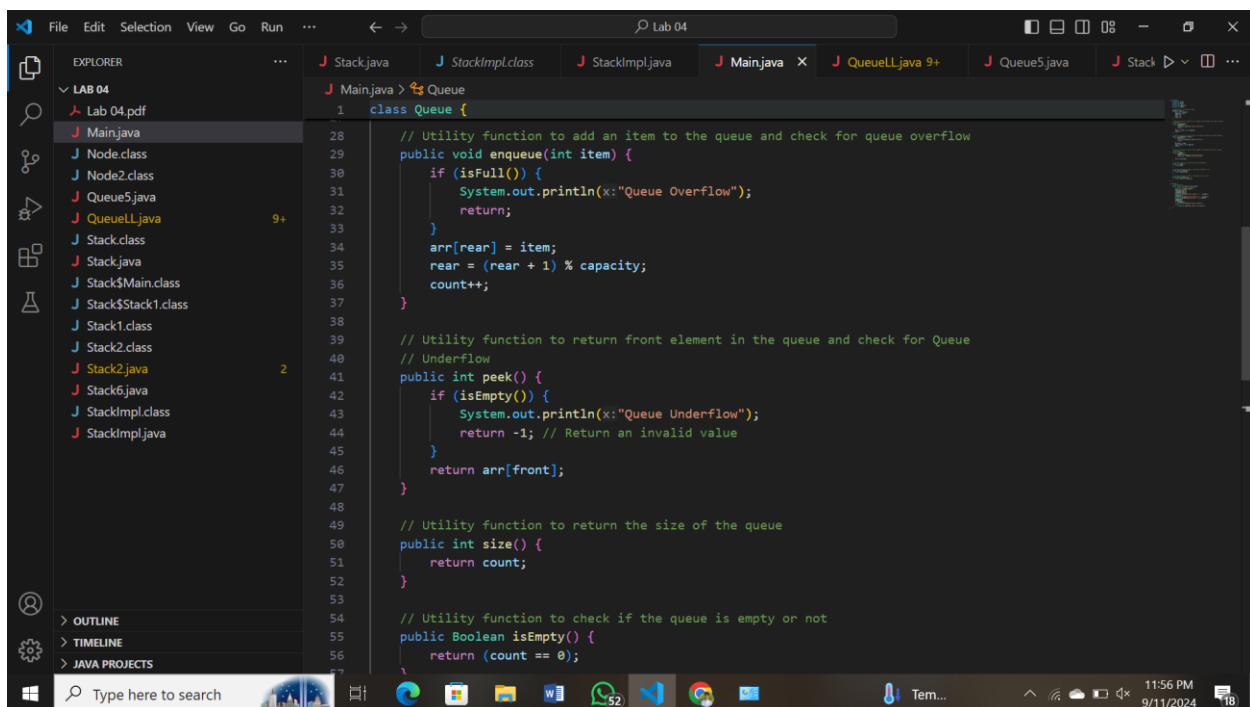
OUTPUT 2:



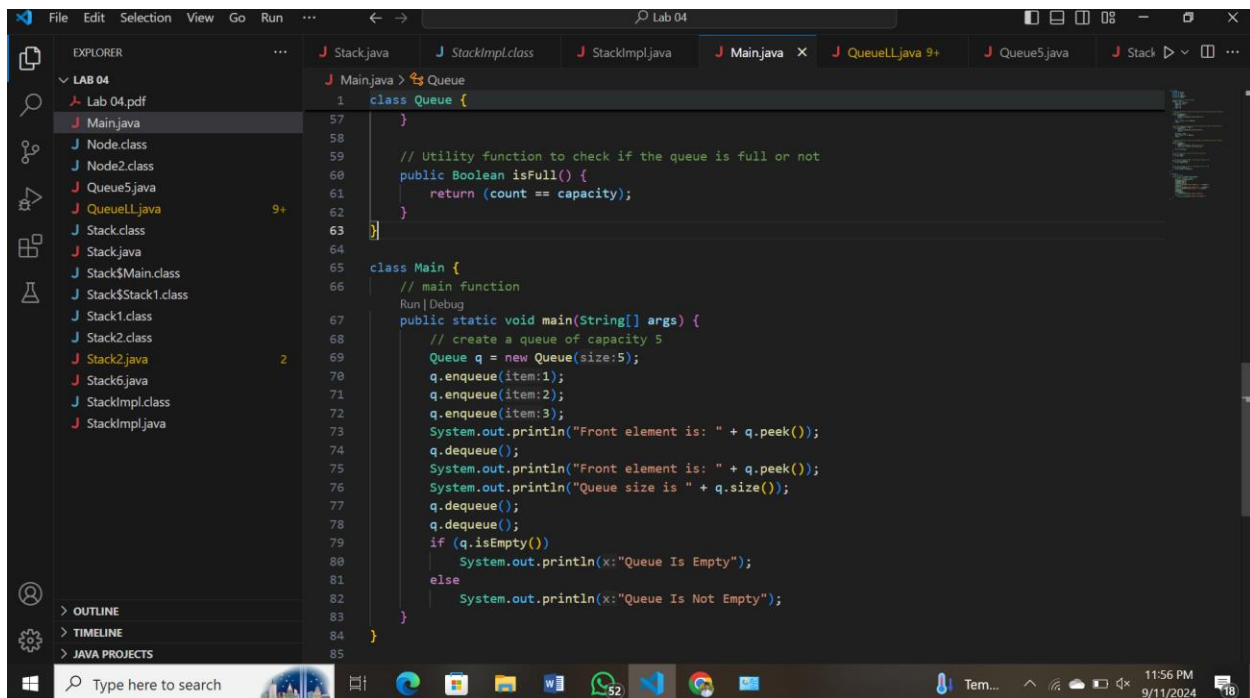
3: Queue using array: Understand provided code and implement all required methods in Queue. Queue Code is given below:



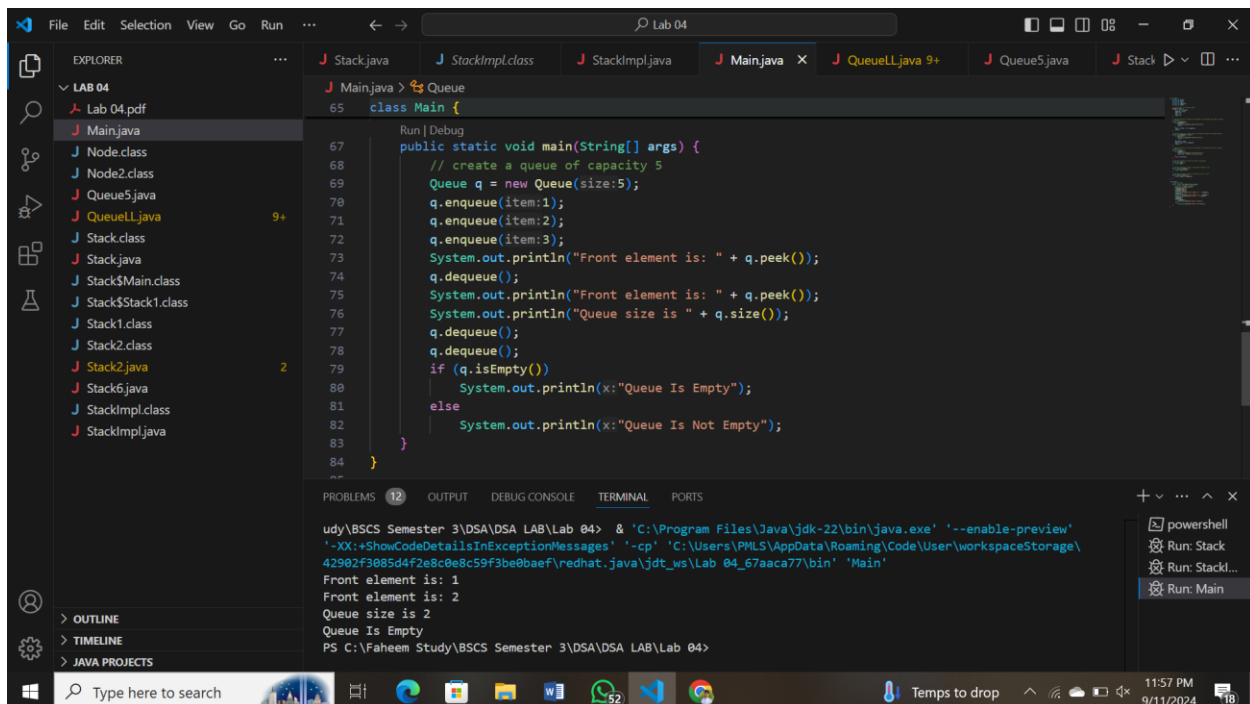
```
1 class Queue {
2     private int arr[];
3     private int front;
4     private int rear;
5     private int capacity;
6     private int count;
7
8     // Constructor to initialize queue
9     Queue(int size) {
10         arr = new int[size];
11         capacity = size;
12         front = 0;
13         rear = 0;
14         count = 0;
15     }
16
17     // Utility function to remove front element from the queue and check for Queue
18     // Underflow
19     public void dequeue() {
20         if (isEmpty()) {
21             System.out.println(x:"Queue Underflow");
22             return;
23         }
24         front = (front + 1) % capacity;
25         count--;
26     }
27
28     // Utility function to add an item to the queue and check for queue overflow
29     public void enqueue(int item) {
30         if (isFull()) {
31             System.out.println(x:"Queue Overflow");
32         }
33     }
34 }
```



```
28 // Utility function to add an item to the queue and check for queue overflow
29 public void enqueue(int item) {
30     if (isFull()) {
31         System.out.println(x:"Queue Overflow");
32         return;
33     }
34     arr[rear] = item;
35     rear = (rear + 1) % capacity;
36     count++;
37 }
38
39 // Utility function to return front element in the queue and check for Queue
40 // Underflow
41 public int peek() {
42     if (isEmpty()) {
43         System.out.println(x:"Queue Underflow");
44         return -1; // Return an invalid value
45     }
46     return arr[front];
47 }
48
49 // Utility function to return the size of the queue
50 public int size() {
51     return count;
52 }
53
54 // Utility function to check if the queue is empty or not
55 public Boolean isEmpty() {
56     return (count == 0);
57 }
```

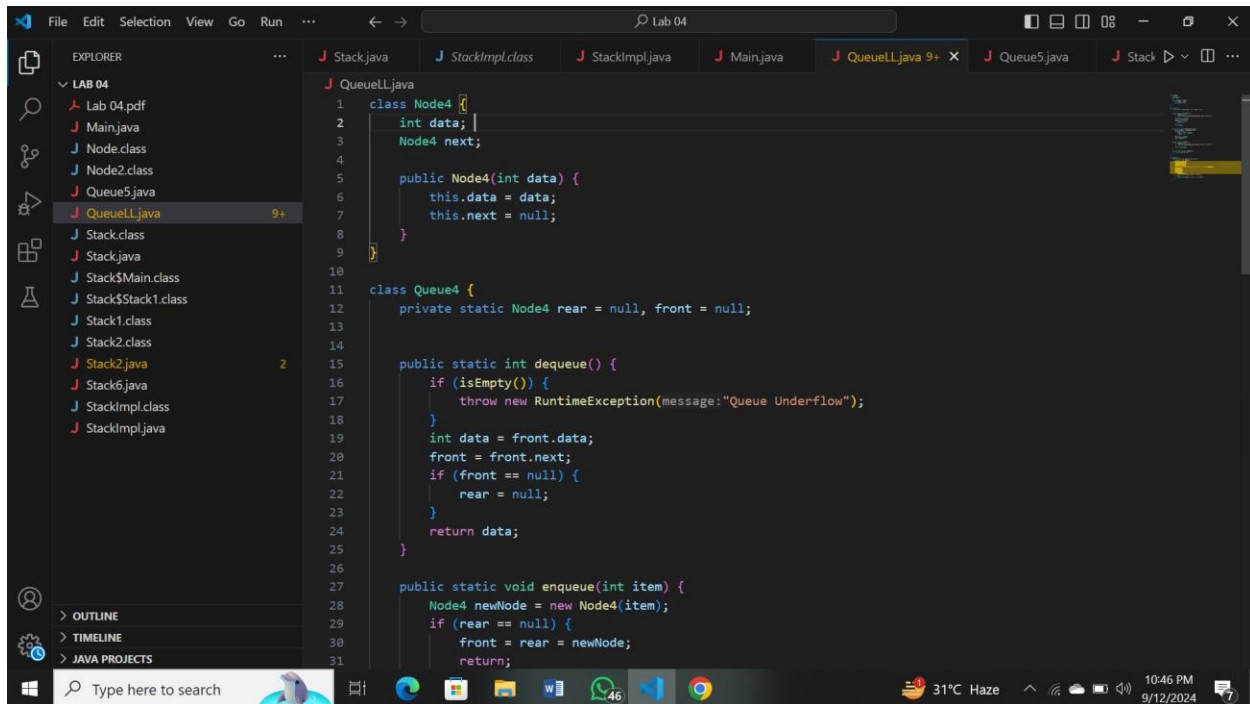


### OUTPUT 3:

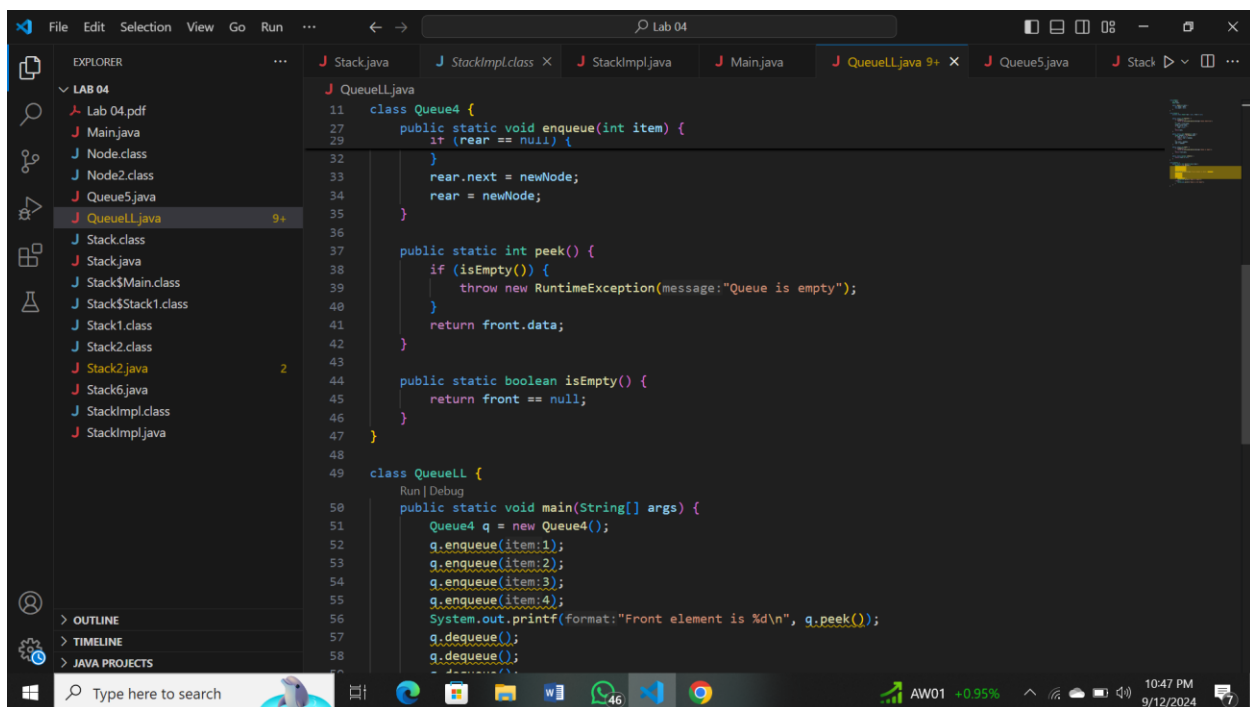




4. Queue using Linked list: Understand provided code and implement all required methods in Queue. Queue Code is given below:

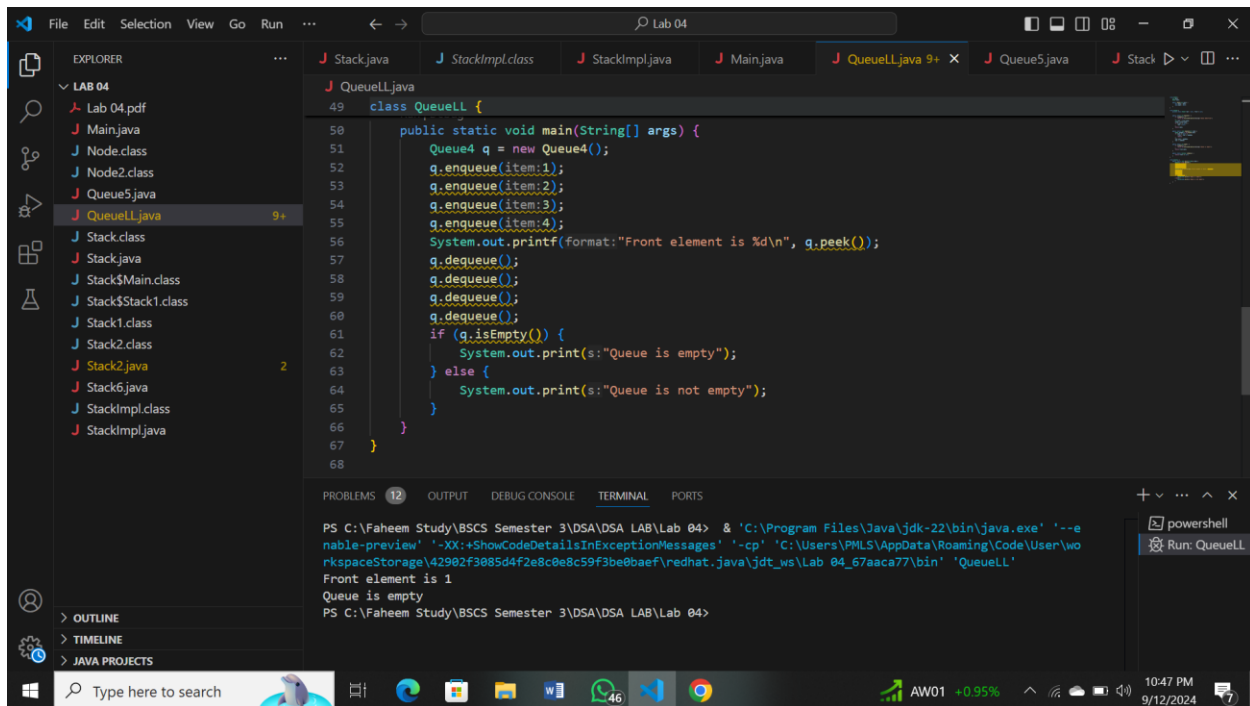


```
1 class Node4 {
2     int data;
3     Node4 next;
4
5     public Node4(int data) {
6         this.data = data;
7         this.next = null;
8     }
9 }
10
11 class Queue4 {
12     private static Node4 rear = null, front = null;
13
14     public static int dequeue() {
15         if (isEmpty()) {
16             throw new RuntimeException(message:"Queue Underflow");
17         }
18         int data = front.data;
19         front = front.next;
20         if (front == null) {
21             rear = null;
22         }
23         return data;
24     }
25
26     public static void enqueue(int item) {
27         Node4 newNode = new Node4(item);
28         if (rear == null) {
29             front = rear = newNode;
30         }
31     }
```



```
11 class Queue4 {
12     public static void enqueue(int item) {
13         if (rear == null) {
14             front = rear = new Node4(item);
15         } else {
16             rear.next = new Node4(item);
17             rear = rear.next;
18         }
19     }
20
21     public static int peek() {
22         if (isEmpty()) {
23             throw new RuntimeException(message:"Queue is empty");
24         }
25         return front.data;
26     }
27
28     public static boolean isEmpty() {
29         return front == null;
30     }
31 }
32
33 class QueueLL {
34     public static void main(String[] args) {
35         Queue4 q = new Queue4();
36         q.enqueue(item:1);
37         q.enqueue(item:2);
38         q.enqueue(item:3);
39         q.enqueue(item:4);
40         System.out.printf("Front element is %d\n", q.peek());
41         q.dequeue();
42         q.dequeue();
43     }
44 }
```

#### OUTPUT 4:

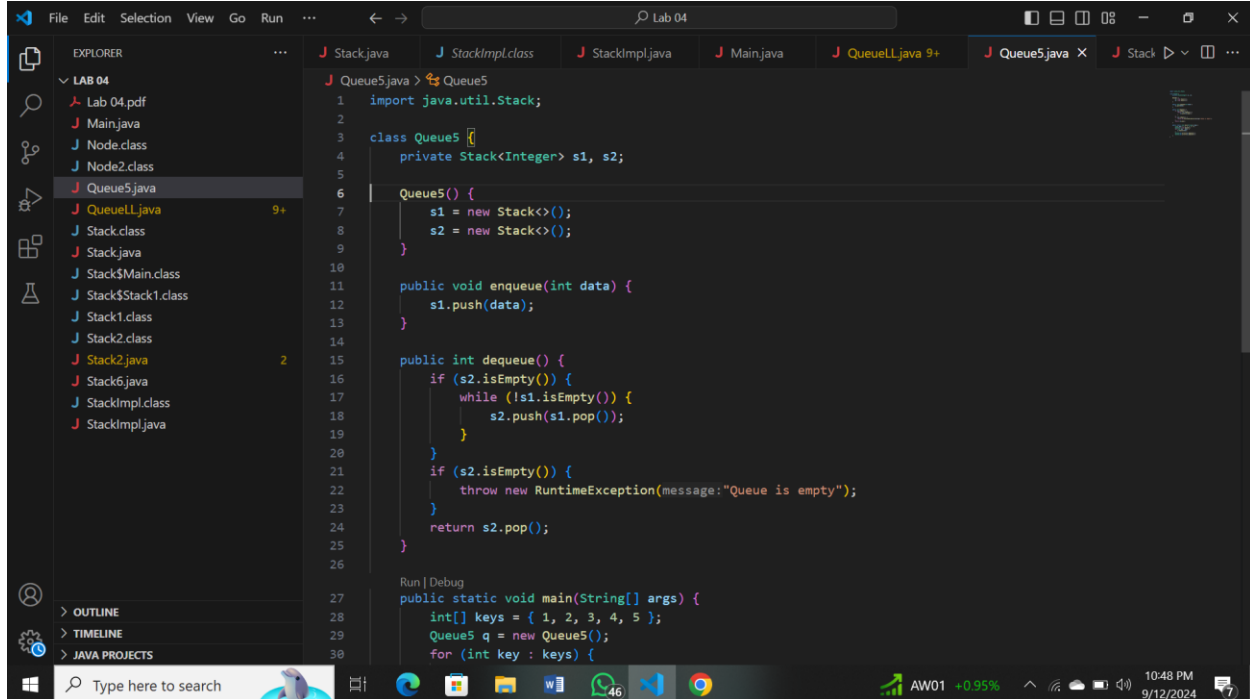


```
class QueueLL {
    public static void main(String[] args) {
        Queue4 q = new Queue4();
        q.enqueue(item:1);
        q.enqueue(item:2);
        q.enqueue(item:3);
        q.enqueue(item:4);
        System.out.printf(format:"Front element is %d\n", q.peak());
        q.dequeue();
        q.dequeue();
        q.dequeue();
        q.dequeue();
        if (q.isEmpty()) {
            System.out.print(s:"Queue is empty");
        } else {
            System.out.print(s:"Queue is not empty");
        }
    }
}
```

PS C:\Faheem Study\BSCS Semester 3\DSA\LAB\Lab 04> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-e nable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\PMLS\AppData\Roaming\Code\User\workspaceStorage\42902f3085d4f2e8c0e8c59f3be0baef\redhat.java\jdt\_ws\Lab 04\_67aaca77\bin' 'QueueLL'

Front element is 1  
Queue is empty  
PS C:\Faheem Study\BSCS Semester 3\DSA\LAB\Lab 04>

5. Queue using two Stacks: Understand provided code and implement all required methods in Queue Class. Sample Code is given below:



```
import java.util.Stack;

class Queue5 {
    private Stack<Integer> s1, s2;

    Queue5() {
        s1 = new Stack<>();
        s2 = new Stack<>();
    }

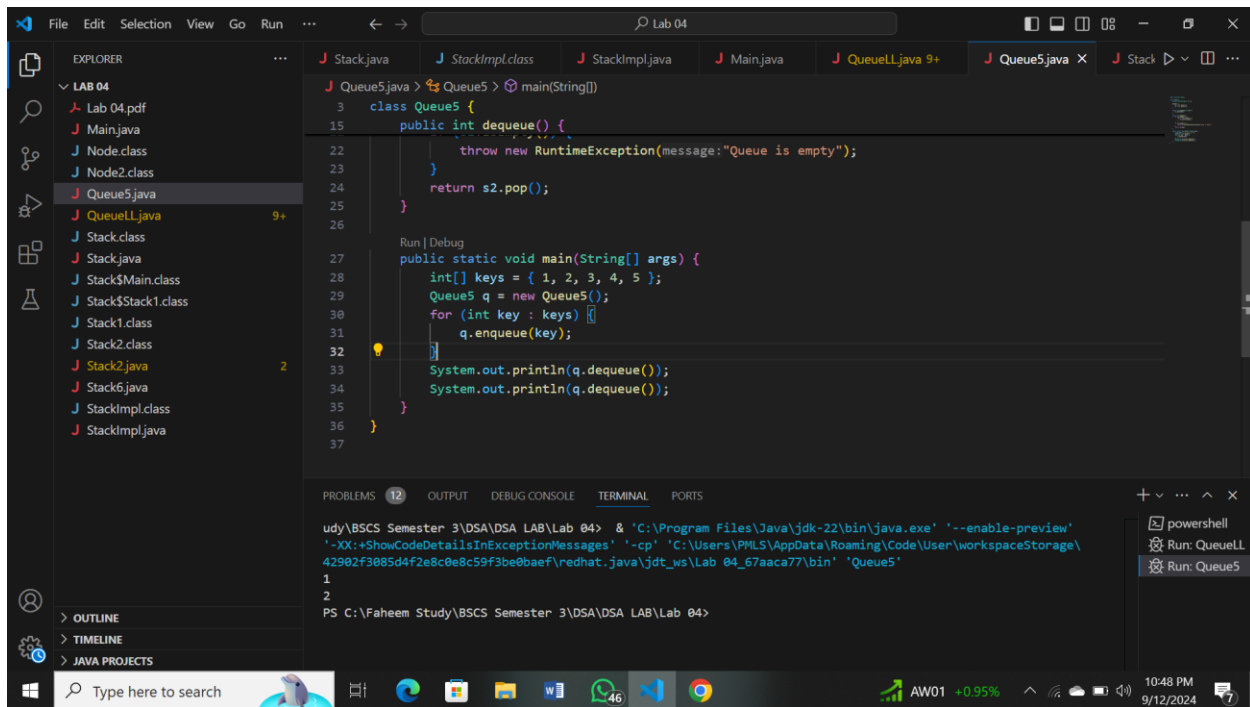
    public void enqueue(int data) {
        s1.push(data);
    }

    public int dequeue() {
        if (s2.isEmpty()) {
            while (!s1.isEmpty()) {
                s2.push(s1.pop());
            }
        }
        if (s2.isEmpty()) {
            throw new RuntimeException(message:"Queue is empty");
        }
        return s2.pop();
    }

    Run | Debug
    public static void main(String[] args) {
        int[] keys = { 1, 2, 3, 4, 5 };
        Queue5 q = new Queue5();
        for (int key : keys) {
```



OUTPUT:

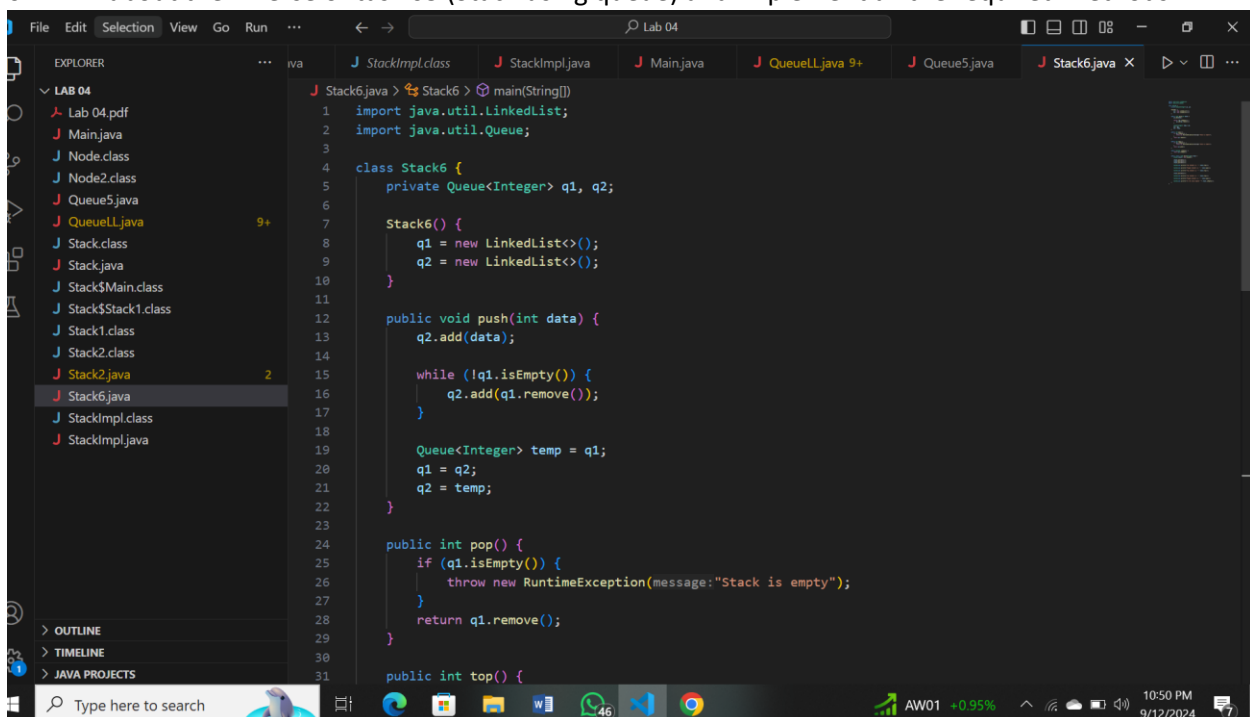


```
File Edit Selection View Go Run ... Lab 04
EXPLORER
LAB 04
  Lab 04.pdf
  Main.java
  Node.class
  Node2.class
  Queue5.java
  QueueLL.java 9+
  Stack.class
  Stack.java
  Stack$Main.class
  Stack$Stack1.class
  Stack1.class
  Stack2.class
  Stack2.java 2
  Stack6.java
  StackImpl.class
  StackImpl.java

J Queue5.java > Queue5 > main(String[])
3 class Queue5 {
15 public int dequeue() {
22     throw new RuntimeException(message:"Queue is empty");
23 }
24 return s2.pop();
25 }
26
Run | Debug
27 public static void main(String[] args) {
28     int[] keys = { 1, 2, 3, 4, 5 };
29     Queue5 q = new Queue5();
30     for (int key : keys) {
31         q.enqueue(key);
32     }
33     System.out.println(q.dequeue());
34     System.out.println(q.dequeue());
35 }
36 }
37

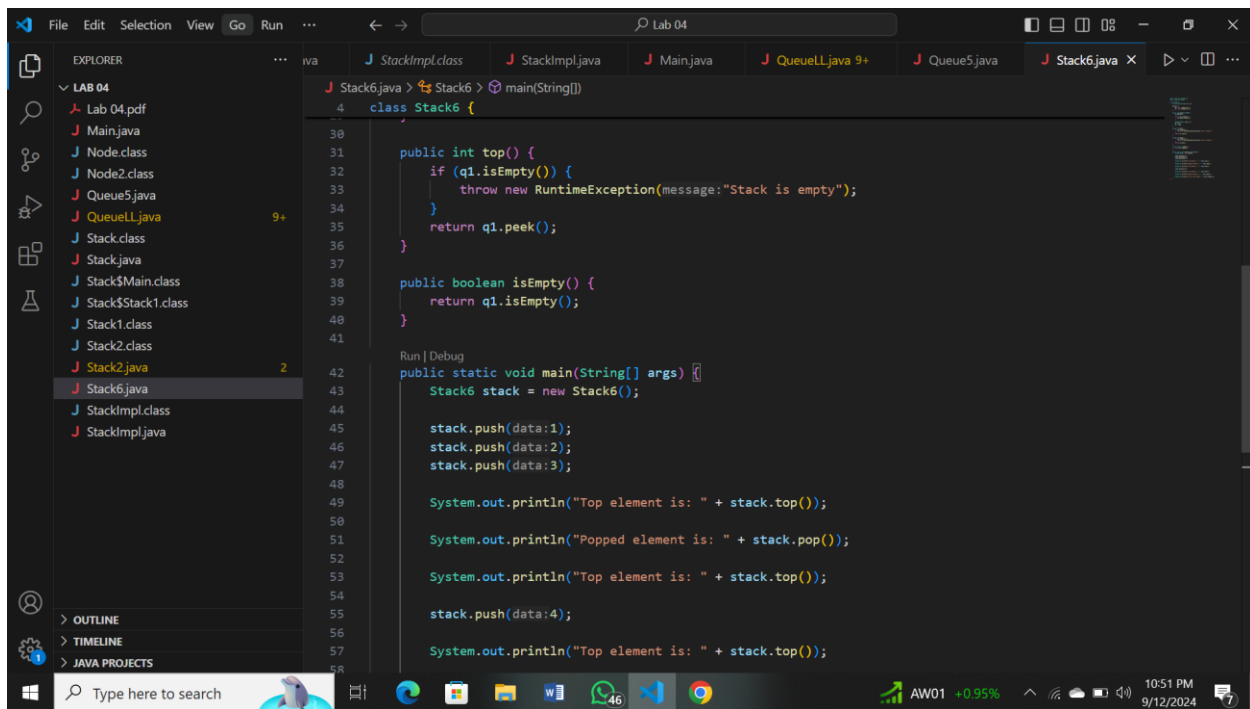
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS
ud\BSCS Semester 3\DSA LAB\Lab 04> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview'
'-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\PMLS\AppData\Roaming\Code\User\workspaceStorage\
42902f3085d4f2e8c0e8c59f3be0baef\redhat.java\jdt_ws\Lab_04_67aaca77\bin' 'Queue5'
1
2
PS C:\Faheem Study\BSCS Semester 3\DSA LAB\Lab 04>
```

6. Think about the inverse of task 05 (Stack using queue) and implement all the required methods.



```
File Edit Selection View Go Run ... Lab 04
EXPLORER
LAB 04
  Lab 04.pdf
  Main.java
  Node.class
  Node2.class
  Queue5.java
  QueueLL.java 9+
  Stack.class
  Stack.java
  Stack$Main.class
  Stack$Stack1.class
  Stack1.class
  Stack2.class
  Stack2.java 2
  Stack6.java
  StackImpl.class
  StackImpl.java

J Stack6.java > Stack6 > main(String[])
1 import java.util.LinkedList;
2 import java.util.Queue;
3
4 class Stack6 {
5     private Queue<Integer> q1, q2;
6
7     Stack6() {
8         q1 = new LinkedList<>();
9         q2 = new LinkedList<>();
10    }
11
12    public void push(int data) {
13        q2.add(data);
14
15        while (!q1.isEmpty()) {
16            q2.add(q1.remove());
17        }
18
19        Queue<Integer> temp = q1;
20        q1 = q2;
21        q2 = temp;
22    }
23
24    public int pop() {
25        if (q1.isEmpty()) {
26            throw new RuntimeException(message:"Stack is empty");
27        }
28        return q1.remove();
29    }
30
31    public int top() {
```



OUTPUT 6:

