



9/14/2024

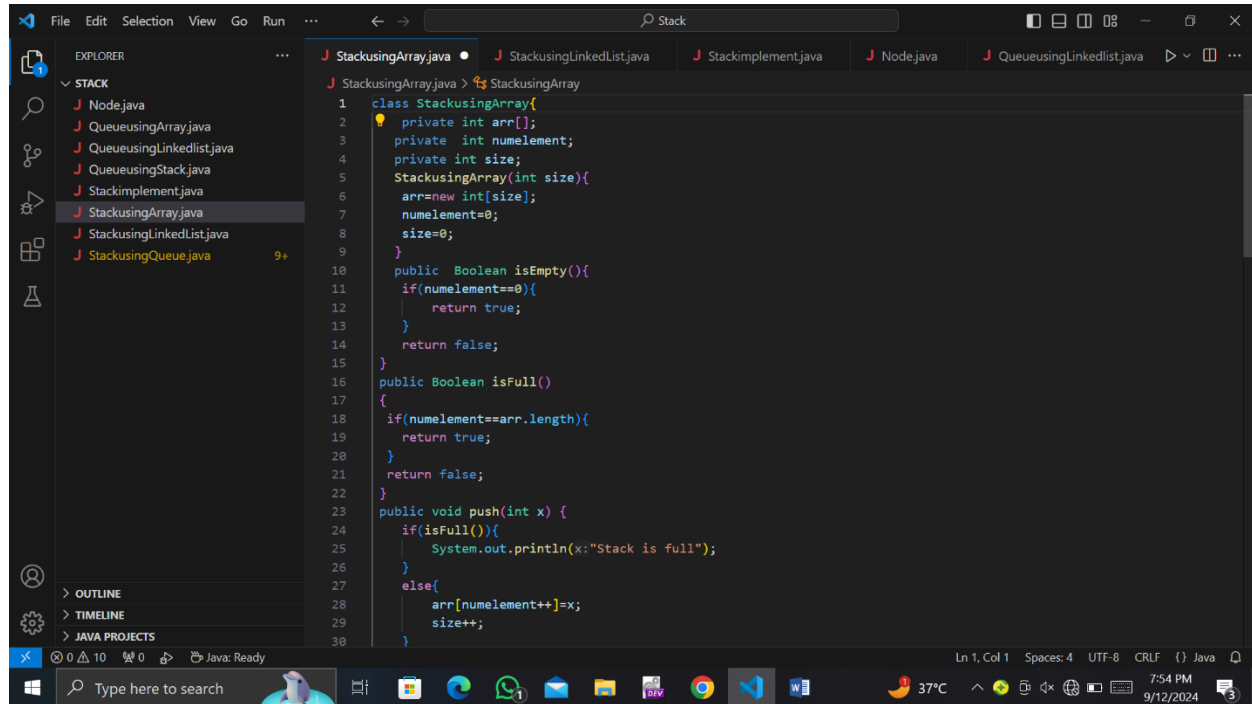
Data Structures

Lab 4 exercise solutions

Uzair Ali Memon S/O Yar Muhammad Memon
B-S CS III Section A
023-23-0350
To: Ma'am Marina Rajpoot

Task #1

Stack using Array



The screenshot shows a code editor with the following Java code for `StackusingArray.java`:

```
1 class StackusingArray{
2     private int arr[];
3     private int numelement;
4     private int size;
5     StackusingArray(int size){
6         arr=new int[size];
7         numelement=0;
8         size=0;
9     }
10    public Boolean isEmpty(){
11        if(numelement==0){
12            return true;
13        }
14        return false;
15    }
16    public Boolean isFull()
17    {
18        if(numelement==arr.length){
19            return true;
20        }
21        return false;
22    }
23    public void push(int x) {
24        if(isFull()){
25            System.out.println(x:"Stack is full");
26        }
27        else{
28            arr[numelement++]=x;
29            size++;
30        }
31    }
```

The IDE interface includes a sidebar with a file explorer showing a project named "STACK" containing files like `Node.java`, `QueueusingArray.java`, `QueueusingLinkedList.java`, `QueueusingStack.java`, `Stackimplement.java`, `StackusingArray.java`, `StackusingLinkedList.java`, and `StackusingQueue.java`. The bottom status bar indicates the current cursor position is at line 1, column 1, with 4 spaces, UTF-8 encoding, and CRLF line endings.

```
StackusingArray.java
31     size++;
32 }
33 }
34 public int pop()
35 {
36     if(isEmpty()){
37         System.out.println(x:"Stack is empty");
38         return -1;
39     }
40     else{
41         size--;
42         return arr[--numelement];
43     }
44 }
45 public int top()
46 {
47     if(isEmpty()){
48         System.out.println(x:"Stack is empty");
49         return -1;
50     }
51     else{
52         return arr[numelement-1];
53     }
54 }
55 public int size(){
56     return size;
57 }
58 Run | Debug
59 public static void main (String[] args)
60 {
```

```
StackusingArray.java
60 StackusingArray stack = new StackusingArray(size:3);
61 stack.push(x:1); // Inserting 1 in the stack
62 stack.push(x:2);
63 System.out.println("Top element is: " + stack.top());
64 System.out.println("Removing : "+stack.pop()); // removing the top 2
65 System.out.println("Removing : "+stack.pop()); // removing the top 1
66 stack.push(x:3); // Inserting 3 in the stack
67 System.out.println("Top element is: " + stack.top()); // Inserting 2 in the stack
68 System.out.println("Stack size is " + stack.size());
69 System.out.println("Removing : "+stack.pop()); // removing the top 3
70 System.out.println(stack.pop());
71 }
72 }
73 }
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Run: StackusingArray

```
PS E:\bscs3\DSA lab\lab4\Stack> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\BOSG\AppData\Roaming\Code\User\workspaceStorage\3d8c8ae7e1e987ce78fb91bfa6c1286b\redhat.java\jdt_ws\Stack_6b238647\bin'
'StackusingArray'
Top element is: 2
Removing : 2
Removing : 1
Top element is: 3
Stack size is 1
Removing : 3
Stack is empty
-1
PS E:\bscs3\DSA lab\lab4\Stack>
```

Task #2

Stack using Linked List

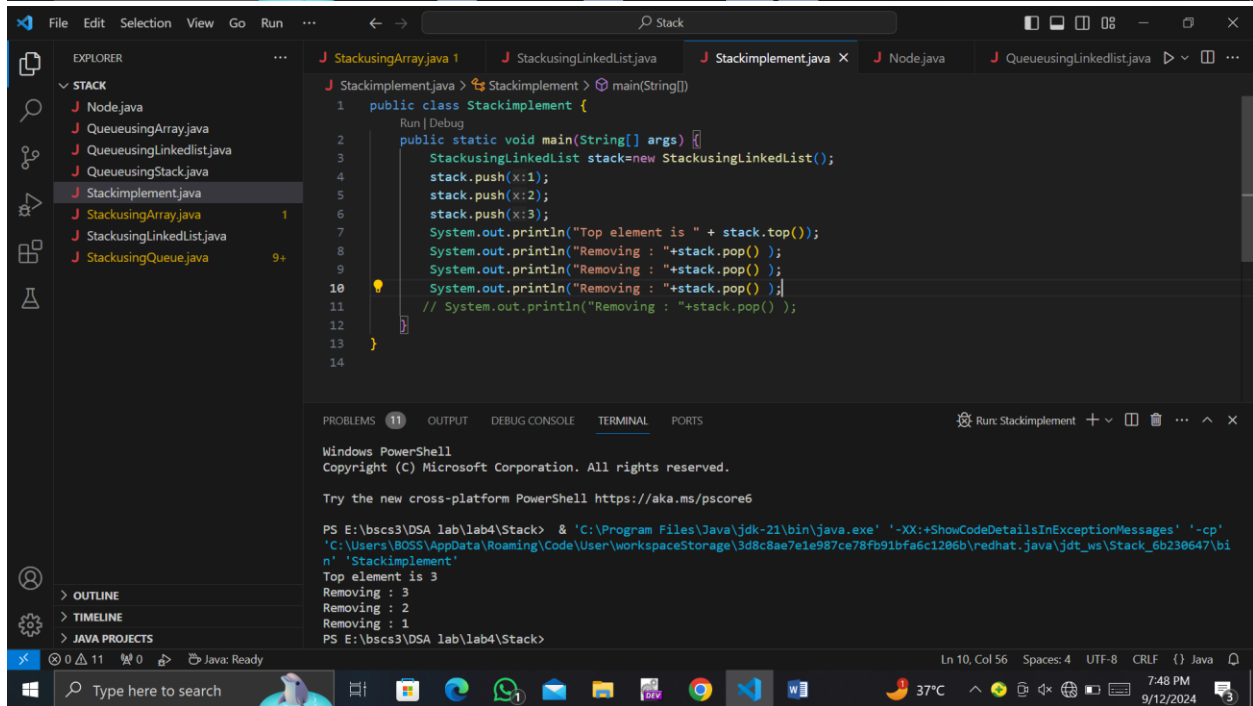
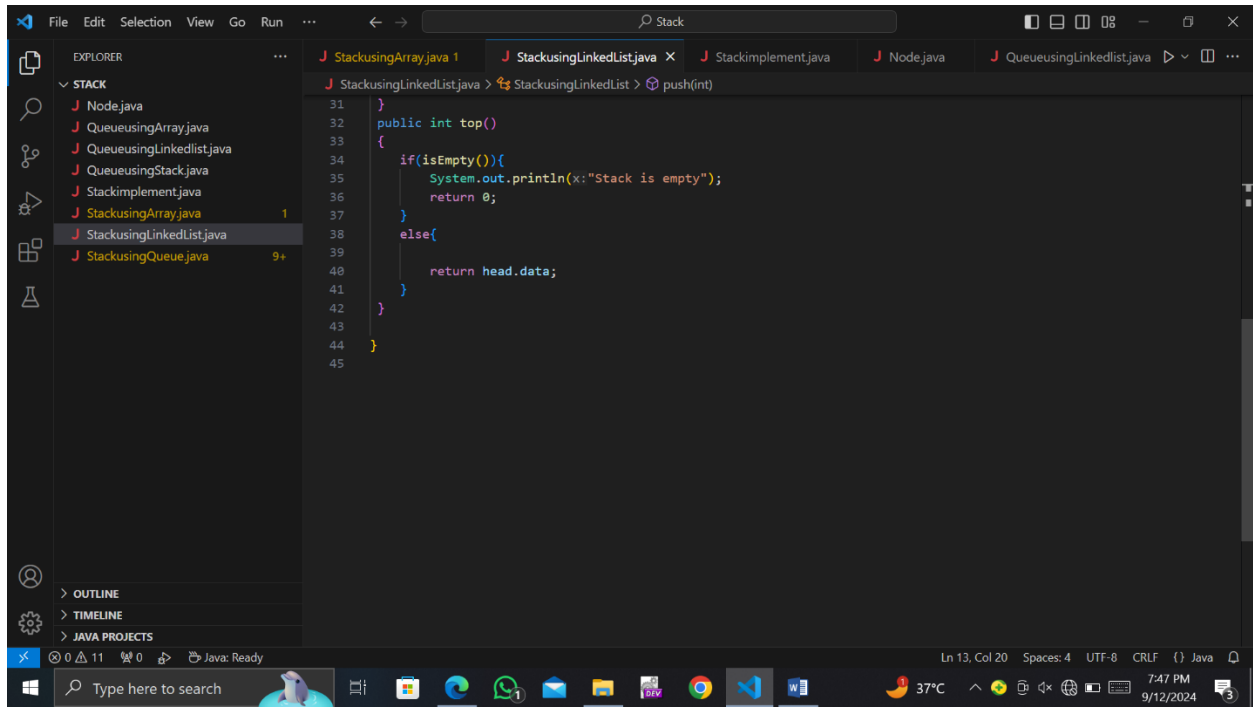
The image displays two screenshots of a Java IDE, likely IntelliJ IDEA, showing the implementation of a Stack using a Linked List.

Top Screenshot: The Explorer panel on the left shows a project named "STACK" with several files. The "Node.java" file is selected, and its code is displayed in the editor. The code defines a `Node` class with an `int data` field, a `Node next` field, and a constructor `Node(int data)` that initializes `this.data = data` and `this.next = null`.

```
1 public class Node {
2     int data;
3     Node next;
4     public Node(int data){
5         this.data=data;
6         this.next=null;
7     }
8 }
9
```

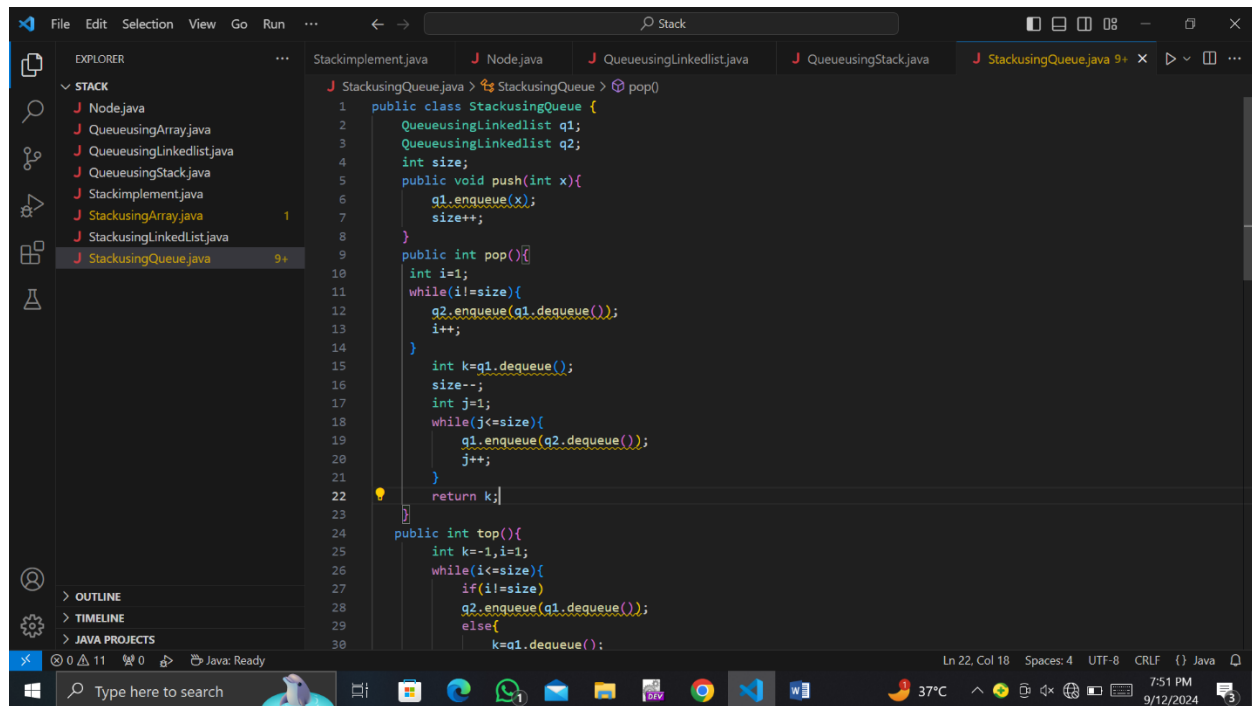
Bottom Screenshot: The Explorer panel shows the "StackusingLinkedList.java" file selected. The code in the editor implements a `StackusingLinkedList` class. It includes a `private Node head` field, an `isEmpty()` method, a `push(int x)` method (commented as "insert at the beginning"), and a `pop()` method. The `push` method creates a new `Node` and sets it as the `head` if the stack is empty, or inserts it before the current `head` otherwise. The `pop` method checks if the stack is empty and prints a message if so, then returns the `data` of the current `head` and updates `head` to `head.next`.

```
1 public class StackusingLinkedList {
2     private Node head;
3     public boolean isEmpty()
4     {
5         if(head==null){
6             return true;
7         }
8         return false;
9     }
10    public void push(int x) // insert at the beginning
11    {
12        Node newNode=new Node(x);
13        if(head==null){
14            head=newNode;
15        }
16        else{
17            newNode.next=head;
18            head=newNode;
19        }
20    }
21    public int pop(){
22        if(isEmpty()){
23            System.out.println(x:"Stack is empty");
24            return 0;
25        }
26        else{
27            int k=head.data;
28            head=head.next;
29            return k;
30        }
31    }
32 }
```



Task #3

Stack using Queue



```
1 public class StackusingQueue {
2     QueueusingLinkedList q1;
3     QueueusingLinkedList q2;
4     int size;
5     public void push(int x){
6         q1.enqueue(x);
7         size++;
8     }
9     public int pop(){
10        int i=1;
11        while(i!=size){
12            q2.enqueue(q1.dequeue());
13            i++;
14        }
15        int k=q1.dequeue();
16        size--;
17        int j=1;
18        while(j<=size){
19            q1.enqueue(q2.dequeue());
20            j++;
21        }
22        return k;
23    }
24    public int top(){
25        int k=-1,i=1;
26        while(i<=size){
27            if(i!=size)
28                q2.enqueue(q1.dequeue());
29            else{
30                k=q1.dequeue();
```

```
StackusingQueue.java
StackusingQueue > pop()
31     q2.enqueue(k);
32     }
33     i++;
34     }
35     QueueusingLinkedList temp=q1;
36     q1=q2;
37     q2=temp;
38     if(k==1){
39         System.out.println(x:"Stack is empty");
40     }
41     return k;
42 }

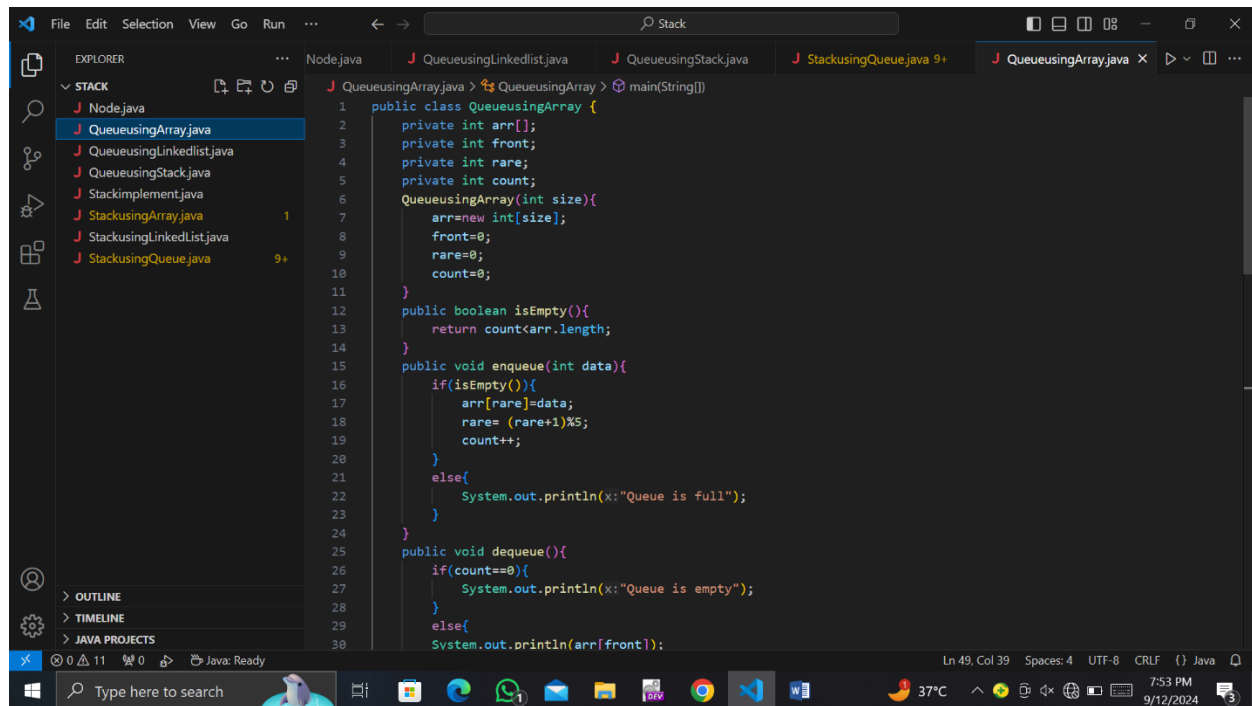
Run | Debug
public static void main(String[] args) {
43     StackusingQueue stack=new StackusingQueue();
44     stack.push(x:1);
45     stack.push(x:2);
46     stack.push(x:3);
47     System.out.println("Top element is " + stack.top());
48     System.out.println("Removing : "+stack.pop() );
49     stack.push(x:4);
50     System.out.println("Top element is " + stack.top());
51     System.out.println("Removing : "+stack.pop() );
52     System.out.println("Top element is " + stack.top());
53     System.out.println("Removing : "+stack.pop() );
54     System.out.println("Top element is " + stack.top());
55     System.out.println("Removing : "+stack.pop() );
56     System.out.println("Top element is " + stack.top());
57     System.out.println("Removing : "+stack.pop() );
58     System.out.println("Top element is " + stack.top());
59 }
```

```
StackusingQueue.java
StackusingQueue > pop()
56     System.out.println("Removing : "+stack.pop() );
57     System.out.println("Top element is " + stack.top());
58     }
59 }
60 }

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: StackusingQueue
'C:\Users\B055\AppData\Roaming\Code\User\workspaceStorage\3d8c8ae7e1e987ce78fb91bfa6c1206b\redhat.java\jdt_ws\Stack_6b230647\bin' 'StackusingQueue'
Top element is 3
Removing : 3
Top element is 4
Removing : 4
Top element is 2
Removing : 2
Top element is 1
Removing : 1
Stack is empty
Top element is -1
PS E:\bscs3\DSA lab\lab4\Stack>
```

Task #4

Queue using Array



The screenshot shows a code editor with the following Java code for a Queue using Array:

```
1 public class QueueusingArray {
2     private int arr[];
3     private int front;
4     private int rare;
5     private int count;
6     QueueusingArray(int size){
7         arr=new int[size];
8         front=0;
9         rare=0;
10        count=0;
11    }
12    public boolean isEmpty(){
13        return count<arr.length;
14    }
15    public void enqueue(int data){
16        if(isEmpty()){
17            arr[rare]=data;
18            rare= (rare+1)%5;
19            count++;
20        }
21        else{
22            System.out.println(x:"Queue is full");
23        }
24    }
25    public void dequeue(){
26        if(count==0){
27            System.out.println(x:"Queue is empty");
28        }
29        else{
30            System.out.println(arr[front]);
```

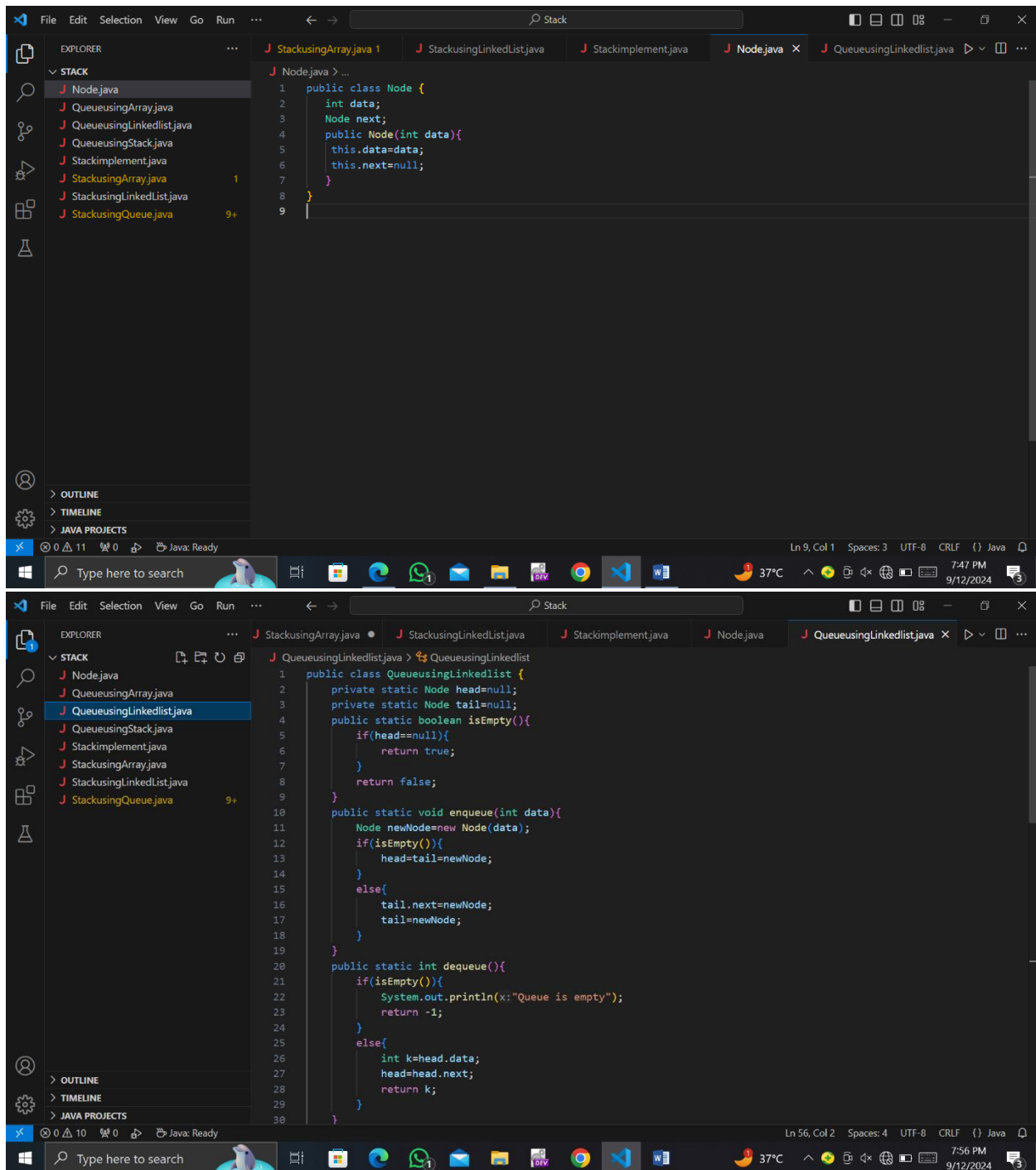
The IDE interface includes a sidebar with a file explorer showing a project named 'STACK' with files like 'Node.java', 'QueueusingArray.java', 'QueueusingLinkedList.java', 'QueueusingStack.java', 'Stackimplement.java', 'StackusingArray.java', 'StackusingLinkedList.java', and 'StackusingQueue.java'. The main editor area shows the code for 'QueueusingArray.java'. The status bar at the bottom indicates 'Ln 49, Col 39', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Java'.


```
File Edit Selection View Go Run ... Stack
EXPLORER
STACK
Node.java
QueueusingArray.java
QueueusingLinkedList.java
QueueusingStack.java
Stackimplement.java
StackusingArray.java 1
StackusingLinkedList.java
StackusingQueue.java 9+
OUTLINE
TIMELINE
JAVA PROJECTS
QueueusingArray.java > QueueusingArray > main(String[])
31 front=(front+1)%5;
32 count--;
33 }
34 }
35 public int peek()
36 {
37     return arr[front];
38 }
39 public int size()
40 {
41     return count;
42 }
Run | Debug
43 public static void main(String[] args) {
44     QueueusingArray q=new QueueusingArray(size:7);
45     q.enqueue(data:1);
46     q.enqueue(data:2);
47     q.enqueue(data:3);
48     System.out.println("Queue size is " + q.size());
49     System.out.println(" Front and top element is : "+q.peek());
50     System.out.print(s:"Removing : ");
51     q.dequeue();
52     System.out.println(" Front and top element is : "+q.peek());
53     System.out.print(s:"Removing : ");
54     q.dequeue();
55     System.out.println("Queue size is " + q.size());
56     System.out.println(" Front and top element is : "+q.peek());
57     System.out.print(s:"Removing : ");
58     q.dequeue();
59     System.out.println(" Front and top element is : "+q.peek());
Ln 49, Col 39 Spaces: 4 UTF-8 CRLF {} Java
```

```
File Edit Selection View Go Run ... Stack
EXPLORER
STACK
Node.java
QueueusingArray.java
QueueusingLinkedList.java
QueueusingStack.java
Stackimplement.java
StackusingArray.java 1
StackusingLinkedList.java
StackusingQueue.java 9+
OUTLINE
TIMELINE
JAVA PROJECTS
QueueusingArray.java > QueueusingArray > main(String[])
58 q.dequeue();
59 System.out.println(" Front and top element is : "+q.peek());
60 System.out.println("Queue size is " + q.size());
61 }
62 }
63 }
PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: QueueusingArray
'C:\Users\B055\AppData\Roaming\Code\User\workspaceStorage\3d8c8ae7e1e987ce78fb91bfa6c1206b\redhat.java\jdt_ws\Stack_6b230647\bin'
'QueueusingArray'
Queue size is 3
Front and top element is : 1
Removing : 1
Front and top element is : 2
Removing : 2
Queue size is 1
Front and top element is : 3
Removing : 3
Front and top element is : 0
Queue size is 0
PS E:\bscs3\DSA lab\lab4\Stack>
Ln 49, Col 39 Spaces: 4 UTF-8 CRLF {} Java
```

Task #5

Queue using Linked List



```
30 }
31
32 public static int peek(){
33     if(isEmpty()){
34         System.out.println("Queue is empty");
35         return -1;
36     }
37     else{
38         return head.data;
39     }
40 }
41
42 Run | Debug
43 public static void main(String args[]){
44     enqueue(data:1);
45     enqueue(data:2);
46     enqueue(data:3);
47     System.out.println("Front and top element is : "+peek());
48     System.out.println("Removing : "+dequeue());
49     System.out.println("Front and top element is : "+peek());
50     System.out.println("Removing : "+dequeue());
51     System.out.println("Front and top element is : "+peek());
52     System.out.println("Removing : "+dequeue());
53     enqueue(data:4);
54     System.out.println("Front and top element is : "+peek());
55     System.out.println("Removing : "+dequeue());
56     System.out.println("Front and top element is : "+peek());
57 }
```

```
52     System.out.println("Front and top element is : "+peek());
53     System.out.println("Removing : "+dequeue());
54     System.out.println("Front and top element is : "+peek());
55 }
56
57

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: QueueusingLinkedList
Front and top element is : 1
Removing : 1
Front and top element is : 2
Removing : 2
Front and top element is : 3
Removing : 3
Queue is empty
Removing : -1
Front and top element is : 4
Removing : 4
Queue is empty
Front and top element is : -1
PS E:\bscs3\DSA lab\lab4\Stack>
```

Task #6

Queue using Stack

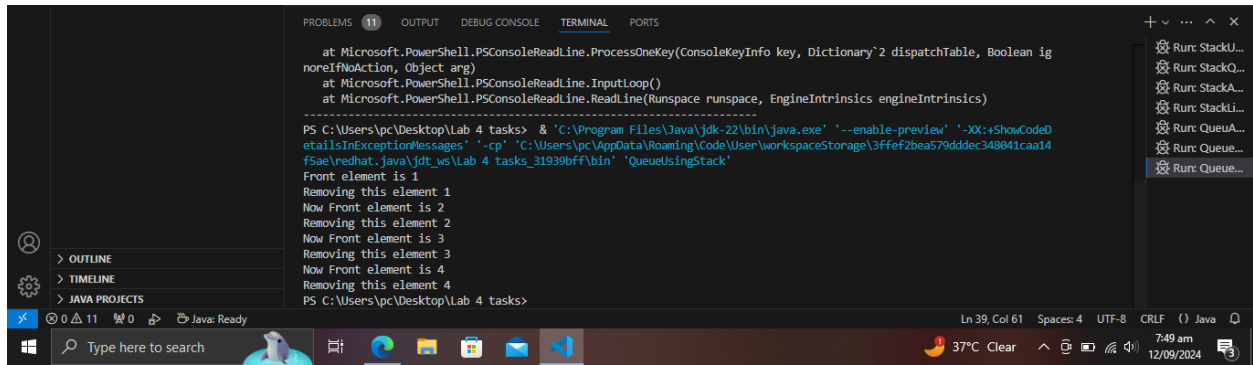
The image displays two screenshots of a Visual Studio Code editor window, showing the implementation of a Queue using two Stacks. The Explorer pane on the left lists the project files, including `StackUsingQueue.java`, which is the active file.

Top Screenshot: Shows the `enqueue` and `dequeue` methods of the `QueueUsingStack` class.

```
1 public class QueueUsingStack {
2     StackUsingLinkedList s1;
3     StackUsingLinkedList s2;
4     public void enqueue(int x){
5         s1.push(x);
6     }
7     public int dequeue(){
8         while(s1.isEmpty()){
9             s2.push(s1.pop());
10        }
11        int k=s2.pop();
12        while(s2.isEmpty()){
13            s1.push(s2.pop());
14        }
15        return k;
16    }
17    public int peek(){
18        return s2.top();
19    }
20    Run | Debug
21    public static void main(String args[]){
22        QueueUsingStack q=new QueueUsingStack();
23        q.enqueue(x:1);
24        q.enqueue(x:2);
25        q.enqueue(x:3);
26        System.out.println("Front and top element is : "+q.peek());
27        System.out.println("Removing : "+q.dequeue());
28        System.out.println("Front and top element is : "+q.peek());
29        System.out.println("Removing : "+q.dequeue());
30        System.out.println("Front and top element is : "+q.peek());
31    }
32 }
```

Bottom Screenshot: Shows the continuation of the `main` method, demonstrating the queue operations.

```
30        System.out.println("Removing : "+q.dequeue());
31        q.enqueue(x:4);
32        System.out.println("Front and top element is : "+q.peek());
33        System.out.println("Removing : "+q.dequeue());
34        System.out.println("Front and top element is : "+q.peek());
35    }
36 }
37
38 }
```



THE END