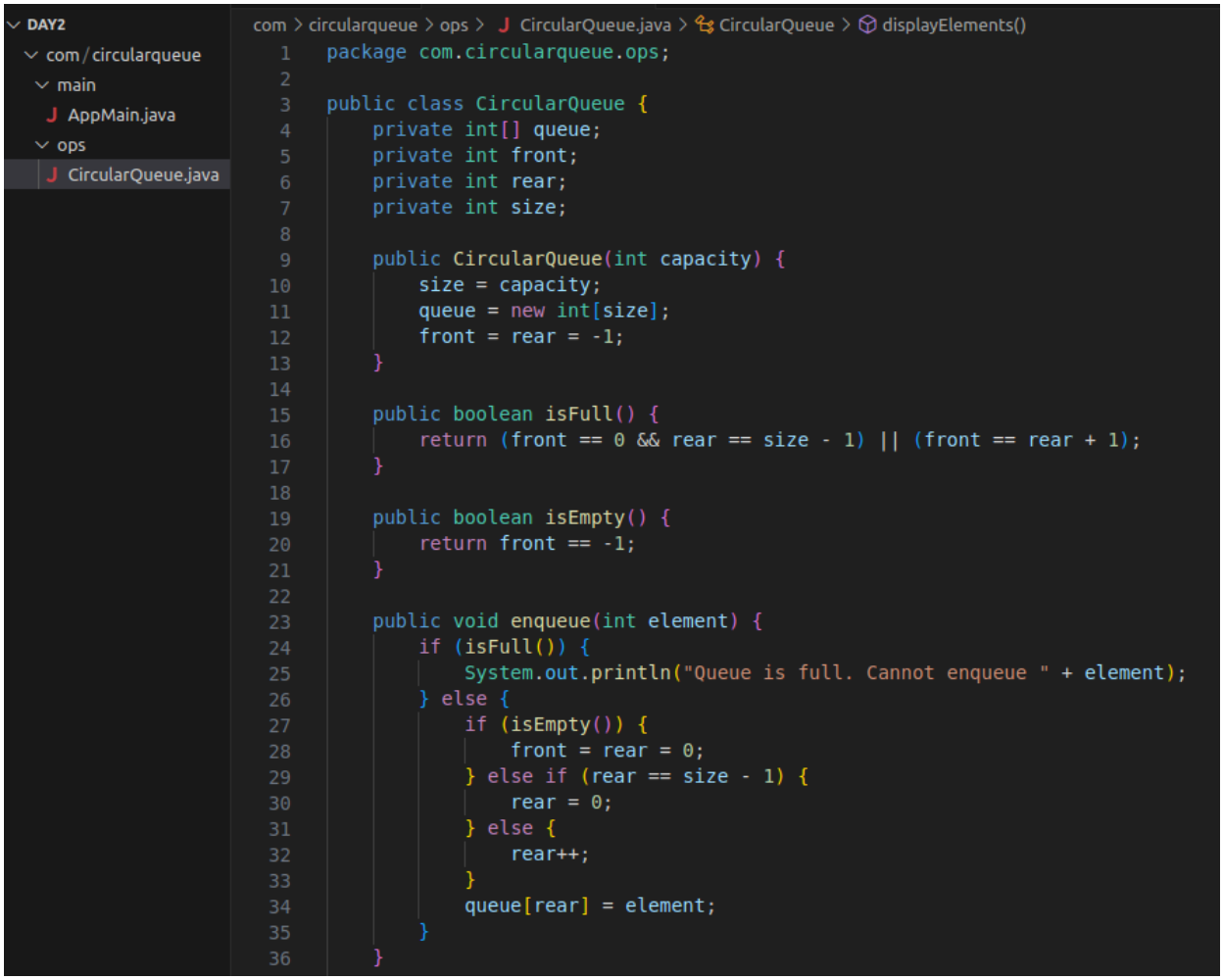
LAB ASSIGNMENT NO – 2

Qn 1. Write a Java program to implement a circular queue of size 5. Following functionalities must be implemented in the program.

A. A function to check if queue is full
B. A function to check if queue is empty
C. A function to insert an element to queue if queue is not full. Add the elements 10, 25, 38, 45, 60 to circular queue and display the elements.
D. Add 70 to queue
E. Remove 2 elements from queue and display the elements.
F. Add 90 to queue and display the elements.

```java
package com.circularqueue.main;
import com.circularqueue.ops.CircularQueue;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        CircularQueue queue = new CircularQueue(capacity:5);

        // Add elements to the queue
        queue.enqueue(element:10);
        queue.enqueue(element:25);
        queue.enqueue(element:38);
        queue.enqueue(element:45);
        queue.enqueue(element:60);

        // Display current elements
        queue.displayElements();

        // Add 70 to the queue
        queue.enqueue(element:70);

        // Remove 2 elements and display the elements
        queue.dequeue();
        queue.dequeue();
        queue.displayElements();

        // Add 90 to the queue and display the elements
        queue.enqueue(element:90);
        queue.displayElements();
    }
}
```


DAY2
com / circularqueue
main
AppMain.java
ops
CircularQueue.java
com > circularqueue > ops > CircularQueue.java > CircularQueue > displayElements()
1 package com.circularqueue.ops;
2
3 public class CircularQueue {
4 private int[] queue;
5 private int front;
6 private int rear;
7 private int size;
8
9 public CircularQueue(int capacity) {
10 size = capacity;
11 queue = new int[size];
12 front = rear = -1;
13 }
14
15 public boolean isFull() {
16 return (front == 0 && rear == size - 1) || (front == rear + 1);
17 }
18
19 public boolean isEmpty() {
20 return front == -1;
21 }
22
23 public void enqueue(int element) {
24 if (isFull()) {
25 System.out.println("Queue is full. Cannot enqueue " + element);
26 } else {
27 if (isEmpty()) {
28 front = rear = 0;
29 } else if (rear == size - 1) {
30 rear = 0;
31 } else {
32 rear++;
33 }
34 queue[rear] = element;
35 }
36 }

```java
    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty. Cannot dequeue.");
            return -1; // You can choose a different error value if needed.
        } else {
            int removedElement = queue[front];
            if (front == rear) {
                front = rear = -1;
            } else if (front == size - 1) {
                front = 0;
            } else {
                front++;
            }
            return removedElement;
        }
    }

    public void displayElements() {
        if (isEmpty()) {
            System.out.println("Queue is empty.");
        } else {
            System.out.print("Current elements in the queue: ");
            int i = front;
            do {
                System.out.print(queue[i] + " ");
                if (i == rear && rear != size - 1) {
                    break;
                }
                if (i == size - 1) {
                    i = 0;
                } else {
                    i++;
                }
            } while (i != (rear + 1) % size);
            System.out.println();
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/27thoct/Day2$  cd /med
ia/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/27thoct/Day2 ; /usr/bin/env /usr/lib/jvm/jav
a-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/769dec34
2b4c1fdc1f444257db24522f/redhat.java/jdt_ws/Day2_6b1cf3/bin com.circularqueue.main.AppMain

Current elements in the queue: 10 25 38 45 60
Queue is full. Cannot enqueue 70
Current elements in the queue: 38 45 60
Current elements in the queue: 38 45 60 90
```

Qn 2. Create a Linked List to store mark of students in ascending order.

A. Insert student with mark = 70
B. Insert student with mark = 60
C. Insert student with mark = 80
D. Insert student with mark = 90
E. Insert student with mark = 75
F. Display all marks
G. Delete student with mark = 60
H. Delete student with mark = 90
I. Delete student with mark = 90
J. Display all marks

```java
package com.main;
import com.student.StudentMarksList;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        StudentMarksList studentList = new StudentMarksList();

        // Insert students with marks
        studentList.insert(mark:70);
        studentList.insert(mark:60);
        studentList.insert(mark:80);
        studentList.insert(mark:90);
        studentList.insert(mark:75);

        // Display all marks
        studentList.displayMarks();

        // Delete students by mark
        studentList.delete(mark:60);
        studentList.delete(mark:90);
        studentList.delete(mark:90);

        // Display all marks after deletion
        studentList.displayMarks();
    }
}
```

```java
package com.student;

public class Student {
    int mark;
    Student next;

    public Student(int mark) {
        this.mark = mark;
    }

}
```

```java
package com.student;

public class StudentMarksList {
    private Student head;

    public void insert(int mark) {
        Student newStudent = new Student(mark);
        if (head == null || mark < head.mark) {
            newStudent.next = head;
            head = newStudent;
        } else {
            Student current = head;
            while (current.next != null && mark > current.next.mark) {
                current = current.next;
            }
            newStudent.next = current.next;
            current.next = newStudent;
        }
    }

    public void displayMarks() {
        Student current = head;
        System.out.print("Student Marks (ascending order): ");
        while (current != null) {
            System.out.print(current.mark + " ");
            current = current.next;
        }
        System.out.println();
    }
```

```java
    public void delete(int mark) {
        if (head == null) {
            System.out.println("List is empty. Cannot delete.");
            return;
        }

        if (head.mark == mark) {
            head = head.next;
            System.out.println("Deleted student with mark " + mark);
            return;
        }

        Student current = head;
        while (current.next != null && current.next.mark != mark) {
            current = current.next;
        }

        if (current.next != null) {
            current.next = current.next.next;
            System.out.println("Deleted student with mark " + mark);
        } else {
            System.out.println("Student with mark " + mark + " not found.");
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/DAY2 2$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/4b3cf6a510e842a524ee9c2d1a09690e/redhat.java/jdt_ws/DAY2\ 2_e5a1f53d/bin com.main.Main
Student Marks (ascending order): 60 70 75 80 90
Deleted student with mark 60
Deleted student with mark 90
Student with mark 90 not found.
Student Marks (ascending order): 70 75 80
```