NAME: ADWAIT S PURAO
UID: 2021300101
EXP NO. :2
AIM: To implement the circular queue functions

THEORY:

A Circular Queue is a special version of queue where the last element of the queue is connected to the first element of the queue forming a circle. The circular queue solves the major limitation of the normal queue. In a normal queue, after a bit of insertion and deletion, there will be non-usable empty space.

It is also known as a ***Ring Buffer***.

Operations on circular queue

- **Front:** It is used to get the front element from the Queue.
- **Rear:** It is used to get the rear element from the Queue.
- **enQueue(value):** This function is used to insert the new value in the Queue. The new element is always inserted from the rear end.
- **deQueue():** This function deletes an element from the Queue. The deletion in a Queue always takes place from the front end.

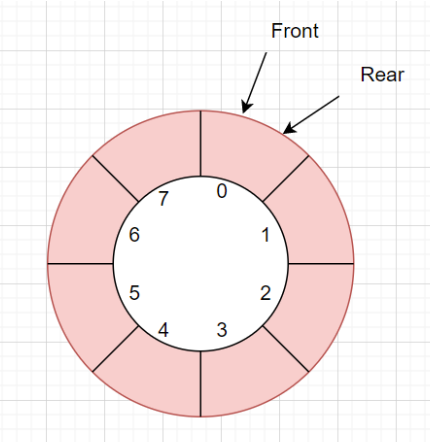
Applications on circular queue

- **Memory management:** The circular queue provides memory management. As we have already seen that in linear queue, the memory is not managed very efficiently. But in case of a circular queue, the memory is managed efficiently by placing the elements in a location which is unused.
- **CPU Scheduling:** The operating system also uses the circular queue to insert the processes and then execute them.
- **Traffic system:** In a computer-control traffic system, traffic light is one of the best examples of the circular queue. Each light of traffic light gets ON one by one after every jinterval of time. Like red light gets ON for one minute then yellow light for one minute and then green light. After green light, the red light gets ON.

Initial condition , now queue is empty:

Front =0, Rear=0

Front==Rear

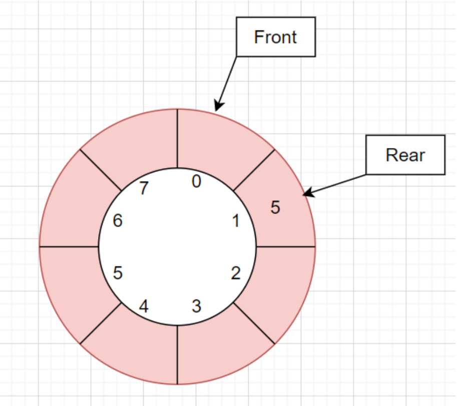


Enqueue operation

Check whether queue is Full or not

Increment rear by one

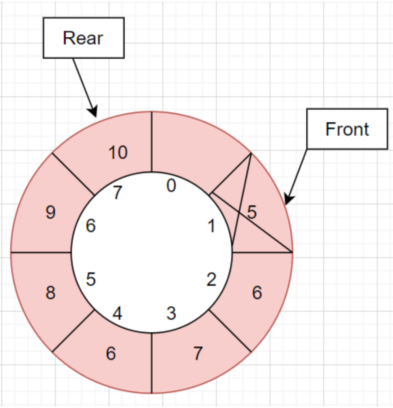
Insert element at index=rear



Dequeue operation

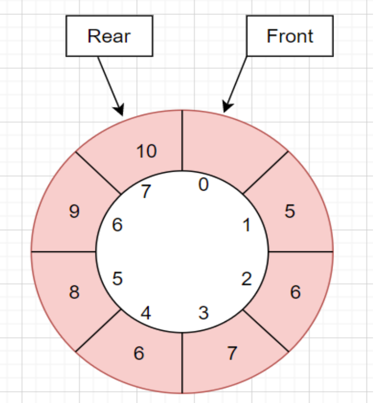
Check whether queue is empty or not

Increment front by one



Queue is Full:

When (rear+1)%size==front



ALGORITHM:

Class Queue:

Data members:

Size, front, top

1) Queue constructor

    Initialize size, array

    And set front=rear=0

2) Boolean isEmpty() method

    Return true if front==rear

```
3) Boolean isFull() method
    Return true if (rear+1)%size == front
4) Void getRear() method
    If Queue is not empty
        print a[rear]
    Else
        Print Queue is empty
5) Void getFront() method
    If Queue is not empty
        Print  a[(front+1)%size]
    Else
        Print Queue is empty
6) Void display()  method
    Initialize a new variable f to front
    If Queue is not empty
        While f is not equal to rear
            f=(f+1)%size
            print a[f]
    Else
        Print queue is empty
7) Void Enq(int element) method
    If Queue is Full
        Print Overflow State
    Else
        rear=(rear+1)%size
        Insert element at index rear
        Print Enqueued element
8) Void Deq() method
    If Queue is empty
        Print Underflow state
    Else
        Print Dequeued a[(front+1)%size]
        front=(front+1)%size
```

Class DSA_EXP_2:

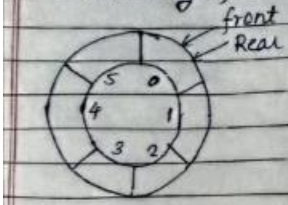
```
1)Main method
    Take input n the size of the Queue
    Create an object q of class queue
    Initialize flag to zero
Do while flag is not equal to 1
    Print the menu
    Take the choice as input in variable ch
        Switch(ch)
            Case 1:
                Take input the element you want to enqueuer
                Call the Enq method from queue class
                Show the current status of Queue
                Call the display method of the queue
            Case 2:
                Call the Dequeue method of the queue class
                Show the current status of Queue
                Call the display method of the queue
            Case 3:
                Show the current status of Queue
                Call the display method of the queue
                Call the getFront method to show the Front-most element
            Case 4:
                Show the current status of Queue
                Call the display method of the queue
                Call the getRear method to show the Rear-most element
            Case 5:
                Print program finished
                Set flag to 1
            Default case:
                Print invalid choice
```

PROBLEM SOLVING ON CONCEPT:

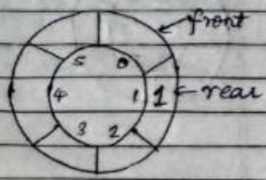
Name: Adwait. S. Purao
UID: 2021300101
Batch: B2

Date
Page

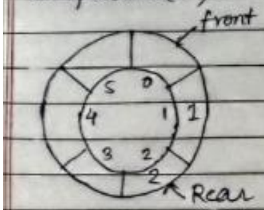
Initially, front=rear=0



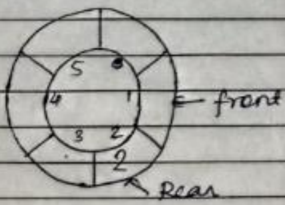
Enqueue (1)



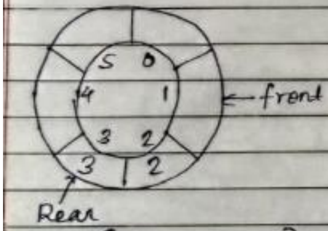
Enqueue (2)



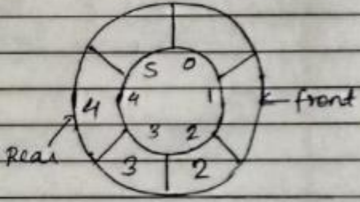
Dequeue()



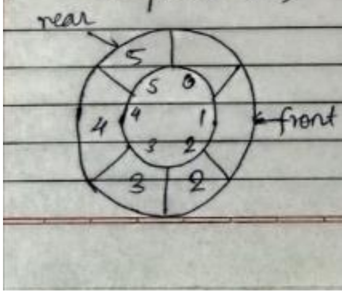
Enqueue (3)



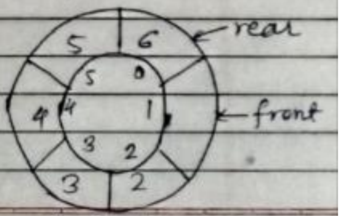
Enqueue (4)



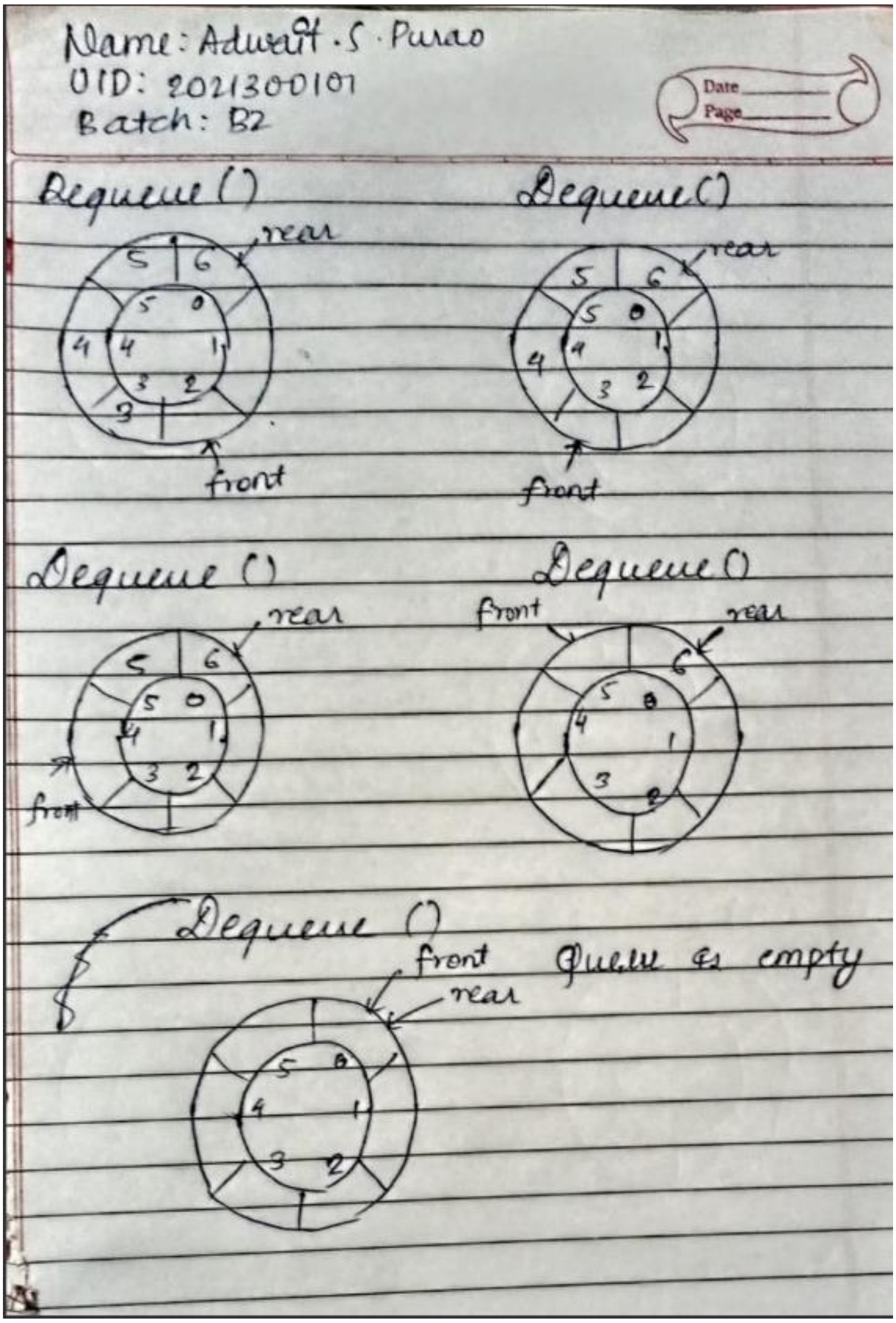
Enqueue (5)



Enqueue (6)



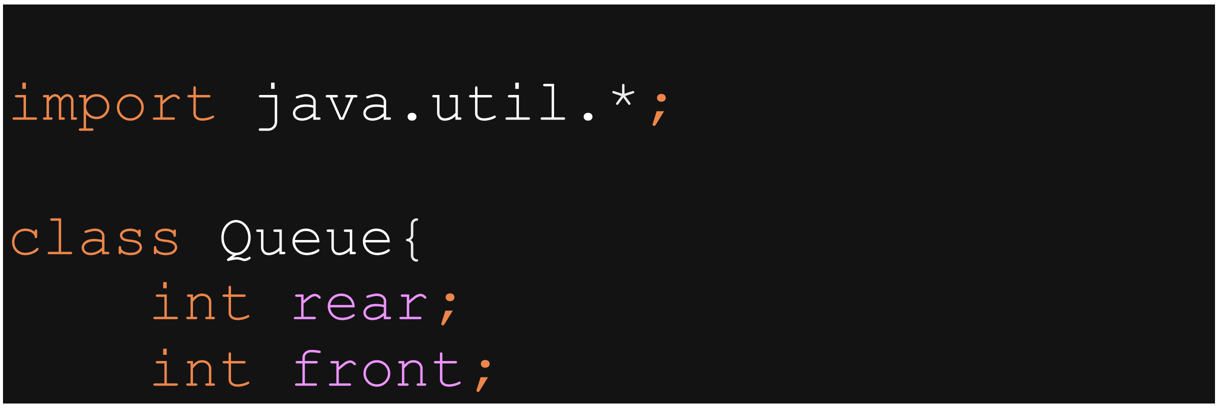
Queue is full now



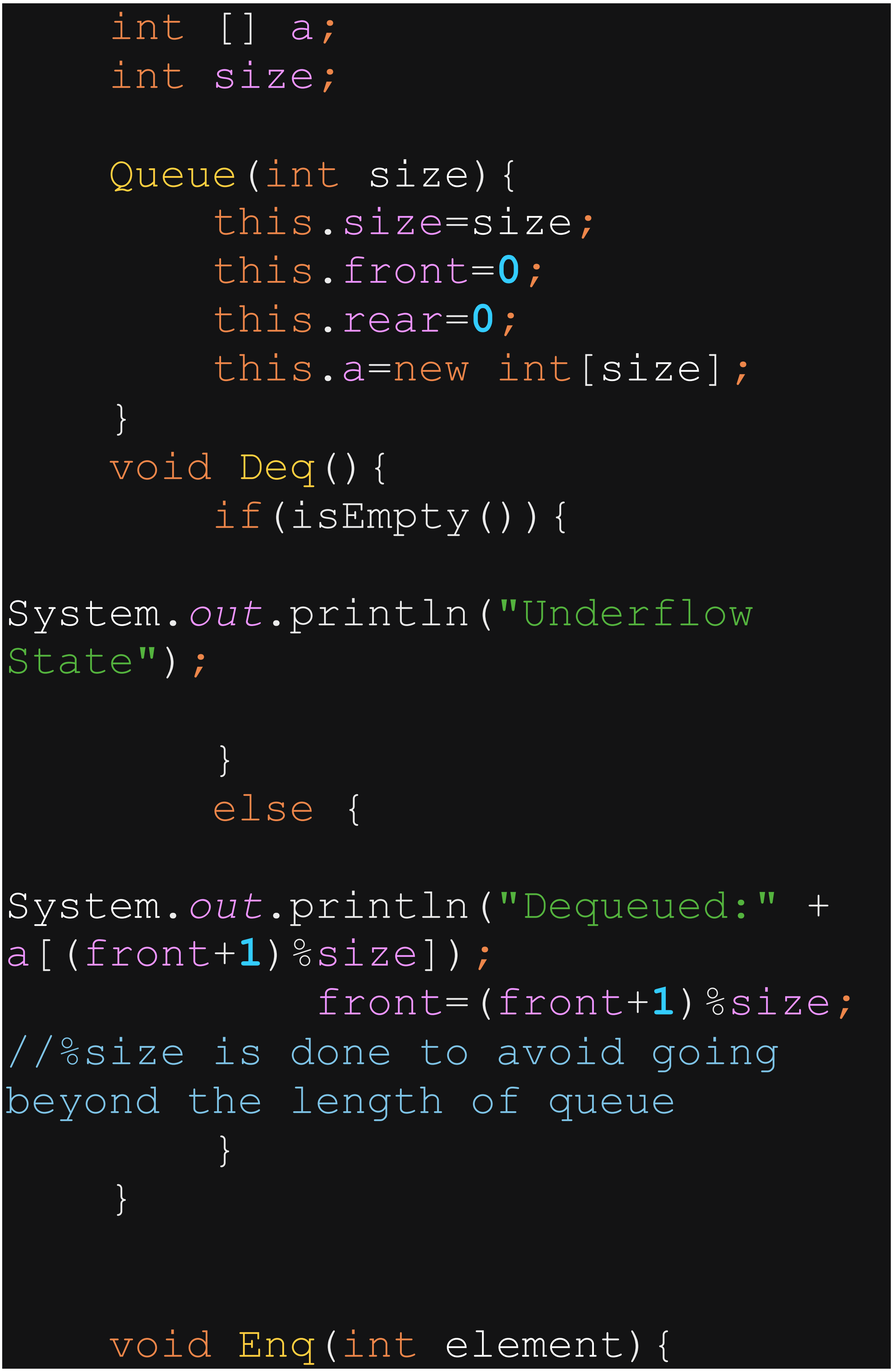
Name: Adwait .S. Purao
UID: 2021300101
Batch: B2

Date
Page

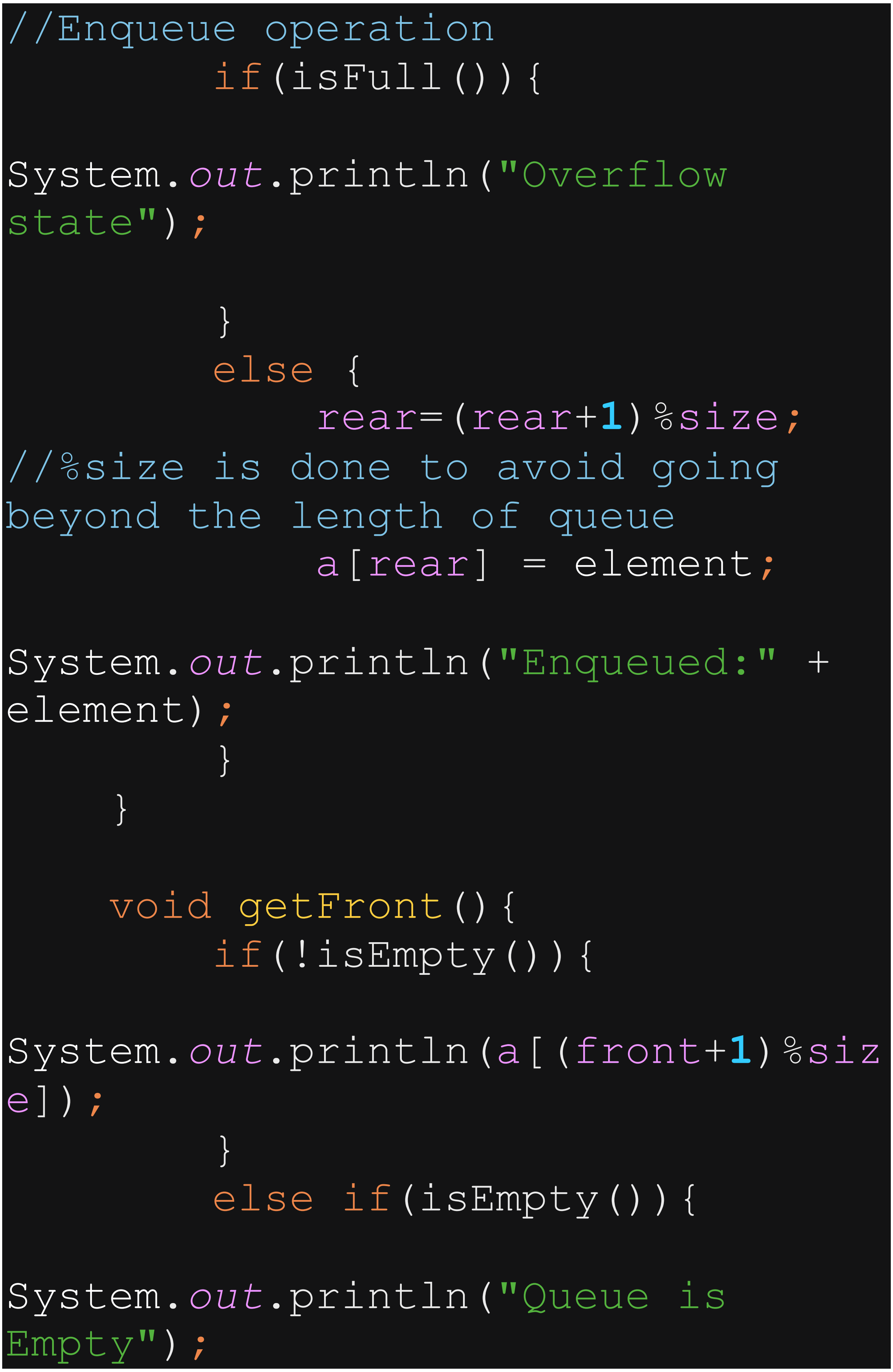
CODE:



```
import java.util.*;

class Queue{
    int rear;
    int front;
```

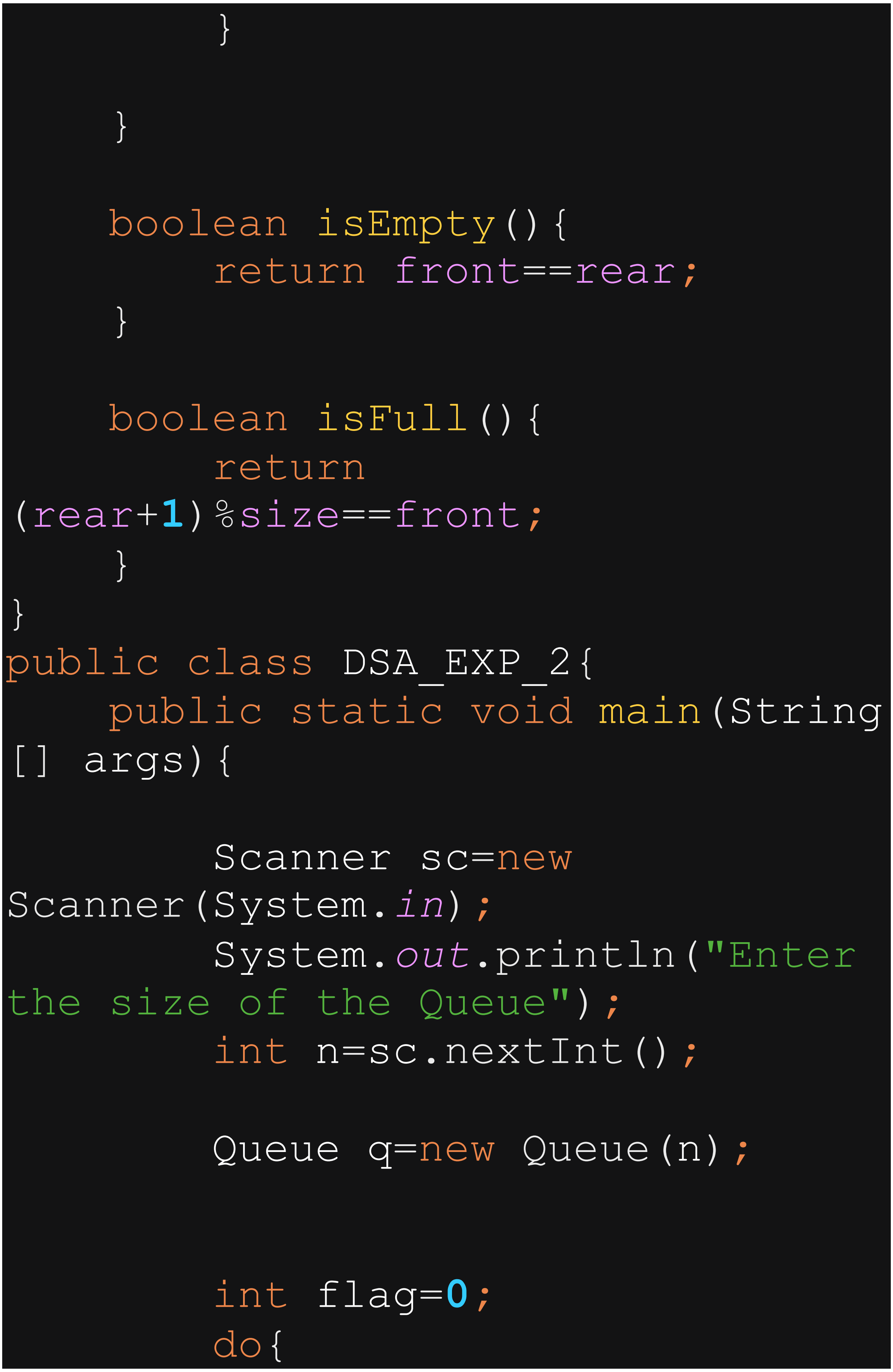


```
    int [] a;
    int size;

    Queue(int size){
        this.size=size;
        this.front=0;
        this.rear=0;
        this.a=new int[size];
    }
    void Deq(){
        if(isEmpty()){

System.out.println("Underflow
State");

        }
        else {

System.out.println("Dequeued:" +
a[(front+1)%size]);
            front=(front+1)%size;
//%size is done to avoid going
beyond the length of queue
        }
    }


    void Enq(int element){
```

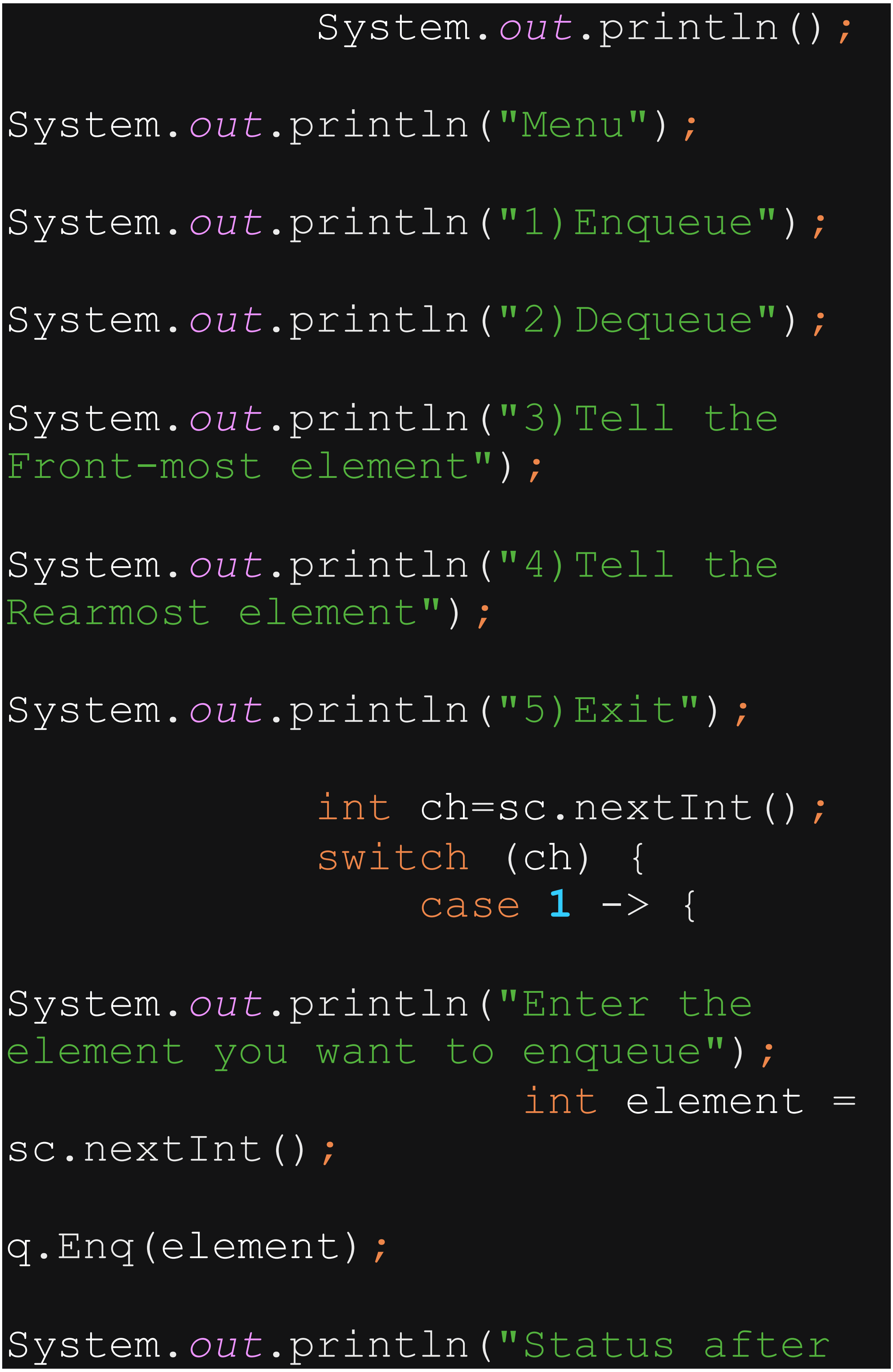


```
//Enqueue operation
        if(isFull()){

System.out.println("Overflow
state");

        }
        else {
            rear=(rear+1)%size;
//%size is done to avoid going
beyond the length of queue
            a[rear] = element;

System.out.println("Enqueued:" +
element);
        }
    }

    void getFront(){
        if(!isEmpty()){

System.out.println(a[(front+1)%siz
e]);
        }
        else if(isEmpty()){

System.out.println("Queue is
Empty");
```

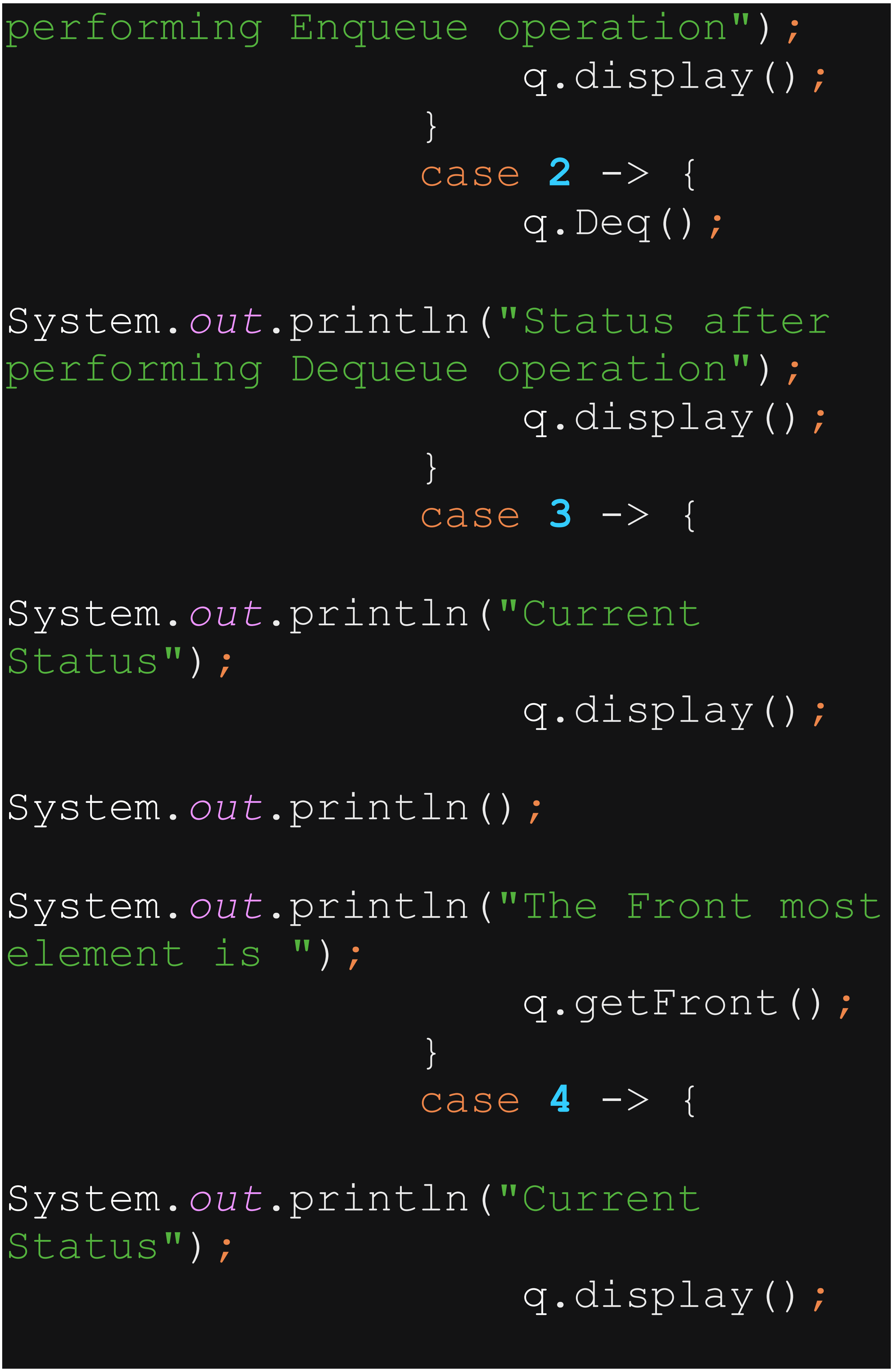
```
        }
    }

    void display(){
        int f=front;
        if(!isEmpty()){
            while(f!=rear){
                f=(f+1)%size;

System.out.print(a[f]+" ");
            }
        }
        else{

System.out.println("Queue is 
empty");
        }
    }

    void getRear(){
        if(!isEmpty()){

System.out.println(a[rear]);
        }
        else if(isEmpty()){

System.out.println("Queue is 
Empty");
```



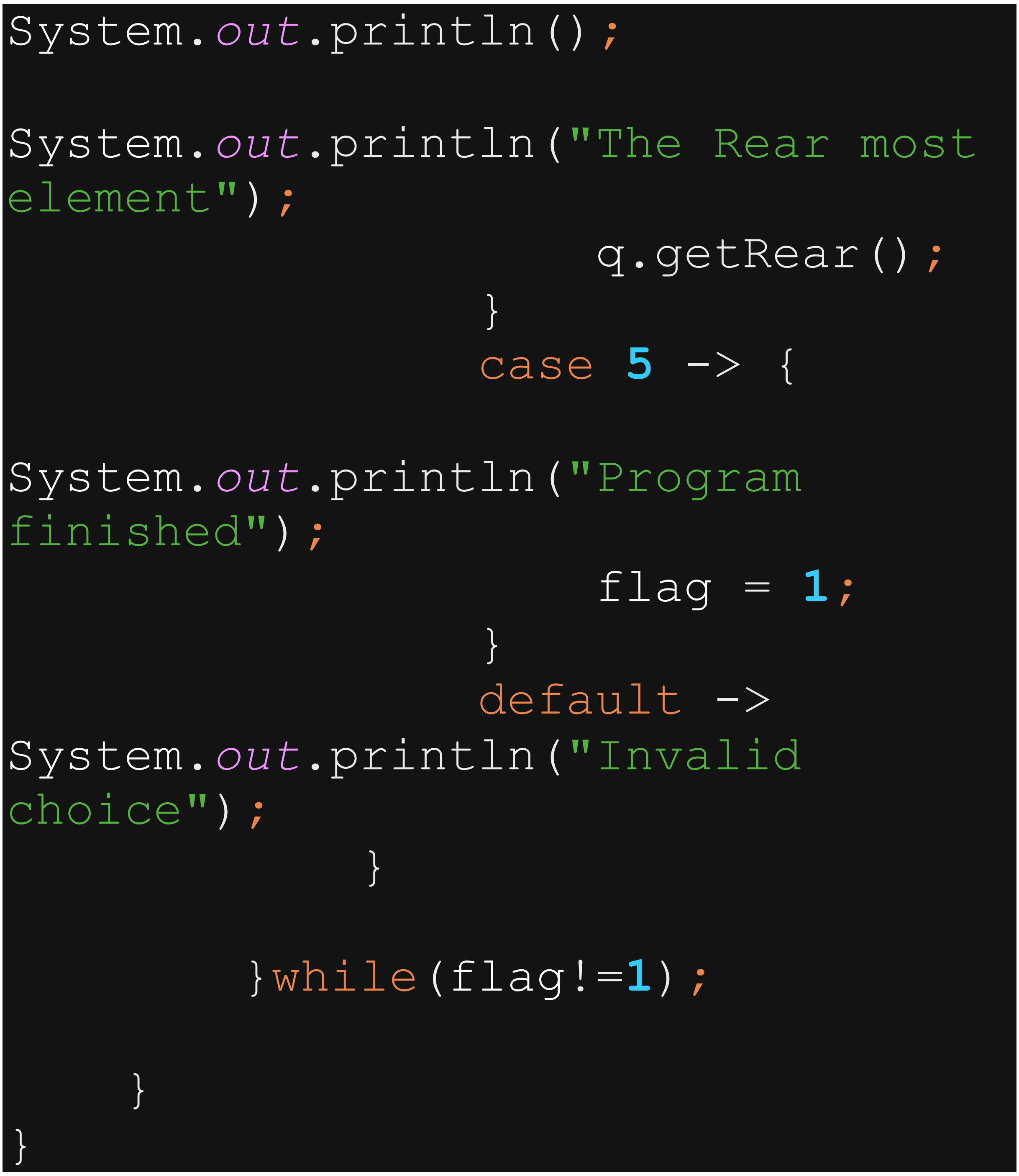
```
        }

    }

    boolean isEmpty(){
        return front==rear;
    }

    boolean isFull(){
        return
(rear+1)%size==front;
    }
}
public class DSA_EXP_2{
    public static void main(String
[] args){

        Scanner sc=new
Scanner(System.in);
        System.out.println("Enter
the size of the Queue");
        int n=sc.nextInt();

        Queue q=new Queue(n);


        int flag=0;
        do{
```



```
            System.out.println();

System.out.println("Menu");

System.out.println("1)Enqueue");

System.out.println("2)Dequeue");

System.out.println("3)Tell the
Front-most element");

System.out.println("4)Tell the
Rearmost element");

System.out.println("5)Exit");

            int ch=sc.nextInt();
            switch (ch) {
                case 1 -> {

System.out.println("Enter the
element you want to enqueue");
                    int element =
sc.nextInt();

q.Enq(element);

System.out.println("Status after
```

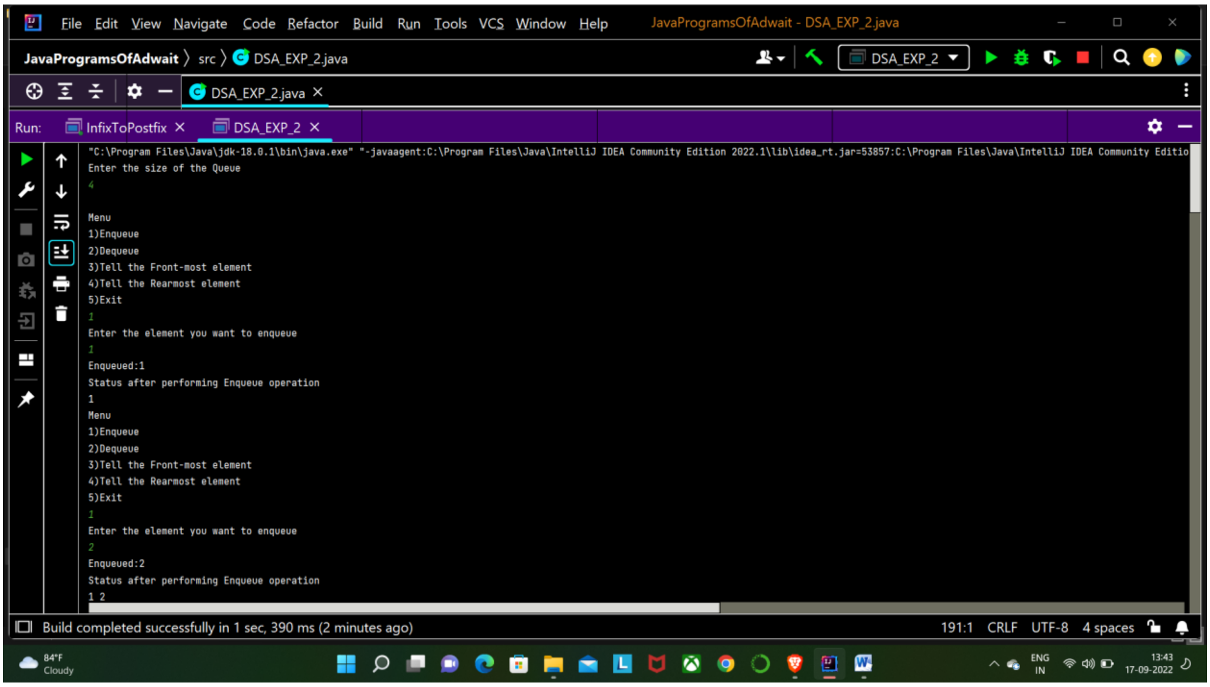


```
performing Enqueue operation");
                    q.display();
                }
                case 2 -> {
                    q.Deq();

System.out.println("Status after
performing Dequeue operation");
                    q.display();
                }
                case 3 -> {

System.out.println("Current
Status");
                    q.display();

System.out.println();

System.out.println("The Front most
element is ");
                    q.getFront();
                }
                case 4 -> {

System.out.println("Current
Status");
                    q.display();
```



```
System.out.println();

System.out.println("The Rear most
element");
                    q.getRear();
                }
                case 5 -> {

System.out.println("Program
finished");
                    flag = 1;
                }
                default ->
System.out.println("Invalid
choice");
            }

        }while(flag!=1);

    }
}
```

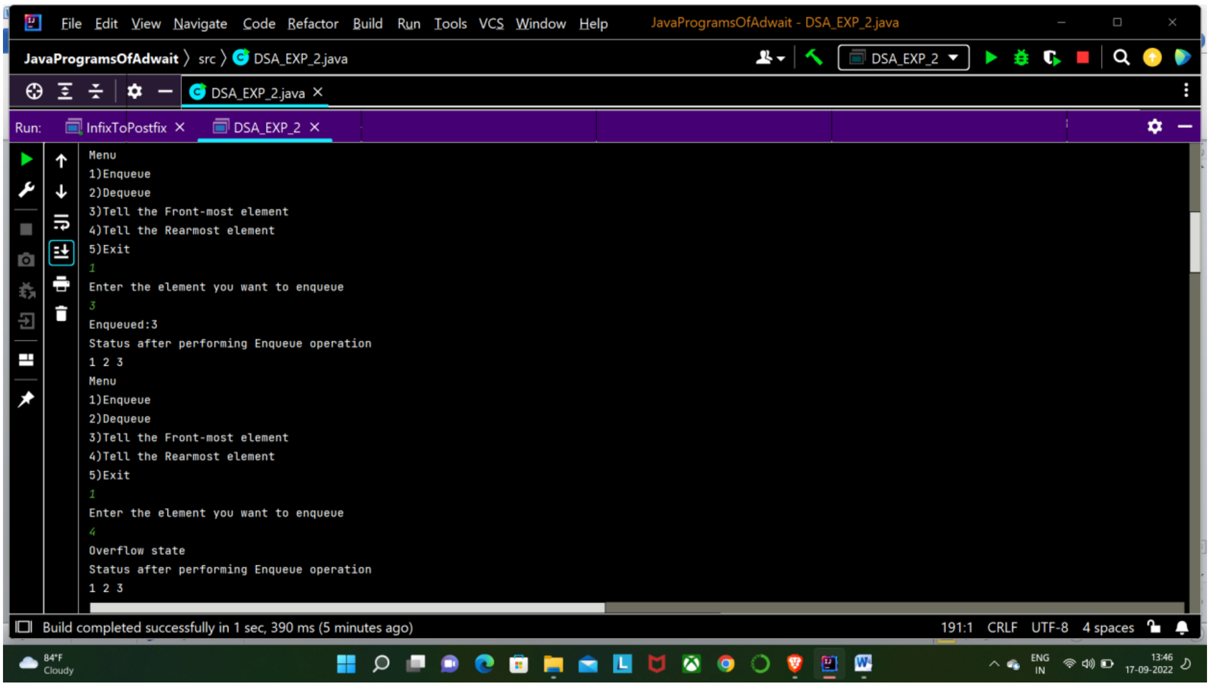
OUTPUT SCREENSHOT:



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2

```
"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\Java\IntelliJ IDEA Community Edition 2022.1\lib\idea_rt.jar=53857:C:\Program Files\Java\IntelliJ IDEA Community Editio
Enter the size of the Queue
4

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
1
Enter the element you want to enqueue
1
Enqueued:1
Status after performing Enqueue operation
1
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
1
Enter the element you want to enqueue
2
Enqueued:2
Status after performing Enqueue operation
1 2
```

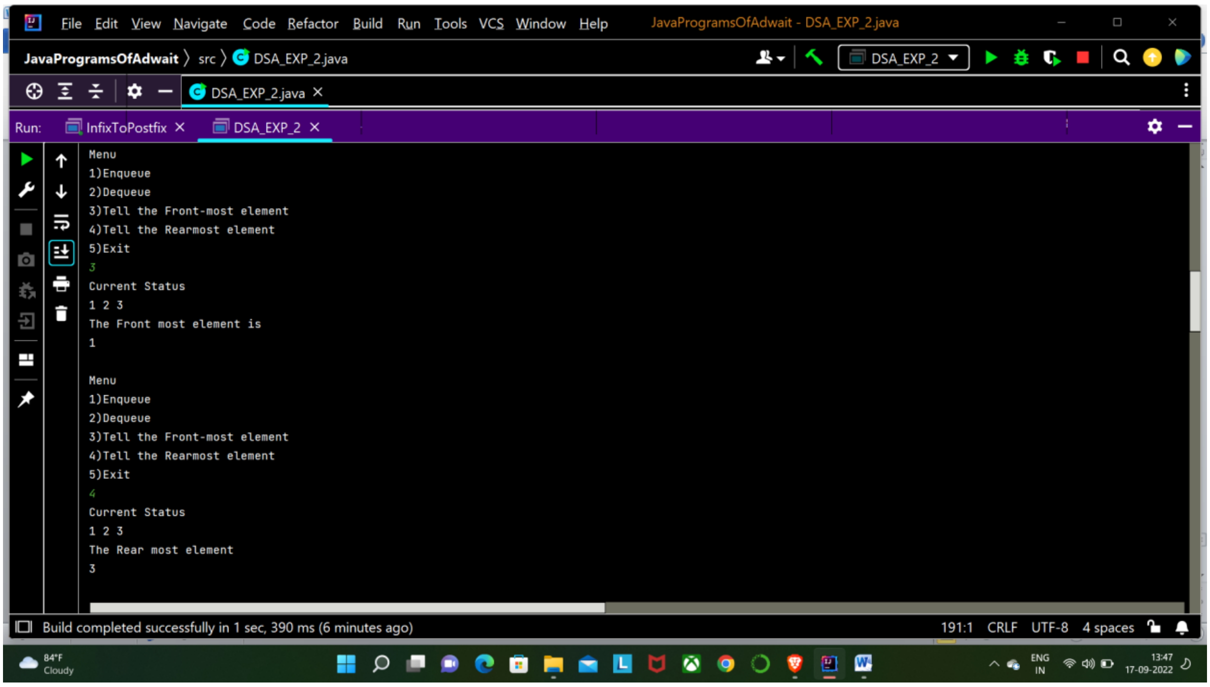
Build completed successfully in 1 sec, 390 ms (2 minutes ago)
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN
13:43
17-09-2022



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2

```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
1
Enter the element you want to enqueue
3
Enqueued:3
Status after performing Enqueue operation
1 2 3
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
1
Enter the element you want to enqueue
4
Overflow state
Status after performing Enqueue operation
1 2 3
```

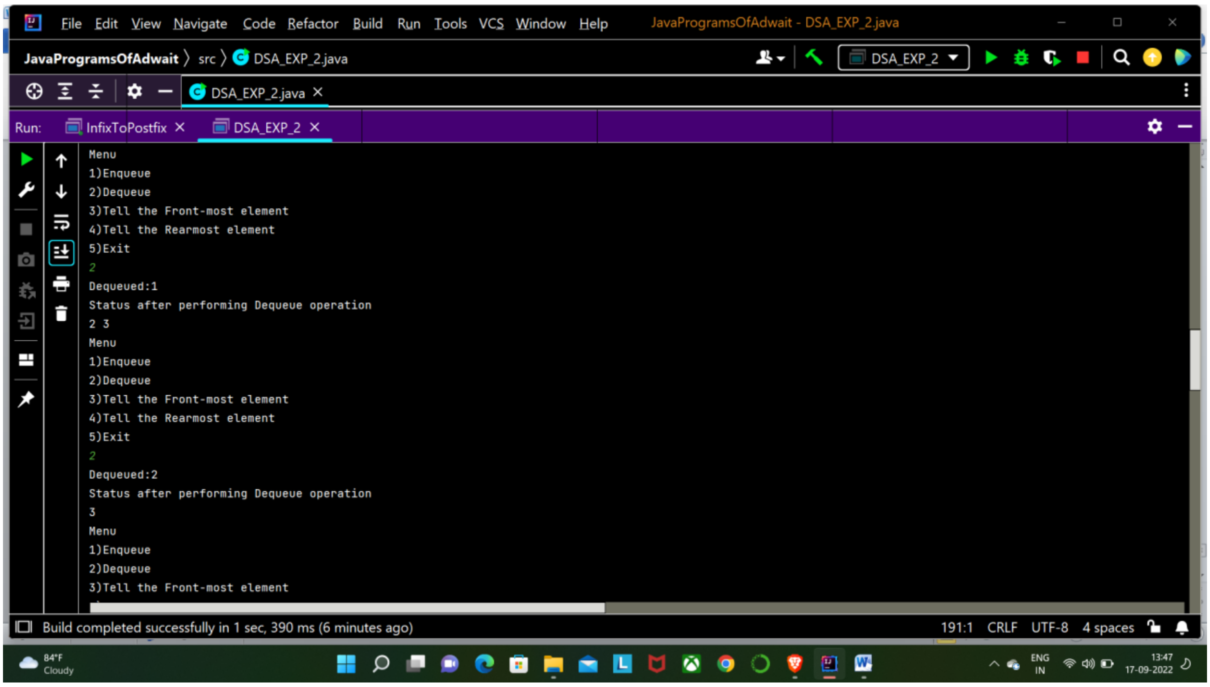
Build completed successfully in 1 sec, 390 ms (5 minutes ago)
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN
13:46
17-09-2022


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
DSA_EXP_2.java
Run: InfixToPostfix DSA_EXP_2

```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
3
Current Status
1 2 3
The Front most element is
1

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
4
Current Status
1 2 3
The Rear most element
3
```

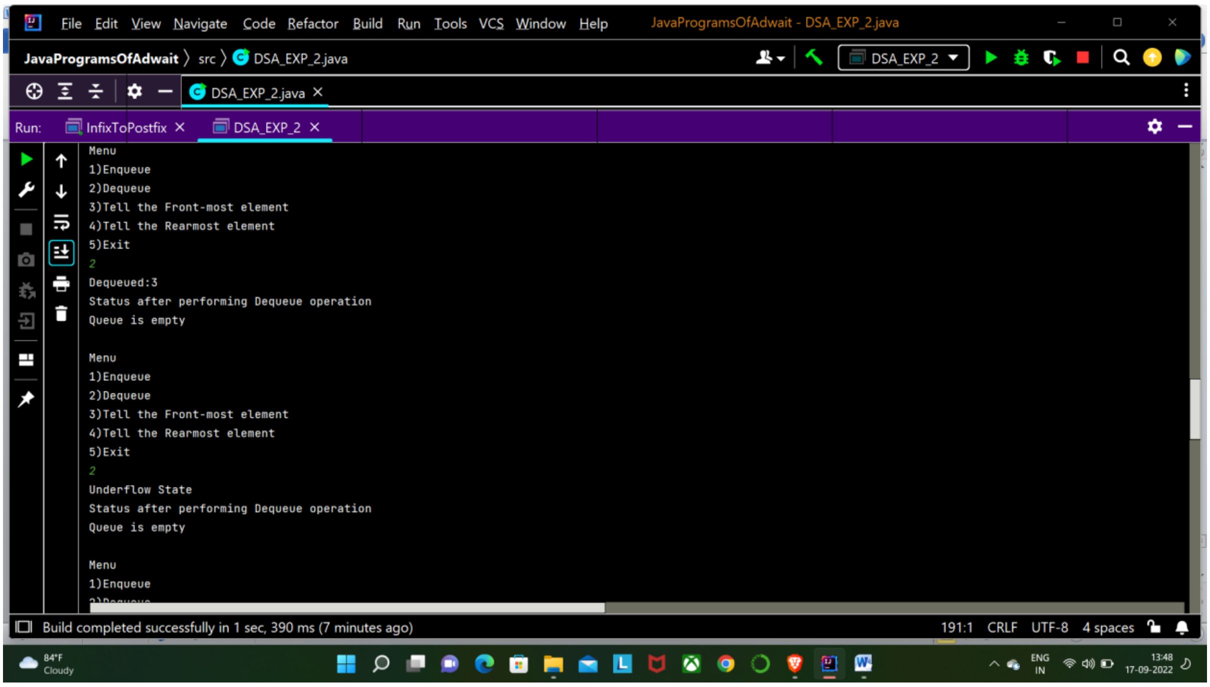
Build completed successfully in 1 sec, 390 ms (6 minutes ago)

191:1 CRLF UTF-8 4 spaces
84°F
Cloudy
ENG
IN
13:47
17-09-2022


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
DSA_EXP_2.java
Run: InfixToPostfix DSA_EXP_2

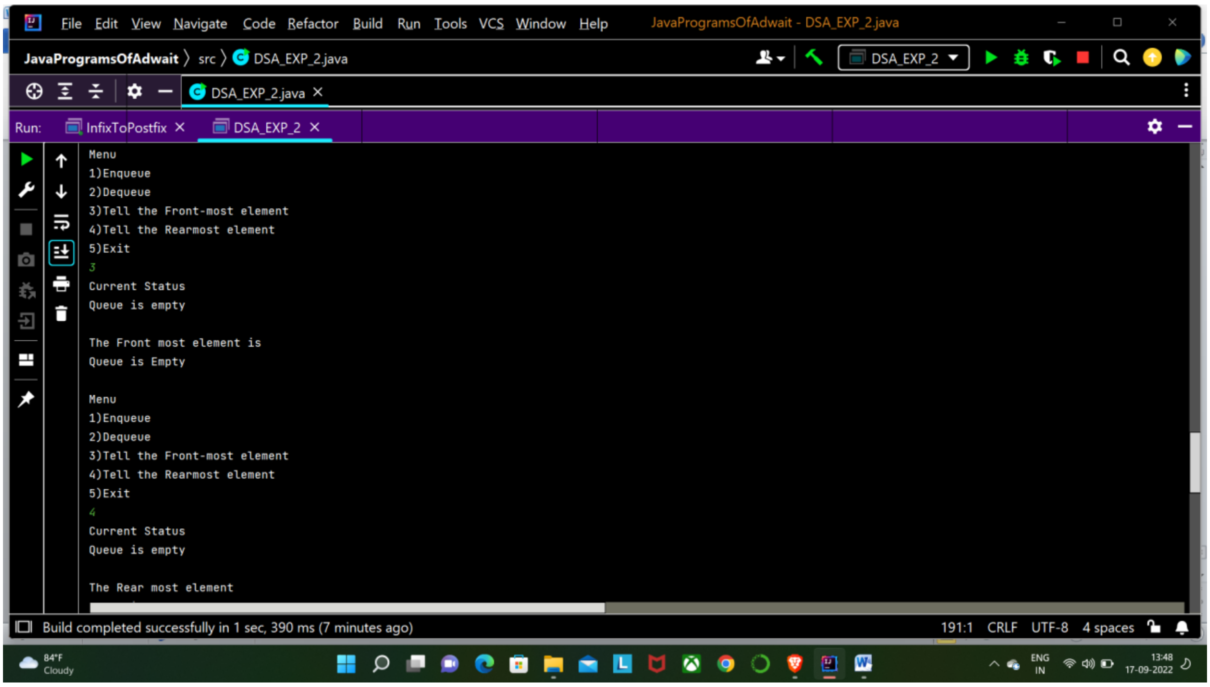
```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
2
Dequeued:1
Status after performing Dequeue operation
2 3
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
2
Dequeued:2
Status after performing Dequeue operation
3
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
```

Build completed successfully in 1 sec, 390 ms (6 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN
13:48 17-09-2022

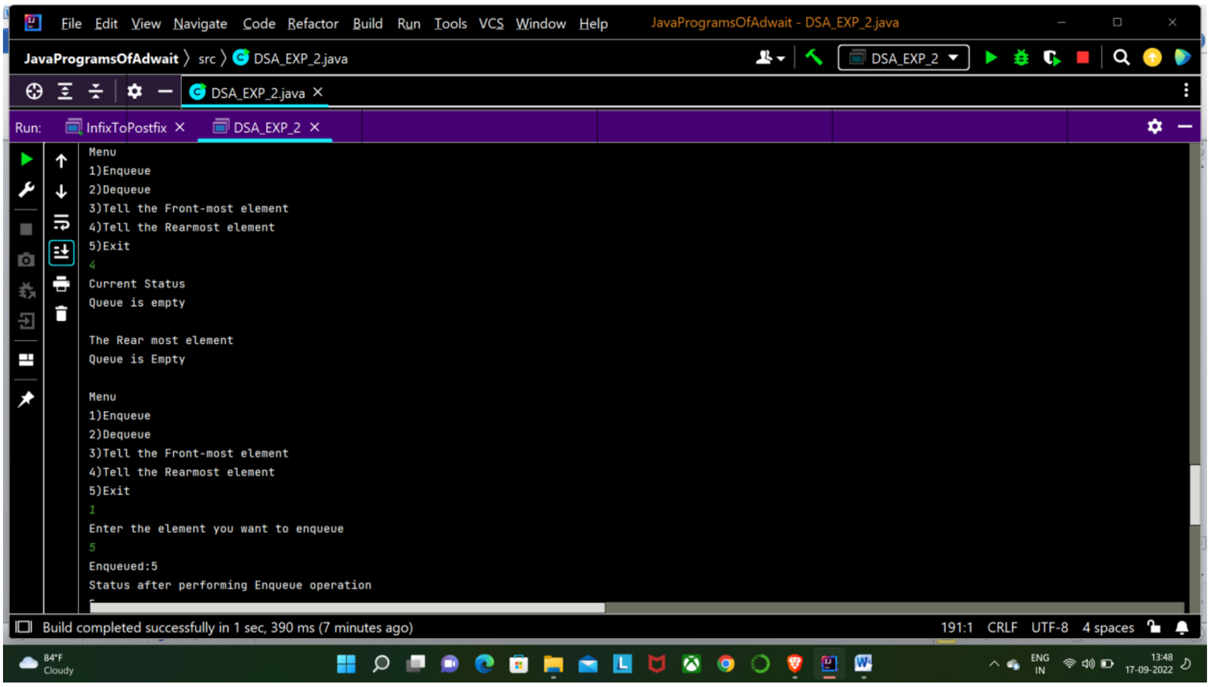
```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
2
Dequeued:3
Status after performing Dequeue operation
Queue is empty

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
2
Underflow State
Status after performing Dequeue operation
Queue is empty

Menu
1)Enqueue
2)Dequeue
```

Build completed successfully in 1 sec, 390 ms (7 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN
13:48 17-09-2022

```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
3
Current Status
Queue is empty

The Front most element is
Queue is Empty

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
4
Current Status
Queue is empty

The Rear most element
```

Build completed successfully in 1 sec, 390 ms (7 minutes ago)

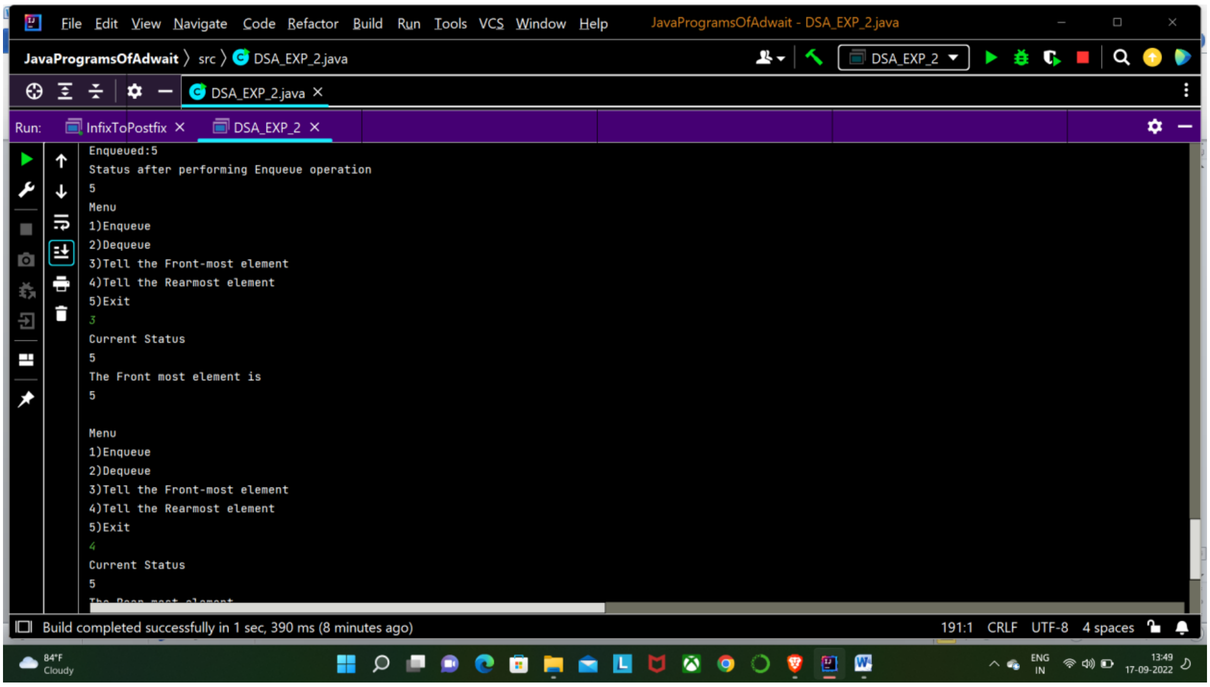


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2

```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
4
Current Status
Queue is empty

The Rear most element
Queue is Empty

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
1
Enter the element you want to enqueue
5
Enqueued:5
Status after performing Enqueue operation
5
```

Build completed successfully in 1 sec, 390 ms (7 minutes ago)

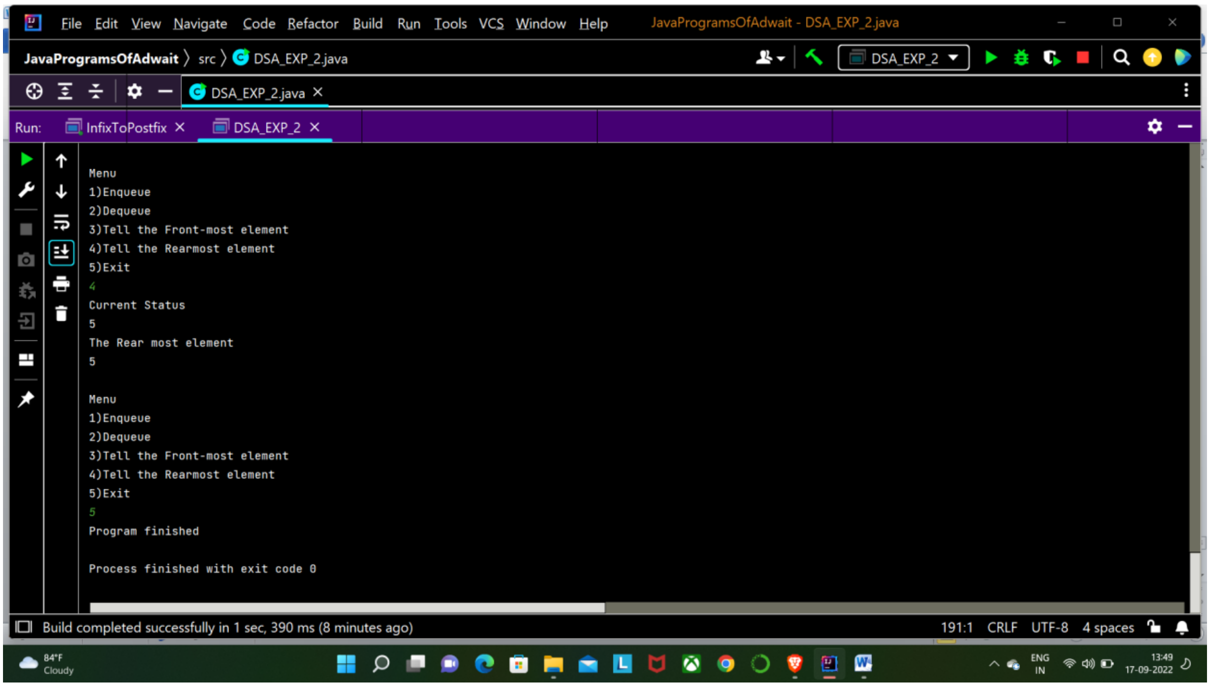
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN 13:48 17-09-2022



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
Run: InfixToPostfix DSA_EXP_2

```
Enqueued:5
Status after performing Enqueue operation
5
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
3
Current Status
5
The Front most element is
5

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
4
Current Status
5
The Rear most element
```

Build completed successfully in 1 sec, 390 ms (8 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProgramsOfAdwait - DSA_EXP_2.java
JavaProgramsOfAdwait 〉 src 〉 DSA_EXP_2.java
DSA_EXP_2
DSA_EXP_2.java
Run: InfixToPostfix DSA_EXP_2
Build completed successfully in 1 sec, 390 ms (8 minutes ago)
191:1 CRLF UTF-8 4 spaces
84°F Cloudy
ENG IN
13:49 17-09-2022

```
Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
4
Current Status
5
The Rear most element
5

Menu
1)Enqueue
2)Dequeue
3)Tell the Front-most element
4)Tell the Rearmost element
5)Exit
5
Program finished

Process finished with exit code 0
```

CONCLUSION:

In this experiment we learnt about the circular queues. We learnt their advantages over the normal queues i.e. they save space and are reusable. We performed the Enqueue, Dequeue , getFront and getRear operations on the queue with a menu driven program. But in circular queue one space is always wasted which is more preferred than wasting a lot of memory using normal queues.