Practical-4

Aim : Write a program to design a circular queue(k) which Should implement the below functions

Enqueue

Dequeue

Front

Rear

Terms :-

・ Instance Variables: ・

o queue: An integer array to store the elements of the circular queue.

o front: An integer representing the front index of the queue.

o rear: An integer representing the rear index of the queue.

o size: An integer representing the current size of the queue.

o capacity: An integer representing the maximum capacity of the queue.

・ Constructor (CircularQueue(int k)): ・

o Initializes the queue with a specified capacity k.

o Initializes front to 0, rear to -1, and size to 0.

・ Methods: ・

o enQueue(int value): Adds an element to the rear of the queue.

o deQueue(): Removes an element from the front of the queue.

o Front(): Returns the element at the front of the queue without removing it.

o Rear(): Returns the element at the rear of the queue without removing it.

o isEmpty(): Checks if the queue is empty.

o isFull(): Checks if the queue is full.

・ Main Method: ・

o Creates an instance of CircularQueue with a capacity of 5.

o Demonstrates various operations on the circular queue like enqueue, dequeue, checking front, checking rear, etc.

Algorithm:-

1. Initialization:

・ Create a CircularQueue instance cq with a capacity of 5.

2. Enqueue Operation:

・ enQueue(int value)

o Check if the queue is not full (!isFull()).

o Calculate the new rear index using circular logic.

o Place the value at the new rear index.

o Increment the size.

o Return true.

3. Dequeue Operation:

・ deQueue()

o Check if the queue is not empty (!isEmpty()).

o Calculate the new front index using circular logic.

o Decrement the size.

o Return true.

4. Front Operation:

・ Front()

o Check if the queue is not empty.

o Return the element at the front index.

5. Rear Operation:

・ Rear()

o Check if the queue is not empty.

o Return the element at the rear index.

6. isEmpty Operation:

・ isEmpty()

o Check if the size of the queue is 0.

o Return true if the queue is empty.

7. isFull Operation:

- isFull()

o Check if the size of the queue is equal to the capacity.

o Return true if the queue is full.

8. Main Method Execution:

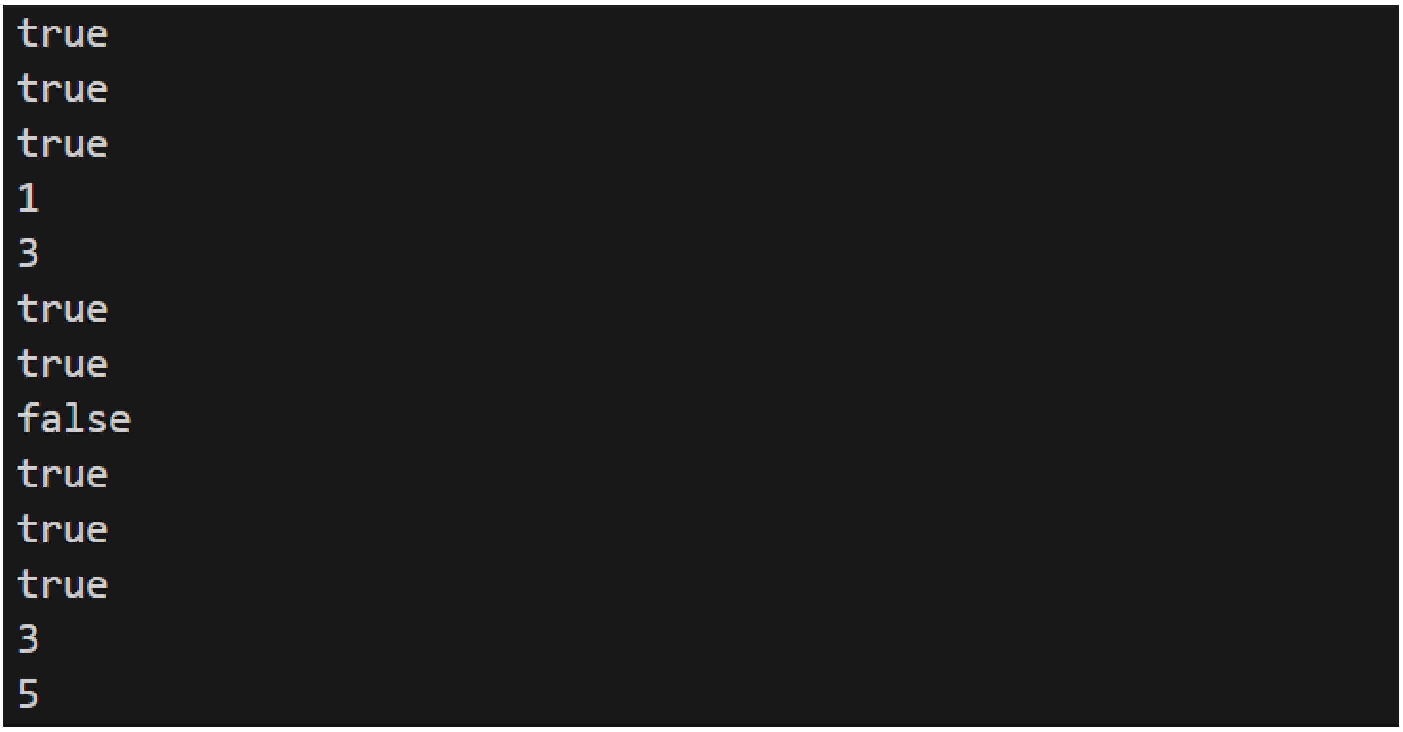
- Instantiate a CircularQueue object with a capacity of 5.
- Perform a series of enqueuing, dequeuing, and checking operations.
- Print the results of these operations.

Program:-

```
class CircularQueue {
   private int[] queue;
   private int front, rear, size, capacity;

   public CircularQueue(int k) {
      capacity = k;
      queue = new int[k];
      front = 0;
      rear = -1;
      size = 0;
   }
//method  for enqueue
   public boolean enQueue(int value) {
      if (!isFull()) {
         rear = (rear + 1) % capacity;
         queue[rear] = value;
```

```
            size++;
            return true;
        }
        return false;
    }
//method for dequeue
    public boolean deQueue() {
        if (!isEmpty()) {
            front = (front + 1) % capacity;
            size--;
            return true;
        }
        return false;
    }

    public int Front() {
        if (!isEmpty()) {
            return queue[front];
        }
        return -1; // Return -1 if the queue is empty
    }

    public int Rear() {
        if (!isEmpty()) {
            return queue[rear];
        }
        return -1; // Return -1 if the queue is empty
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public boolean isFull() {
        return size == capacity;
    }

    public static void main(String[] args) {
        CircularQueue cq = new CircularQueue(5);
        System.out.println(cq.enQueue(1)); // true
        System.out.println(cq.enQueue(2)); // true
        System.out.println(cq.enQueue(3)); // true
        System.out.println(cq.Front()); // 1
        System.out.println(cq.Rear()); // 3
        System.out.println(cq.enQueue(4)); // true
        System.out.println(cq.enQueue(5)); // true
        System.out.println(cq.enQueue(6)); // false
        System.out.println(cq.isFull()); // true
        System.out.println(cq.deQueue()); // true
        System.out.println(cq.deQueue()); // true
        System.out.println(cq.Front()); // 3
        System.out.println(cq.Rear()); // 5
    }
}
```

Output:-



```
true
true
true
1
3
true
true
false
true
true
true
3
5
```