Assignment No - 2

Name : Parth Gaidhani
22311929/224070 Type your text

PROBLEM STATEMENT: "2. Pizza parlor accepting maximum N orders. Orders are served in FCFS basis. Order once placed can't be cancelled. Write Java program to simulate the system using circular QUEUE."

OBJECTIVES:
To simulate a pizza parlor system where orders are handled in the order they arrive.
To efficiently manage a queue with limited capacity using a **circular queue** data structure.
To ensure that no orders are canceled and the serving follows the FCFS principle.

SYSTEM REQUIREMENTS:

1. **Operating System:** Any (Linux/Windows/MacOS)
2. **Programming Language:** Java
3. **IDE:** Any (Eclipse, IntelliJ, VS Code, etc.)
4. **Memory Requirements:** Standard system requirements for running Java applications.

THEORY:

Circular Queue:
A **circular queue** is a linear data structure that follows the FIFO (First In, First Out) principle but connects the end of the queue back to the front to form a circle. It has a fixed size, and when the queue reaches its maximum capacity, new elements are added in the place of the oldest ones, provided those orders have been served.
Key Operations:
Enqueue: Add an order to the rear of the queue.
Dequeue: Serve (remove) an order from the front of the queue.
IsFull: Check if the queue is full.
IsEmpty: Check if the queue is empty.

CODE:

```
import java.util.Scanner;

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

   // Constructor to initialize the queue
   public CircularQueue(int capacity) {
      this.capacity = capacity;
      queue = new int[capacity];
      front = -1;
      rear = -1;
      size = 0;
   }

   // Check if the queue is full
   public boolean isFull() {
      return size == capacity;
   }

   // Check if the queue is empty
   public boolean isEmpty() {
      return size == 0;
   }

   // Enqueue (add order to queue)
   public void enqueue(int order) {
      if (isFull()) {
         System.out.println("Queue is full. Cannot take more orders.");
      } else {
         rear = (rear + 1) % capacity;
         queue[rear] = order;
         size++;
         if (front == -1) {  // If first element is being added
            front = rear;
         }
         System.out.println("Order " + order + " placed successfully.");
      }
   }

   // Dequeue (serve the order)
   public void dequeue() {
      if (isEmpty()) {
```

```
            System.out.println("No orders to serve.");
        } else {
            System.out.println("Order " + queue[front] + " served.");
            front = (front + 1) % capacity;
            size--;
        }
    }

    // Display current orders in the queue
    public void display() {
        if (isEmpty()) {
            System.out.println("No orders in the queue.");
        } else {
            System.out.print("Current orders in queue: ");
            for (int i = 0; i < size; i++) {
                System.out.print(queue[(front + i) % capacity] + " ");
            }
            System.out.println();
        }
    }
}

public class PizzaParlor {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the maximum number of orders: ");
        int N = sc.nextInt();

        CircularQueue ordersQueue = new CircularQueue(N);
        int choice;

        do {
            System.out.println("\n1. Place Order\n2. Serve Order\n3. Display Orders\n4.
                    Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter order number to place: ");
                    int order = sc.nextInt();
                    ordersQueue.enqueue(order);
                    break;
                case 2:
                    ordersQueue.dequeue();
                    break;
                case 3:
                    ordersQueue.display();
                    break;
                case 4:
```

```
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    } while (choice != 4);

    sc.close();
  }
}
```

OUTPUT:

```
Enter the maximum number of orders: 10

1. Place Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 1
Enter order number to place: 3
Order 3 placed successfully.

1. Place Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 1
Enter order number to place: 4
Order 4 placed successfully.

1. Place Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 2
Order 3 served.

1. Place Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 3
Current orders in queue: 4
```

CONCLUSION:

The simulation of the pizza parlor system using a circular queue effectively manages a limited number of orders, ensuring that the first order placed is the first order served (FCFS principle).

The circular queue prevents overflow by efficiently using available space and ensuring orders are served without any cancellations.