Major Final Projects of Java With DS

1. Create a Restaurant Order Management System using the data structure.

Guidelines for projects:-

Restaurant Order Management System:

1. Class Definition: Create classes for Menu Item, Order, and Restaurant.

2. Data Structures: Use a Queue for customer orders, a Stack for order preparation, and a linked list for menu items.

3. Menu Management: Implement methods in the Restaurant class to add, remove, and display menu items.

4. Order Flow: Develop order taking, error handling, and order preparation. Include order numbering.
5. Order Status: Create methods to display the status of orders in the queue and those being prepared.

Program :

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;
import java.util.Scanner;

class MenuItem {
    String name;
    double price;

    public MenuItem(String name) {
        this.name = name;
        // Generate a random price between 5.0 and 15.0 for each item
        this.price = 5.0 + Math.random() * 10.0;
    }

    @Override
    public String toString() {
        return name + " - $" + price;
    }
}

class Order {
    static int orderNumberCounter = 1;
    int orderNumber;
    Queue<MenuItem> items;

    public Order() {
        this.orderNumber = orderNumberCounter++;
        this.items = new LinkedList<>();
    }

    public void addItem(MenuItem item) {
        items.add(item);
    }

    public Queue<MenuItem> getItems() {
        return items;
    }

    public int getOrderNumber() {
        return orderNumber;
    }
}
```

```java
class Restaurant {
    LinkedList<MenuItem> menu;
    Queue<Order> orderQueue;
    Stack<Order> preparationStack;

    public Restaurant() {
        this.menu = new LinkedList<>();
        this.orderQueue = new LinkedList<>();
        this.preparationStack = new Stack<>();
    }

    public void addMenuItem(MenuItem item) {
        menu.add(item);
    }

    public void removeMenuItem(MenuItem item) {
        menu.remove(item);
    }

    public void displayMenu() {
        System.out.println("Menu:");
        for (MenuItem item : menu) {
            System.out.println(item);
        }
        System.out.println();
    }

    public void takeOrder(Order order) {
        orderQueue.add(order);
        System.out.println("Order #" + order.getOrderNumber() + " taken.");
    }
```

```
  public void displayOrderStatus() {
    System.out.println("Orders in Queue:");
    for (Order order : orderQueue) {
      System.out.println("Order #" + order.getOrderNumber() + ": " + order.getItems());
    }

    System.out.println("\nOrders being Prepared:");
    for (Order order : preparationStack) {
      System.out.println("Order #" + order.getOrderNumber() + ": " + order.getItems());
    }
    System.out.println();
  }

  public void prepareOrders() {
    while (!orderQueue.isEmpty()) {
      Order order = orderQueue.poll();
      preparationStack.push(order);
      System.out.println("Order #" + order.getOrderNumber() + " is being prepared.");
    }
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Restaurant restaurant = new Restaurant();

    // Adding menu items
    System.out.println("Add menu items, type 'done' when finished:");
    while (true) {
      System.out.print("Item: ");
```

```
            String itemName = scanner.next();
            if (itemName.equals("done")) {
                break;
            }
            MenuItem menuItem = new MenuItem(itemName);
            restaurant.addMenuItem(menuItem);
        }

        // Displaying menu
        restaurant.displayMenu();

        // Placing orders
        while (true) {
            Order order = new Order();
            System.out.println("Place order #" + order.getOrderNumber() + " (type 'done' when
finished):");
            while (true) {
                System.out.print("Item: ");
                String itemName = scanner.next();
                if (itemName.equals("done")) {
                    break;
                }
                MenuItem menuItem = new MenuItem(itemName);
                order.addItem(menuItem);
            }

            restaurant.takeOrder(order);

            System.out.print("Do you want to place another order? (yes/no): ");
            String response = scanner.next().toLowerCase();
            if (!response.equals("yes")) {
                break;
            }
        }

        // Displaying order status
        restaurant.displayOrderStatus();

        // Preparing orders
        restaurant.prepareOrders();

        // Displaying final order status
        restaurant.displayOrderStatus();

        scanner.close();
    }
}
```

Explination :

1. MenuItem Class: Represents a menu item with a name and a randomly generated price.

2. Order Class: Represents an order with an order number and a queue of menu items.

3. Restaurant Class: Represents a restaurant with a menu (linked list), an order queue (queue), and a preparation stack (stack). Includes methods to add, remove, and display menu items, take orders, display order status, and prepare orders.

4. Main Class: The main program where the user interacts with the system. It allows the user to input menu items and place orders, and it displays the order status throughout the process.

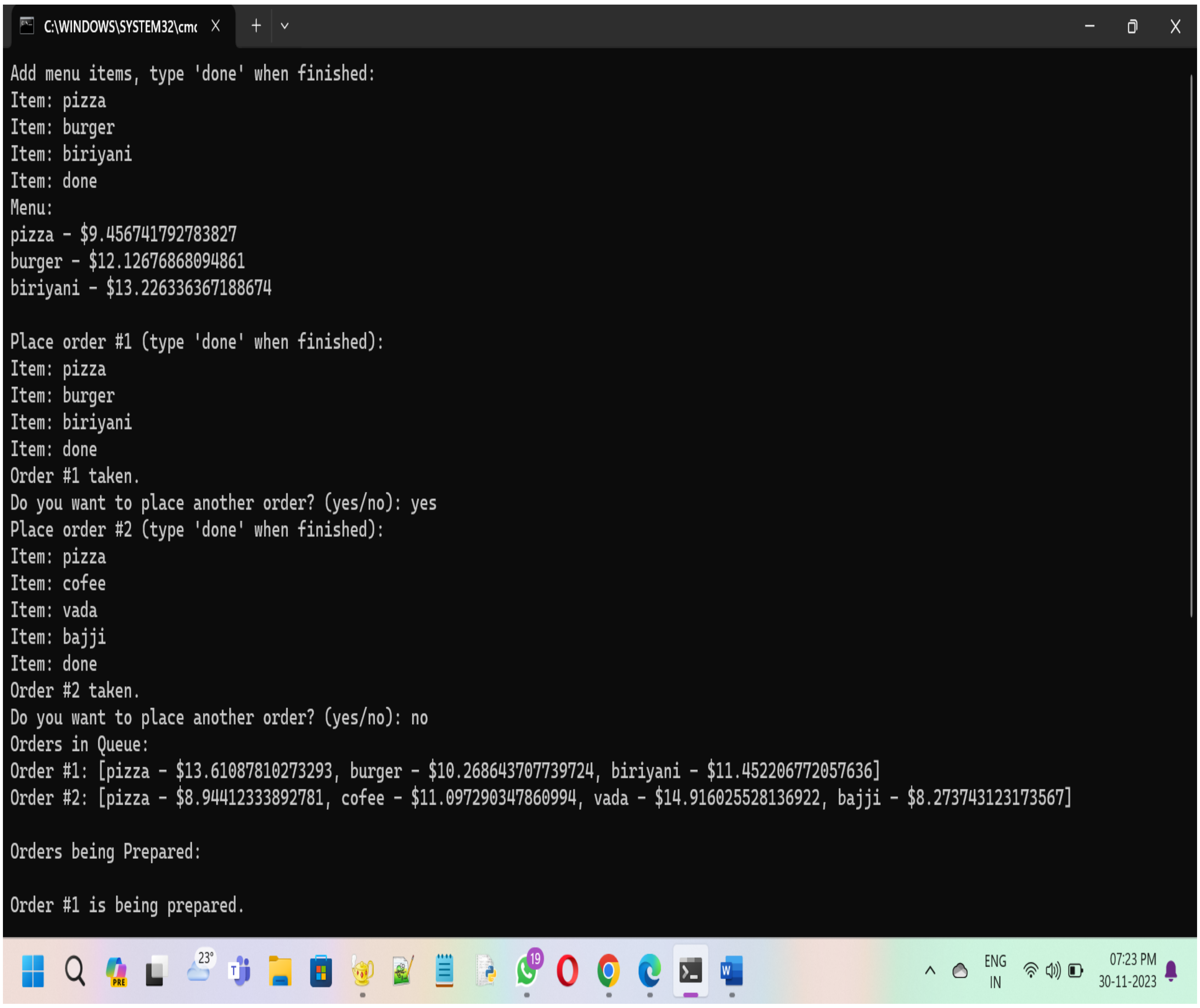
The prices for menu items are generated when the MenuItem objects are created. The user can input menu items and place orders, and the program displays the order status at various stages. The order preparation process involves moving orders from the order queue to the preparation stack.

Output :

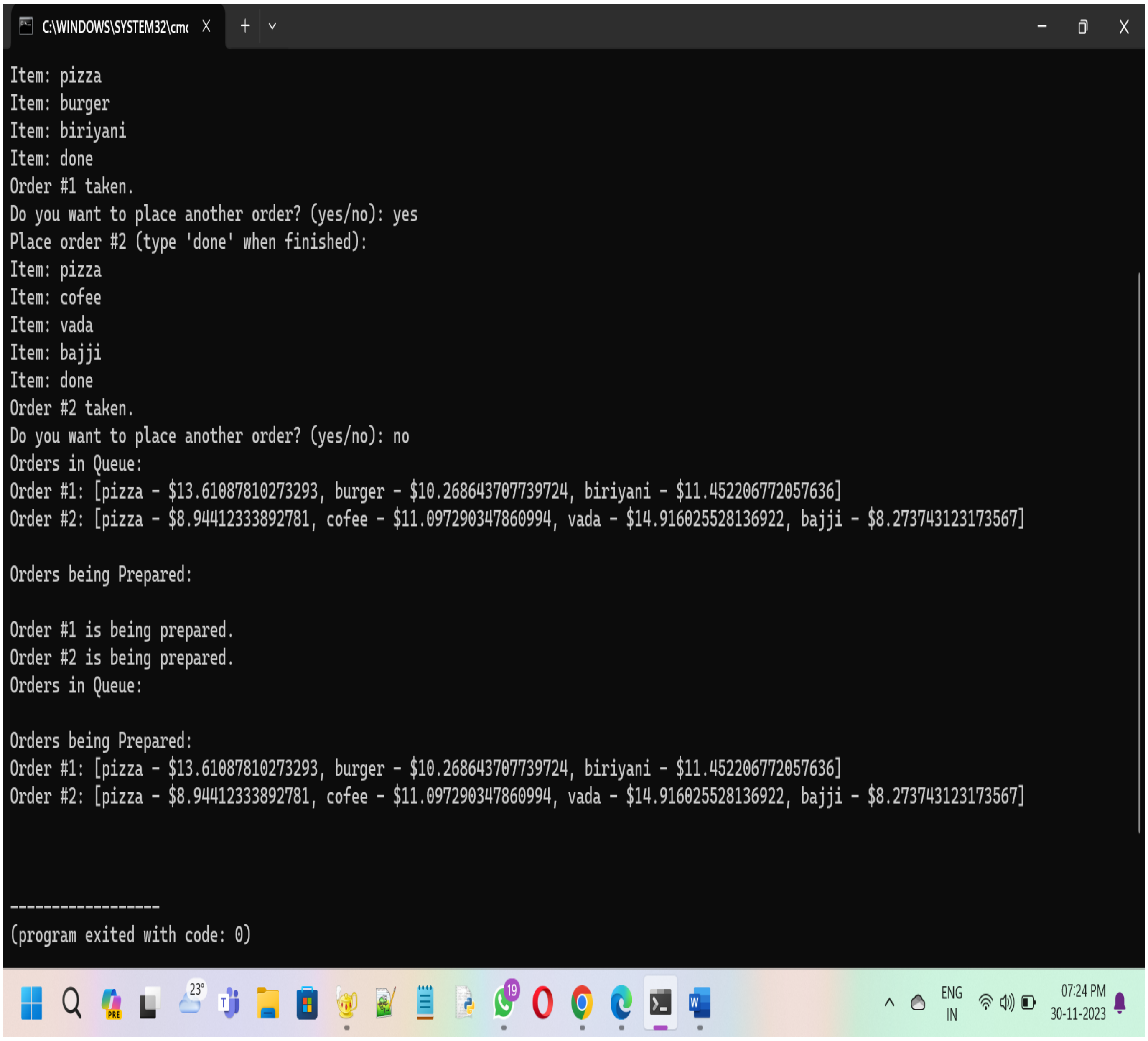
```
Add menu items, type 'done' when finished:
Item: pizza
Item: burger
Item: biriyani
Item: done
Menu:
pizza - $9.456741792783827
burger - $12.12676868094861
biriyani - $13.226336367188674

Place order #1 (type 'done' when finished):
Item: pizza
Item: burger
Item: biriyani
Item: done
Order #1 taken.
Do you want to place another order? (yes/no): yes
Place order #2 (type 'done' when finished):
Item: pizza
Item: cofee
```

```
Item: vada

Item: bajji

Item: done

Order #2 taken.

Do you want to place another order? (yes/no): no

Orders in Queue:

Order #1: [pizza - $13.61087810273293, burger - $10.268643707739724,
biriyani - $11.452206772057636]

Order #2: [pizza - $8.94412333892781, cofee - $11.097290347860994, vada -
$14.916025528136922, bajji - $8.273743123173567]


Orders being Prepared:


Order #1 is being prepared.

Order #2 is being prepared.

Orders in Queue:


Orders being Prepared:

Order #1: [pizza - $13.61087810273293, burger - $10.268643707739724,
biriyani - $11.452206772057636]

Order #2: [pizza - $8.94412333892781, cofee - $11.097290347860994, vada -
$14.916025528136922, bajji - $8.273743123173567]
```

Picture for output which is done in laptop :



```
Add menu items, type 'done' when finished:
Item: pizza
Item: burger
Item: biriyani
Item: done
Menu:
pizza - $9.456741792783827
burger - $12.12676868094861
biriyani - $13.226336367188674

Place order #1 (type 'done' when finished):
Item: pizza
Item: burger
Item: biriyani
Item: done
Order #1 taken.
Do you want to place another order? (yes/no): yes
Place order #2 (type 'done' when finished):
Item: pizza
Item: cofee
Item: vada
Item: bajji
Item: done
Order #2 taken.
Do you want to place another order? (yes/no): no
Orders in Queue:
Order #1: [pizza - $13.61087810273293, burger - $10.268643707739724, biriyani - $11.452206772057636]
Order #2: [pizza - $8.94412333892781, cofee - $11.097290347860994, vada - $14.916025528136922, bajji - $8.273743123173567]

Orders being Prepared:

Order #1 is being prepared.
```



```
Item: pizza
Item: burger
Item: biriyani
Item: done
Order #1 taken.
Do you want to place another order? (yes/no): yes
Place order #2 (type 'done' when finished):
Item: pizza
Item: cofee
Item: vada
Item: bajji
Item: done
Order #2 taken.
Do you want to place another order? (yes/no): no
Orders in Queue:
Order #1: [pizza - $13.61087810273293, burger - $10.268643707739724, biriyani - $11.452206772057636]
Order #2: [pizza - $8.944123338927781, cofee - $11.097290347860994, vada - $14.916025528136922, bajji - $8.273743123173567]

Orders being Prepared:

Order #1 is being prepared.
Order #2 is being prepared.
Orders in Queue:

Orders being Prepared:
Order #1: [pizza - $13.61087810273293, burger - $10.268643707739724, biriyani - $11.452206772057636]
Order #2: [pizza - $8.944123338927781, cofee - $11.097290347860994, vada - $14.916025528136922, bajji - $8.273743123173567]



------------------
(program exited with code: 0)
```