BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Course Code	: CSE 332
Course Title	: Advanced Programming Lab
Course Teacher	: Humayra Ferdous Lecturer, Dept. of CSE

Project Report

Pizza Order and Bill Generator

<u>Team Members:</u>

1. Abdullah Bin Shawkat (22235103204)
2. Ayesha Siddika Mohona(22235103206)
3. Mohammad Mahmudul Hasan Rodra (22235103211)
4. Jayanta Das(22235103221)
5. Md Nazmul Hasan Siam (22235103223)

Introduction

This project is a command-line interface (CLI) pizza ordering system in Java. The goal was to create a simple yet functional ordering system where customers can:

- View the available pizzas and their prices
- Choose the size of the pizza (Small, Medium, Large)
- Select one or more toppings
- Specify the quantity
- Generate an itemized bill (with taxes) and save it as a text file
- View past orders by retrieving previously saved text bills

This project demonstrates **Object-Oriented Programming (OOP)** concepts and is designed to be **extensible** for future enhancements.

Workflow (Class Navigation)

The application is structured with separate Java classes, each with a specific responsibility:

- **Pizza.java**

  Represents a pizza with attributes: name, size, and base price.
  It includes methods to calculate the final price based on size (size multiplier).

- **Topping.java**

  Defines the topping with its name and price.

- **OrderItem.java**

  Represents an ordered pizza item, including:
  Pizza details
  Selected toppings
  Quantity

- **Order.java**

  Manages a list of `OrderItem`s, and calculates:
  Subtotal
  Tax(10%)

Total Amount

- **OrderStorage.java**

  Orders are stored in separate text files (`bill_YYYYMMDD_HHMMSS.txt`).When viewing past orders, the app reads and displays the content of these files.

- **Bill.java**

  Handles:
  Generating the bill in CLI format
  Saving it as a `.txt` file with a timestamped filename for record-keeping

- **PizzaShop.java**

  The main driver class that
  1. Greets the user and shows available pizzas and toppings
  2. Takes user input for pizza choice, size, toppings, and quantity
  3. Manages the ordering process
  4. Saves each bill to a text file
  5. Optionally shows previously saved orders

Code -

```
public class Pizza {
   private String name;
   private String size;
   private double basePrice;
   public Pizza(String name, String size, double basePrice) {
       this.name = name;
       this.size = size;
       this.basePrice = basePrice;
   }
   public String getName() { return name; }
   public String getSize() { return size; }
   public double getBasePrice() { return basePrice; }
   public double getSizeMultiplier() {
       return switch (size.toLowerCase()) {
           case "small" -> 1.0;
           case "medium" -> 1.5;
           case "large" -> 2.0;
           default -> 1.0;
       };
   }
   public double getPrice() {
       return basePrice * getSizeMultiplier();
   }
}
class Topping {
   private String name;
   private double price;
   public Topping(String name, double price) {
       this.name = name;
       this.price = price;
   }
   public String getName() { return name; }
   public double getPrice() { return price; }
}
import java.util.List;
class OrderItem {
   private Pizza pizza;
   private List<Topping> toppings;
   private int quantity;
   public OrderItem(Pizza pizza, List<Topping> toppings, int
quantity) {
       this.pizza = pizza;
       this.toppings = toppings;
       this.quantity = quantity;
   }

   public double getTotalPrice() {
       double toppingCost =
toppings.stream().mapToDouble(Topping::getPrice).sum();
       return (pizza.getPrice() + toppingCost) * quantity;
   }

   @Override
   public String toString() {
       StringBuilder sb = new StringBuilder();
       sb.append(quantity).append(" x
").append(pizza.getSize()).append("
").append(pizza.getName()).append(" -
").append(String.format("৳%.2f",
pizza.getPrice())).append("\n");
       if (!toppings.isEmpty()) {
           sb.append("  Toppings:\n");
           for (Topping t : toppings) {
                sb.append("    ").append(t.getName()).append("
- ").append(String.format("৳%.2f",
t.getPrice())).append("\n");
            }
        }
        sb.append("  Item Total:
").append(String.format("৳%.2f",
getTotalPrice())).append("\n");
        return sb.toString();
    }
}
import java.util.ArrayList;
import java.util.List;
class Order {
    private List<OrderItem> items = new ArrayList<>();
    private final double TAX_RATE = 0.1;
    public void addItem(OrderItem item) {
        items.add(item);
    }
    public double getSubtotal() {
        return
items.stream().mapToDouble(OrderItem::getTotalPrice).sum();
    }
    public double getTax() {
        return getSubtotal() * TAX_RATE;
    }
    public double getTotal() {
        return getSubtotal() + getTax();
    }
    public List<OrderItem> getItems() {
        return items;}}
class OrderStorage {
    private static final String ORDERS_DIR = "orders";
    public static void saveOrder(Order order) {
        try {
            File dir = new File(ORDERS_DIR);
            if (!dir.exists()) {
                dir.mkdir();
            }
            String timestamp =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMM
dd_HHmmss"));
            String filename = ORDERS_DIR + "/order_" +
timestamp + ".txt";
```

```
           FileWriter writer = new FileWriter(filename);
           writer.write("--- Order ---\n");
           for (OrderItem item : order.getItems()) {
               writer.write(item.toString());
           }
           writer.write("-----------------\n");
           writer.write(String.format("Subtotal: $%.2f\n",
order.getSubtotal()));
           writer.write(String.format("Tax (10%%): $%.2f\n",
order.getTax()));
           writer.write(String.format("Total: $%.2f\n",
order.getTotal()));
           writer.close();

           System.out.println("Order saved to " + filename);
       } catch (IOException e) {
           System.out.println("Error saving order: " +
e.getMessage());
       }
   }
   public static void displayAllOrders() {
       File dir = new File(ORDERS_DIR);
       if (!dir.exists() || dir.listFiles() == null ||
dir.listFiles().length == 0) {
           System.out.println("No previous orders found.");
           return;
       }
       File[] orderFiles = dir.listFiles();
       for (File file : orderFiles) {
           System.out.println("\n--- Order from file: " +
file.getName() + " ---");
           try (Scanner reader = new Scanner(file)) {
               while (reader.hasNextLine()) {
                   System.out.println(reader.nextLine());
               }
           } catch (IOException e) {
               System.out.println("Error reading file: " +
file.getName());
           }
       }
   }
}
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
class Bill {
   public static void generateBill(Order order) {
       StringBuilder sb = new StringBuilder();
       sb.append("--- Pizza Shop Bill ---\n");
       for (OrderItem item : order.getItems()) {
           sb.append(item);
       }
       sb.append("-------------------------\n");
       sb.append(String.format("Subtotal: ৳%.2f\n",
order.getSubtotal()));
       sb.append(String.format("Tax (10%%): ৳%.2f\n",
order.getTax()));
       sb.append(String.format("Total: ৳%.2f\n",
order.getTotal()));
       System.out.println(sb);
       try {
           String filename = "bill_" +
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMM
dd_HHmmss")) + ".txt";
           FileWriter writer = new FileWriter(filename);
           writer.write(sb.toString());
           writer.close();
           System.out.println("Bill saved to " + filename);
       } catch (IOException e) {
           System.out.println("Error saving bill.");
       }
   }
}
import java.util.*;
public class PizzaShop {
   static Scanner scanner = new Scanner(System.in);
   static List<Topping> availableToppings = List.of(
           new Topping("Cheese", 100),
           new Topping("Pepperoni", 150),
           new Topping("Olives", 100),
           new Topping("Mushrooms", 120),
           new Topping("Onions", 80)
   );
   static List<String> sizes = List.of("Small", "Medium",
"Large");
   static List<Pizza> pizzas = List.of(
           new Pizza("Margherita", "", 500),
           new Pizza("Pepperoni", "", 600),
           new Pizza("Veggie", "", 550)
   );
   public static void main(String[] args) {
       System.out.println("Welcome to the Pizza Shop!");
       boolean running = true;
       while (running) {
           System.out.println("\nMain Menu:");
           System.out.println("1. Place Order");
           System.out.println("2. View Previous Orders");
           System.out.println("3. Exit");
           System.out.print("Choose an option: ");
           String choice = scanner.nextLine();
           switch (choice) {
               case "1" -> placeOrder();
               case "2" -> OrderStorage.displayAllOrders();
               case "3" -> {
                   running = false;
                   System.out.println("Thank you for visiting
Pizza Shop!");
               }
               default -> System.out.println("Invalid
choice.");
           }
       }
   }
   public static void placeOrder() {
       Order order = new Order();
       boolean ordering = true;

       while (ordering) {
           System.out.println("Available Pizzas:");
           for (int i = 0; i < pizzas.size(); i++) {
               System.out.println((i + 1) + ". " +
pizzas.get(i).getName() + " (৳" + pizzas.get(i).getBasePrice()
+ ")");
           }
           System.out.print("Select a pizza by number: ");
           int pizzaChoice = scanner.nextInt() - 1;
           Pizza basePizza = pizzas.get(pizzaChoice);
           System.out.println("Select size:");
           for (int i = 0; i < sizes.size(); i++) {
               System.out.println((i + 1) + ". " +
sizes.get(i));
```

```java
            }
            System.out.print("Enter size number: ");
            int sizeChoice = scanner.nextInt() - 1;
            String size = sizes.get(sizeChoice);
            Pizza selectedPizza = new
Pizza(basePizza.getName(), size, basePizza.getBasePrice());

            List<Topping> selectedToppings = new ArrayList<>();
            System.out.println("Available Toppings:");
            for (int i = 0; i < availableToppings.size(); i++)
{
                System.out.println((i + 1) + ". " +
availableToppings.get(i).getName() + " (৳" +
availableToppings.get(i).getPrice() + ")");
            }
            System.out.print("Enter topping numbers separated
by spaces (0 to skip): ");
            scanner.nextLine(); // consume newline
            String[] toppingChoices =
scanner.nextLine().split(" ");
            for (String choice : toppingChoices) {
                int toppingIndex = Integer.parseInt(choice) -1;
                if (toppingIndex >= 0 && toppingIndex <
availableToppings.size()) {
selectedToppings.add(availableToppings.get(toppingIndex));
                }
            }
            System.out.print("Enter quantity: ");
            int quantity = scanner.nextInt();
            order.addItem(new OrderItem(selectedPizza,
selectedToppings, quantity));
            System.out.print("Add another pizza? (y/n): ");
            scanner.nextLine();
            ordering =
scanner.nextLine().equalsIgnoreCase("y");
        }
        Bill.generateBill(order);
        OrderStorage.saveOrder(order);
    }
}
```

Summary

The CLI-based pizza ordering system works smoothly and demonstrates clear separation of concerns through its well-organized classes, offers user-friendly prompts for pizza selection, size, toppings, and quantity, calculates accurate bills including a 10% tax, and stores all bills as individual text files for easy access and record-keeping; the system is designed to be extensible for future improvements such as adding a graphical interface, customer loyalty features, or analytics.