Courier Management System

Coding Task 2

Loops and Iteration

---

5. Write a Java program that uses a for loop to display all the orders for a specific customer.

6. Implement a while loop to track the real-time location of a courier until it reaches its destination

---

Loops and iteration are fundamental programming constructs that allow repetitive execution of code blocks until a specified condition is met. In Java, loops such as for, while, and do-while help automate repetitive tasks efficiently. Using loops minimizes redundancy and enhances code readability. This document demonstrates the implementation of loops to manage orders and track courier locations dynamically.

Below are the tasks covered:

Task 2.1: Display All Orders for a Specific Customer

Task 2.2: Track Real-Time Location of a Courier

In this task, I implemented a Java program that utilizes loops to manage customer orders and track courier locations. The program provides options for users to display orders for a specific customer and track a courier's real-time location using loops.

Main Method (psvm)

The main method serves as the entry point of the program. It presents a menu to the user and uses a switch-case statement to call the respective methods based on user input.

```
package main;

import java.util.*;

public class Task2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nTask 2 - Choose an option:");
            System.out.println("1. Display Orders for a Specific Customer");
            System.out.println("2. Track Courier Location");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
```

```
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter customer name to view orders: ");
                    String customer = scanner.nextLine();
                    displayOrders(customer);
                    break;

                case 2:
                    System.out.println("\nStarting courier tracking...");
                    trackCourier();
                    break;
```

```
                case 3:
                    System.out.println("Exiting the program. Thank you!");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice. Please enter 1, 2, or 3.");
            }
    }
}
```

Task 2.1: Display All Orders for a Specific Customer

Write a Java program that uses a for loop to display all the orders for a specific customer.

This method uses a for loop to iterate through the list of orders and display them for a specific customer.

Code :

```
// 1. Display Orders for a Specific Customer (Using For Loop)
public static void displayOrders(String customer) {
    Map<String, List<String>> customerOrders = new HashMap<>();

    customerOrders.put("alice", Arrays.asList("Order 101", "Order 102", "Order 103"));
    customerOrders.put("bob", Arrays.asList("Order 201", "Order 202"));
    customerOrders.put("charlie", Arrays.asList("Order 301"));
```

```
    if (customerOrders.containsKey(customer.toLowerCase())) {
        System.out.println("Orders for " + customer + ":");
        List<String> orders = customerOrders.get(customer);

        for (int i = 0; i < orders.size(); i++) {
            System.out.println((i + 1) + ". " + orders.get(i));
        }
    } else {
        System.out.println("No orders found for " + customer + ".");
    }
}
```

Output :

```
Task 2 - Choose an option:
1. Display Orders for a Specific Customer
2. Track Courier Location
3. Exit
Enter your choice: 1
Enter customer name to view orders: alice
Orders for alice:
1. Order 101
2. Order 102
3. Order 103
```

```
Enter your choice: 1
Enter customer name to view orders: resh
No orders found for resh.
```

Task 2.2: Track Real-Time Location of a Courier

Implement a while loop to track the real-time location of a courier until it reaches its destination

This method uses a while loop to simulate a courier moving towards a destination. The courier moves a random distance in each iteration until reaching the destination.

Code :

```java
// 2. Track Courier Location Until Destination (Using While Loop)
public static void trackCourier() {
    int currentLocation = 0;
    int destination = 10;
    Random random = new Random();

    while (currentLocation < destination) {
        int step = random.nextInt( bound: 3) + 1;
        currentLocation += step;

        if (currentLocation > destination) {
            currentLocation = destination;
        }

        System.out.println("Courier moved to location: " + currentLocation);

        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    System.out.println("Courier has reached the destination!");
}
```

Output :

```
Task 2 - Choose an option:
1. Display Orders for a Specific Customer
2. Track Courier Location
3. Exit
Enter your choice: 2

Starting courier tracking...
Courier moved to location: 3
Courier moved to location: 5
Courier moved to location: 8
Courier moved to location: 10
Courier has reached the destination!
```

```
Starting courier tracking...
Courier moved to location: 1
Courier moved to location: 3
Courier moved to location: 4
Courier moved to location: 7
Courier moved to location: 9
Courier moved to location: 10
Courier has reached the destination!
```

Conclusion

In this task, I implemented:

1. A for loop to display orders for a specific customer.
2. A while loop to track a courier's real-time movement.

These implementations demonstrate the effective use of loops for handling iterative processes in Java.
