Courier Management System

Coding Task 1

Control Flow Statements

---

1. Write a program that checks whether a given order is delivered or not based on its status (e.g**., "Processing," "Delivered," "Cancelled").** Use if-else statements for this.

2. Implement a **switch-case statement** to categorize parcels based on their weight into **"Light," "Medium," or "Heavy."**

3. Implement User Authentication **1.** Create a login system for employees and customers using Java **control flow statements.**

4. Implement Courier Assignment Logic **1.** Develop a mechanism to assign couriers to shipments based on predefined criteria **(e.g., proximity, load capacity)** using loops.

Now that the SQL part is completed, we are stepping into the coding part.

The first task focuses on implementing basic control flow statements in Java. These include conditional statements (if-else and switch-case) and loops to handle courier-related operations.

Below are the tasks covered:

Task 1.1: Check Order Status

Task 1.2: Categorize Parcels by Weight

Task 1.3: Implement User and Employee Authentication

Task 1.4: Courier Assignment Logic

Task 1.1: Check Order Status

Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.

This task involves retrieving the status of an order based on the courier ID. The method will prompt the user for the courier ID, retrieve its status from the courier table in the database, and display it using an if-else statement.

The control flow will handle different statuses like "Processing," "Delivered," and "Cancelled."

Code :

```
1   package main;
2
3   import util.DBConnUtil;
4   import java.sql.*;
5   import java.util.Scanner;
```

```java
public class OrderStatusChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Courier ID: ");
        int courierId = scanner.nextInt();
        checkOrderStatus(courierId);
        scanner.close();
    }
```

```java
public static void checkOrderStatus(int courierId) {
    String query = "SELECT Status FROM courier WHERE CourierID = ?";

    try (Connection conn = DBConnUtil.getConnection();
         PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setInt( parameterIndex: 1, courierId);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            String status = rs.getString( columnLabel: "Status");

            // Display order status
            if (status.equalsIgnoreCase( anotherString: "Delivered")) {
                System.out.println("Order has been DELIVERED ✅.");
            } else if (status.equalsIgnoreCase( anotherString: "Processing")) {
                System.out.println("Order is still in PROCESS ⏳.");
            } else if (status.equalsIgnoreCase( anotherString: "Cancelled")) {
                System.out.println("Order has been CANCELLED ❌.");
            } else {
                System.out.println("Order status: " + status);
            }
        } else {
            System.out.println("⚠ No order found with Courier ID: " + courierId);
        }
```

```java
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Output :

```
Enter Courier ID: 2
Order has been DELIVERED ✅.
```

```
Enter Courier ID: 42
Order is Scheduled ⏳.
```

```
Enter Courier ID: 10
Order status: Pending
```

```
Enter Courier ID: 1
Order status: In Transit
```

```
Enter Courier ID: 5
Order status: Shipped
```

```
Enter Courier ID: 89
⚠ No order found with Courier ID: 89
```

Work Flow :

- The user enters a courier ID.
- The program queries the database for the status of that courier.
- The retrieved status is checked using an if-else statement.
- The corresponding message is displayed.

Task 1.2: Categorize Parcels by Weight

Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

This task retrieves the weight of a parcel using the courier ID and categorizes it based on predefined weight ranges. It then displays the weight category using a switch statement.

Code :

```
package main;

import util.DBConnUtil;
import java.sql.*;
import java.util.Scanner;

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

        System.out.print("Enter Courier ID: ");
        int courierId = scanner.nextInt();

        categorizeParcelWeight(courierId);

        scanner.close();
    }
```

```
public static void categorizeParcelWeight(int courierId) {
    String query = "SELECT Weight FROM courier WHERE CourierID = ?";

    try (Connection conn = DBConnUtil.getConnection();
         PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setInt( parameterIndex: 1, courierId);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            double weight = rs.getDouble( columnLabel: "Weight");
            System.out.println("Parcel Weight: " + weight + " kg");

            String category;
            int categoryIndex;

            if (weight <= 2.0) {
                categoryIndex = 1;
            } else if (weight <= 5.0) {
                categoryIndex = 2;
            } else {
                categoryIndex = 3;
            }
```

```
            switch (categoryIndex) {
                case 1:
                    category = "Light";
                    break;
                case 2:
                    category = "Medium";
                    break;
                case 3:
                    category = "Heavy";
                    break;
                default:
                    category = "Unknown";
            }

            System.out.println("Category: " + category);
        } else {
            System.out.println("⚠ No parcel found with Courier ID: " + courie
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    }
}
```

Output :

```
Enter Courier ID: 2
Parcel Weight: 1.2 kg
Category: Light
```

```
Enter Courier ID: 10
Parcel Weight: 5.0 kg
Category: Medium
```

```
Enter Courier ID: 61
Parcel Weight: 12.5 kg
Category: Heavy
```

```
Enter Courier ID: 100
⚠ No parcel found with Courier ID: 100
```

Work Flow :

- The user enters a courier ID.
- The program retrieves the parcel weight from the database.
- A switch statement categorizes the weight into "Light," "Medium," or "Heavy."
- The result is displayed to the user.

Task 1.3: Implement User and Employee Authentication

Create a login system for employees and customers using Java control flow statements.

This task involves implementing a login system where employees and customers enter their credentials. The program verifies the credentials against the database and grants or denies access accordingly.

Code:

UserAuthentication :

```java
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class UserAuthentication {
    public static String authenticate(String email, String password) {
        String username = null;
        String query = "SELECT Name FROM user WHERE Email = ? AND Password = ?";

        try (Connection conn = DBConnUtil.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString( parameterIndex: 1, email);
            stmt.setString( parameterIndex: 2, password);
            ResultSet resultSet = stmt.executeQuery();

            if (resultSet.next()) {
                username = resultSet.getString( columnLabel: "Name");  // Fetch the username
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return username;  // Returns null if authentication fails
    }
```

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

        // Get email and password from user
        System.out.print("Enter Email: ");
        String email = scanner.nextLine();

        System.out.print("Enter Password: ");
        String password = scanner.nextLine();

        // Authenticate and get username
        String username = authenticate(email, password);

        if (username != null) {
            System.out.println("Login successful! Welcome, " + username + " 👋");
        } else {
            System.out.println("Invalid credentials. Please try again.");
        }

        scanner.close();
    }
}
```

EmployeeAuthentication

```java
package main;

import util.DBConnUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class EmployeeAuthentication {
    public static String authenticateEmployee(String email, String password) {
        String employeeName = null;
        String query = "SELECT Name FROM employee WHERE Email = ? AND Password = ?";

        try (Connection conn = DBConnUtil.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString( parameterIndex: 1, email);
            stmt.setString( parameterIndex: 2, password);
            ResultSet resultSet = stmt.executeQuery();

            if (resultSet.next()) {
                employeeName = resultSet.getString( columnLabel: "Name");  // Fetch the employee name
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return employeeName;  // Returns null if authentication fails
    }
```

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

        // Get employee email and password
        System.out.print("Enter Employee Email: ");
        String email = scanner.nextLine();

        System.out.print("Enter Password: ");
        String password = scanner.nextLine();

        // Authenticate and get employee name
        String employeeName = authenticateEmployee(email, password);

        if (employeeName != null) {
            System.out.println("Login successful! Welcome, " + employeeName + " 🎉");
        } else {
            System.out.println("Invalid credentials. Please try again.");
        }

        scanner.close();
    }
```

Output :

```
Enter Employee Email: noah@example.com
Enter Password: Noah White@3
Login successful! Welcome, Noah White 🎉
```

```
Enter Email: emma@example.com
Enter Password: emmapass
Login successful! Welcome, Emma Wilson 👋
```

Work Flow :

- The user enters an email and password.
- The program checks the credentials against the database.
- If the credentials are correct, access is granted; otherwise, access is denied.

Task 1.4: Courier Assignment Logic

Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., proximity, load capacity) using loops.

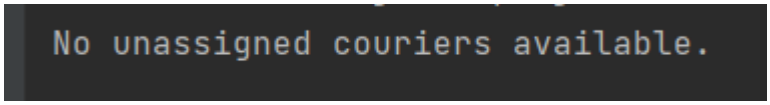
This task involves retrieving unassigned couriers, finding available employees, and assigning the couriers based on predefined criteria like proximity and load capacity.

Code :

```
package main;

import entity.Courier;
import entity.Employee;
import util.DBConnUtil;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class CourierAssignmentService {

    private List<Courier> getUnassignedCouriers() {
        List<Courier> couriers = new ArrayList<>();
        String query = "SELECT * FROM courier WHERE EmployeeID IS NULL";

        try (Connection conn = DBConnUtil.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            while (rs.next()) {
                couriers.add(new Courier(
                        rs.getInt( columnLabel: "CourierID"),
                        rs.getString( columnLabel: "SenderName"),
                        rs.getString( columnLabel: "SenderAddress"),
                        rs.getString( columnLabel: "ReceiverName"),
                        rs.getString( columnLabel: "ReceiverAddress"),
                        rs.getDouble( columnLabel: "Weight"),
                        rs.getString( columnLabel: "Status"),
                        rs.getString( columnLabel: "TrackingNumber"),
                        rs.getString( columnLabel: "DeliveryDate"),
                        rs.getInt( columnLabel: "SenderID"),
                        rs.getInt( columnLabel: "ServiceID")
                ));
            }
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return couriers;
    }

    private List<Employee> getAvailableEmployees() {
        List<Employee> employees = new ArrayList<>();
        String query = "SELECT * FROM employee";

        try (Connection conn = DBConnUtil.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            while (rs.next()) {
                employees.add(new Employee(
                        rs.getInt( columnLabel: "EmployeeID"),
                        rs.getString( columnLabel: "Name"),
                        rs.getString( columnLabel: "Email"),
                        rs.getString( columnLabel: "ContactNumber"),
                        rs.getString( columnLabel: "Role"),
                        rs.getDouble( columnLabel: "Salary")
                ));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return employees;
    }

    public void assignCouriers() {
        List<Courier> unassignedCouriers = getUnassignedCouriers();
        List<Employee> employees = getAvailableEmployees();

        if (unassignedCouriers.isEmpty() || employees.isEmpty()) {
            System.out.println("No couriers to assign or no employees available.");
            return;
        }

        int empIndex = 0;
        for (Courier courier : unassignedCouriers) {
            Employee assignedEmployee = employees.get(empIndex);
            assignCourierToEmployee(courier.getCourierID(), assignedEmployee.getEmployeeID());
            empIndex = (empIndex + 1) % employees.size();
        }
        System.out.println("All unassigned couriers have been assigned to employees.");
    }

    private void assignCourierToEmployee(int courierID, int employeeID) {
        String query = "UPDATE courier SET EmployeeID = ? WHERE CourierID = ?";

        try (Connection conn = DBConnUtil.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setInt( parameterIndex: 1, employeeID);
            stmt.setInt( parameterIndex: 2, courierID);
            stmt.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Output:



Work Flow :

- Unassigned couriers and available employees are retrieved from the database.
- Couriers are assigned to employees using a loop.
- The database is updated accordingly.

Conclusion

In this task, I implemented various control flow statements in Java to manage courier-related operations efficiently. The tasks covered essential programming constructs such as if-else statements, switch-case, loops, and user authentication mechanisms.

1. **Order Status Check** – We retrieved the courier status from the database and displayed it using an if-else structure.
2. **Parcel Weight Categorization** – Implemented a switch-case statement to classify parcels into different weight categories.
3. **User Authentication** – Designed a login system for customers and employees using control flow statements.
4. **Courier Assignment Logic** – Assigned couriers to available employees using database queries and loops.

These implementations showcase how Java control flow statements can be used effectively in real-world scenarios, ensuring structured and logical execution of operations. By integrating these control mechanisms, we improve the efficiency of courier tracking, parcel management, and user authentication. Moving forward, we can enhance these modules by incorporating exception handling, logging mechanisms, and optimizing database queries for better performance.
