Rajalakshmi Engineering College

Name: HARINI G
Email: 241001074@rajalakshmi.edu.in
Roll no: 2116241001074
Phone: 6383888158
Branch: REC
Department: IT - Section 1
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 11

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : Project

1. Problem Statement

In Café Central, the menu is cataloged and stored in a database.

To efficiently manage the restaurant's menu using Java and JDBC, you must build a Restaurant Management System that supports:

Adding new menu items

Updating menu item prices

Viewing details of a menu item

Displaying all menu items in sorted order

You are given two files:

File 1: MenuItem.java (POJO Class)

This class represents the MenuItem entity.

A MenuItem contains the following details:

Field Description

itemId Unique Menu Item ID (Integer)

name Item Name (String)

category Item Category (String)

price Item Price (Double)

Students must write code in the marked area:

```
class MenuItem {
    private int itemId;
    private String name;
    private String category;
    private double price;

    public MenuItem() {}

    public MenuItem(int itemId, String name, String category, double price) {
        // write your code here
    }

    // Include getters and setters
}
```

Expected in this part:

Assign parameter values to instance variables inside the constructor.

Add getters and setters for all attributes.

File 2: MenuItemDAO.java (Data Access Layer)

This class handles all database operations using JDBC.

Students must complete the missing JDBC logic in the following methods:

```
class MenuItemDAO {

    public void addMenuItem(Connection conn, MenuItem menuItem)
throws SQLException {
        // write your code here
    }

    public void updateItemPrice(Connection conn, int itemId, double
newPrice) throws SQLException {
        // write your code here
    }

    public void deleteMenuItem(Connection conn, int itemId) throws
SQLException {
        // write your code here
    }

    public MenuItem viewItemDetails(Connection conn, int itemId) throws
SQLException {
        // write your code here
    }

    public List<MenuItem> displayAllMenuItems(Connection conn) throws
SQLException {
        // write your code here
    }

    private MenuItem mapToMenuItem(ResultSet rs) throws SQLException {
        return new MenuItem(
```

```
        // write your code here
    );
  }
}
```

Expected in this part:

Write SQL queries for INSERT, UPDATE, DELETE, SELECT.

Execute queries using PreparedStatement or Statement.

Map ResultSet rows to MenuItem objects using mapToMenuItem().

Return a List<MenuItem> where required.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri_db

USER: test

PWD: test123

The menu table has already been created with the following structure:

Table Name: menu

Input Format

The first line of input consists of an integer choice, representing the operation to be performed (1 for Add Item, 2 for Restock item, 3 for reduce item, 4 for Display, 5 for Exit).

For choice 1 (Add Menu Item):

- The second line consists of an integer item_id.
- The third line consists of a string name.
- The fourth line consists of a string category.
- The fifth line consists of a double price.

For choice 2 (Update Item Price):

- The second line consists of an integer item_id.
- The third line consists of a double new_price.

For choice 3 (View Item Details):

- The second line consists of an integer item_id.

For choice 4 (Display All Menu Items):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

Output Format

For choice 1 (Add Menu Item):

- Print "Menu item added successfully" if the item was added.
- Print "Failed to add item." if the insertion failed.

For choice 2 (Update Item Price):

- Print "Item price updated successfully" if the price update was successful.
- Print "Item not found." if the specified item ID does not exist.

For choice 3 (View Item Details):

- Display the item details in the format:
- ID: [item_id] | Name: [name] | Category: [category] | Price: [price]
- Print "Item not found." if the specified item ID does not exist.

For choice 4 (Display All Menu Items):

- Display each item on a new line in the format:
- ID | Name | Category | Price
- If no items are available, print nothing (or handle with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Restaurant Management System."

For invalid input:

- Print "Invalid choice. Please try again."

Sample Test Case

Input: 1
11
Margherita Pizza
Main Course
12.99
4
5
Output: Menu item added successfully
ID | Name | Category | Price
11 | Margherita Pizza | Main Course | 12.99
Exiting Restaurant Management System.

Answer

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

class RestaurantManagementSystem {
   public static void main(String[] args) {
      try (Connection conn = DriverManager.getConnection("jdbc:mysql://
localhost/ri_db", "test", "test123");
           Scanner scanner = new Scanner(System.in)) {

          boolean running = true;

          while (running) {
             int choice = scanner.nextInt();

             switch (choice) {
                case 1:
                   addMenuItem(conn, scanner);
                   break;
                case 2:
                   updateItemPrice(conn, scanner);
                   break;
```

```
                case 3:
                    viewItemDetails(conn, scanner);
                    break;
                case 4:
                    displayAllMenuItems(conn);
                    break;
                case 5:
                    System.out.println("Exiting Restaurant Management System.");
                    running = false;
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

// You are using Java
    public static void addMenuItem(Connection conn, Scanner scanner) {
        try{
            int id=scanner.nextInt();
            scanner.nextLine();
            String name=scanner.nextLine();
            String category=scanner.nextLine();
            double price=scanner.nextDouble();
            String sql="INSERT INTO
menu(item_id,name,category,price)VALUES(?,?,?,?)";
            try(PreparedStatement ps = conn.prepareStatement(sql)){
                ps.setInt(1,id);
                ps.setString(2,name);
                ps.setString(3,category);
                ps.setDouble(4,price);

                int rows=ps.executeUpdate();
                if(rows>0){
                    System.out.println("Menu item added successfully");
                }else{
                    System.out.println("Failed to add item.");
                }
            }
        } catch(SQLException e){
```

```
            System.out.println("Failed to add item.");
        }
    }

    public static void updateItemPrice(Connection conn, Scanner scanner) {
        try{
            int id=scanner.nextInt();
            double newPrice = scanner.nextDouble();

            String checkSql = "SELECT * FROM menu WHERE item_id=?";
            try(PreparedStatement checkStmt = conn.prepareStatement(checkSql)){
                checkStmt.setInt(1,id);
                ResultSet rs = checkStmt.executeQuery();

                if(rs.next()){
                    String updateSql = "UPDATE menu SET price=? WHERE item_id=?";
                    try(PreparedStatement updateStmt =
conn.prepareStatement(updateSql)){
                        updateStmt.setDouble(1,newPrice);
                        updateStmt.setInt(2,id);
                        updateStmt.executeUpdate();
                        System.out.println("Item price updated successfully");
                    }
                }else{
                    System.out.println("Item not found.");
                }
            }
        }catch (SQLException e){
            System.out.println("Item not found");
        }
    }

    public static void viewItemDetails(Connection conn, Scanner scanner) {
        try{
            int id=scanner.nextInt();
            String sql="SELECT * FROM menu WHERE item_id=?";
            try(PreparedStatement ps = conn.prepareStatement(sql)){
                ps.setInt(1,id);
                ResultSet rs= ps.executeQuery();

                if(rs.next()){
                    int itemId=rs.getInt("item_id");
```

```
                String name=rs.getString("name");
                String category = rs.getString("category");
                double price = rs.getDouble("price");

                System.out.printf("ID: %d | Name: %s | Category: %s | Price: %.2f
%n",itemId,name,category,price);
            }
            else{
                System.out.println("Item not found.");
            }
        }
    }catch(SQLException e){
        System.out.println("Item not found.");
    }
}

  public static void displayAllMenuItems(Connection conn) {
    String sql = "SELECT * FROM menu ORDER BY item_id";
    try(Statement stmt=conn.createStatement();
    ResultSet rs= stmt.executeQuery(sql)){
        boolean hasData = false;
        System.out.println("ID | Name | Category|Price");
        while(rs.next()){
            hasData=true;
            int id= rs.getInt("item_id");
            String name=rs.getString("name");
            String category=rs.getString("category");
            double price=rs.getDouble("price");
            System.out.printf("%d | %s | %s | %.2f%n", id,name,category,price);
        }
        if(!hasData){}
    }catch(SQLException e){
        e.printStackTrace();
    }
  }
}


 class MenuItem {
   private int itemId;
   private String name;
   private String category;
```

```
    private double price;

    // Constructor
    public MenuItem(int itemId, String name, String category, double price) {
        this.itemId = itemId;
        this.name = name;
        this.category = category;
        this.price = price;
    }

   //Include getters and setters
}
//
```

Status : Correct **Marks : 10/10**

2. Problem Statement

Create a JDBC-based Hospital Management System that handles runtime input to manage patient records. The system should allow users to:

Add a new patient (patient ID, name, age, status).

Update a patient's status.

View a specific patient's record by patient ID.

Display all patient records in the database.

Exit the application.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri_db

USER: test

PWD: test123

The patients table has already been created with the following structure:

Table Name: patients

Input Format

The first line of input consists of an integer choice, representing the operation to be performed:

(1 for Add Patient, 2 for Update Patient Status, 3 for View Patient Record, 4 for Display All Patients, 5 for Exit)

For choice 1 (Add Patient):

- The second line consists of an integer patient_id.
- The third line consists of a string name.
- The fourth line consists of an integer age.
- The fifth line consists of a string status.

For choice 2 (Update Patient Status):

- The second line consists of an integer patient_id.
- The third line consists of a string new_status.

For choice 3 (View Patient Record):

- The second line consists of an integer patient_id.

For choice 4 (Display All Patients):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

Output Format

For choice 1 (Add Patient):

- Print "Patient added successfully" if the patient was added.
- Print "Failed to add patient." if the insertion failed.

For choice 2 (Update Patient Status):

- Print "Patient status updated successfully" if the update was successful.
- Print "Patient not found." if the specified patient ID does not exist.

For choice 3 (View Patient Record):

- Display the patient details in the format:
- ID: [patient_id] | Name: [name] | Age: [age] | Status: [status]
- Print "Patient not found." if the specified patient ID does not exist.

For choice 4 (Display All Patients):

- Display each patient on a new line in the format:
- ID | Name | Age | Status
- If no records are available, print nothing (or handle it with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Hospital Management System."

For invalid input:

- Print "Invalid choice. Please try again."

Sample Test Case

Input: 1
101
John Doe
45
Admitted
4
5
Output: Patient added successfully
ID | Name | Age | Status
101 | John Doe | 45 | Admitted
Exiting Hospital Management System.

Answer

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

```
class HospitalManagementSystem {
    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://
localhost/ri_db", "test", "test123");
             Scanner scanner = new Scanner(System.in)) {

            boolean running = true;

            while (running) {

                int choice = scanner.nextInt();

                switch (choice) {
                    case 1:
                        addPatient(conn, scanner);
                        break;
                    case 2:
                        updatePatientStatus(conn, scanner);
                        break;
                    case 3:
                        viewPatientRecord(conn, scanner);
                        break;
                    case 4:
                        displayAllPatients(conn);
                        break;
                    case 5:
                        System.out.println("Exiting Hospital Management System.");
                        running = false;
                        break;
                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

// You are using Java
    public static void addPatient(Connection conn, Scanner scanner) {
        try{
            int patientId=scanner.nextInt();
```

```
        scanner.nextLine();
        String name=scanner.nextLine();
        int age=scanner.nextInt();
        scanner.nextLine();
        String status=scanner.nextLine();

        String sql="Insert INTO
patients(patient_id,name,age,status)VALUES(?,?,?,?)";
        try(PreparedStatement pstmt = conn.prepareStatement(sql)){
            pstmt.setInt(1,patientId);
            pstmt.setString(2,name);
            pstmt.setInt(3,age);
            pstmt.setString(4,status);
            pstmt.executeUpdate();
            System.out.println("patient added successfully");
        }
    }
    catch(SQLException e){
        System.out.println("Failed to add patient.");
    }
  }

  public static void updatePatientStatus(Connection conn, Scanner scanner) {
    try{
        int patientId=scanner.nextInt();
        scanner.nextLine();
        String newStatus = scanner.nextLine();
        String checkSql= "SELECT * FROM patients WHERE patient_id=?";
        try(PreparedStatement checkStmt=conn.prepareStatement(checkSql)){
            checkStmt.setInt(1,patientId);
              try(ResultSet rs= checkStmt.executeQuery()){
                 if(!rs.next()){
                    System.out.println("Patient not found.");
                    return;
                 }
              }
            }
            String updateSql = "Update patients SET status = ? WHERE
patient_id=?";
            try(PreparedStatement pstmt=conn.prepareStatement(updateSql)){
              pstmt.setString(1,newStatus);
              pstmt.setInt(2,patientId);
```

```
                pstmt.executeUpdate();
                System.out.println("Patient status updated successfully");
            }

        }catch(SQLException e){
            System.out.println("Patient not found.");
        }
    }

    public static void viewPatientRecord(Connection conn, Scanner scanner) {
        try{
            int patientId=scanner.nextInt();
            String sql="SELECT * FROM patients WHERE patient_id=?";
            try(PreparedStatement pstmt=conn.prepareStatement(sql)){
                pstmt.setInt(1,patientId);
                try(ResultSet rs=pstmt.executeQuery()){
                    if(rs.next()){
                        System.out.printf("ID: %d| Name: %s| Age: %d| Status: %s%n",rs.getI
nt("patient_id"),rs.getString("name"),rs.getInt("age"),rs.getString("status"));
                    }
                    else{
                        System.out.println("patient not found.");
                    }
                }
            }
        }catch(SQLException e){
            System.out.println("patient not found.");
        }
    }

    public static void displayAllPatients(Connection conn) {
        try{
            String sql = "SELECT * FROM patients";
            try(Statement stmt=conn.createStatement(); ResultSet rs=
stmt.executeQuery(sql)){
                System.out.println("ID| Name | Age | status");
                while(rs.next()){
                    System.out.printf("%d | %s | %d | %s%n",
rs.getInt("patient_id"),rs.getString("name"),rs.getInt("age"),rs.getString("status"));
                }
            }
        }catch(SQLException e){
```

```
            System.out.println("Error displaying patients: "+e.getMessage());
        }

    }

}
```

Status : Correct **Marks : 10/10**