Mini-Project 01 : Product Management System (JDBC)

Code:

Product.java-

```
package com.product.model;
public class Product
{
   private int id;
   private String name;
   private String category;
   private double price;
   public Product(int id, String name, String category, double price)
   {
      this.id = id;
      this.name = name;
      this.category = category;
      this.price = price;
   }
   public int getId()
   {
      return id;
   }
   public void setId(int id)
   {
      this.id = id;
   }
   public String getName()
   {
      return name;
   }
   public void setName(String name)
   {
      this.name = name;
   }
   public String getCategory()
   {
      return category;
   }
   public void setCategory(String category)
   {
      this.category = category;
   }
   public double getPrice()
   {
      return price;
   }
   public void setPrice(double price)
   {
      this.price = price;
```

```
    }
    @Override
    public String toString()
    {
        return "Product [id=" + id + ", name=" + name + ", category=" + category + ", price=" +
price + "]";
    }
}
```

ProductDao.java-

```
package com.product.dao;
import java.sql.SQLException;
import java.util.List;
import com.product.model.Product;
public interface ProductDao
{
    public void createProductTable() throws SQLException;
    public void addProduct(Product p) throws SQLException;
    public List<Product> showAllProducts() throws SQLException;
    public List<Product> showProductsSortedByName() throws SQLException;
    public void connect() throws SQLException;
    public void updateProduct(Product p) throws SQLException;
    public void deleteProduct(int id) throws SQLException;
    public Product findProductByIdUsingProc(int id) throws SQLException;
    public double findMostExpensiveProductUsingFunc() throws SQLException;
    public String findMostExpensiveProductNameUsingFunc() throws SQLException;
    public double findCheapestProductUsingFunc() throws SQLException;
    public String findCheapestProductNameUsingFunc() throws SQLException;
}
```

ProductDaoImpl.java-

```
package com.product.dao;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import com.product.model.Product;
public class ProductDaoImpl implements ProductDao
{
    private Connection con;
    @Override
    public void createProductTable() throws SQLException
    {
        Statement st = con.createStatement();
```

```
        String sql = "CREATE TABLE IF NOT EXISTS PRODUCTS (id INT PRIMARY KEY,
name VARCHAR(50), category VARCHAR(50), price DOUBLE)";
        st.execute(sql);
        System.out.println("Product table created!");
        st.close();
    }

    @Override
    public void addProduct(Product p) throws SQLException
    {
        PreparedStatement pst = con.prepareStatement("INSERT INTO PRODUCTS VALUES
(?,?,?,?)");
        pst.setInt(1, p.getId());
        pst.setString(2, p.getName());
        pst.setString(3, p.getCategory());
        pst.setDouble(4, p.getPrice());
        int count = pst.executeUpdate();
        System.out.println(count + " record added to products table successfully!");
        pst.close();
    }
    @Override
    public List<Product> showAllProducts() throws SQLException
    {
        Statement st = con.createStatement();
        List<Product> products = new ArrayList<>();
        ResultSet rs = st.executeQuery("SELECT * FROM PRODUCTS");
        while (rs.next())
        {
            products.add(new Product(rs.getInt(1), rs.getString(2), rs.getString(3),
rs.getDouble(4)));
        }
        rs.close();
        st.close();
        return products;
    }
    @Override
    public List<Product> showProductsSortedByName() throws SQLException
    {
        Statement st = con.createStatement();
        List<Product> products = new ArrayList<>();
        ResultSet rs = st.executeQuery("SELECT * FROM PRODUCTS ORDER BY name");
        while (rs.next())
        {
            products.add(new Product(rs.getInt(1), rs.getString(2), rs.getString(3),
rs.getDouble(4)));
        }
        rs.close();
        st.close();
        return products;
    }
```

```
    @Override
    public void connect() throws SQLException
    {
        String url = "jdbc:mysql://localhost:3306/productdb";
        String user = "root";
        String pwd = "Mysql@369#pass";
        con = DriverManager.getConnection(url, user, pwd);
        System.out.println("Connected to the database!");
    }
    @Override
    public void updateProduct(Product p) throws SQLException
    {
        PreparedStatement pst = con.prepareStatement("UPDATE PRODUCTS SET name = ?,
category = ?, price = ? WHERE id = ?");
        pst.setString(1, p.getName());
        pst.setString(2, p.getCategory());
        pst.setDouble(3, p.getPrice());
        pst.setInt(4, p.getId());
        int cnt = pst.executeUpdate();
        System.out.println(cnt + " record updated successfully!");
        pst.close();
    }
    @Override
    public void deleteProduct(int id) throws SQLException
    {
        PreparedStatement pst = con.prepareStatement("DELETE FROM PRODUCTS WHERE
id = ?");
        pst.setInt(1, id);
        int cnt = pst.executeUpdate();
        System.out.println(cnt + " record deleted successfully!");
        pst.close();
    }
    @Override
    public Product findProductByIdUsingProc(int id) throws SQLException
    {
        CallableStatement cs = con.prepareCall("{CALL findProductById(?)}");
        cs.setInt(1, id);
        ResultSet rs = cs.executeQuery();
        Product product = null;
        if (rs.next())
        {
            product = new Product(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getDouble(4));
        }
        rs.close();
        cs.close();
        return product;
    }
    @Override
    public double findMostExpensiveProductUsingFunc() throws SQLException
    {
```

```
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT findMostExpensiveProduct()");
        rs.next();
        double price = rs.getDouble(1);
        rs.close();
        st.close();
        return price;
    }
    @Override
    public String findMostExpensiveProductNameUsingFunc() throws SQLException
    {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT findMostExpensiveProductName()");
        rs.next();
        String name = rs.getString(1);
        rs.close();
        st.close();
        return name;
    }
    @Override
    public double findCheapestProductUsingFunc() throws SQLException
    {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT findCheapestProduct()");
        rs.next();
        double price = rs.getDouble(1);
        rs.close();
        st.close();
        return price;
    }
    @Override
    public String findCheapestProductNameUsingFunc() throws SQLException
    {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT findCheapestProductName()");
        rs.next();
        String name = rs.getString(1);
        rs.close();
        st.close();
        return name;
    }

}
```

Authenticator.java-

```
package com.product.service;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class Authenticator
```

```
{
    private static Map<String, String> userStore = new HashMap<>();

    public static void register()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter new username: ");
        String username = sc.nextLine();
        System.out.print("Enter new password: ");
        String password = sc.nextLine();

        if (userStore.containsKey(username))
        {
            System.out.println("Username already exists. Please try again.");
        } else
        {
            userStore.put(username, password);
            System.out.println("User registered successfully!");
        }
    }

    public static boolean authenticate()
    {
        Scanner sc = new Scanner(System.in);

        try
        {
            System.out.print("Enter username: ");
            String username = sc.nextLine();
            System.out.print("Enter password: ");
            String password = sc.nextLine();

            if (userStore.containsKey(username) && userStore.get(username).equals(password))
            {
                return true;
            }
            else
            {
                System.out.println("Authentication failed. Invalid username or password.");
                return false;
            }
        }
        catch (Exception e)
        {
            System.out.println("An error occurred during authentication: " + e.getMessage());
            return false;
        }
    }
}
```

ProductService.java-

```
package com.product.service;
import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;
import com.product.dao.ProductDaoImpl;
import com.product.model.Product;
public class ProductService
{
        public static void main(String[] args) throws SQLException {
      Scanner sc = new Scanner(System.in);
                boolean authenticated = false;
                while(!authenticated)
                {
                        System.out.println("-----------------------------------------------");
                        System.out.println("User Authentication");
                        System.out.println("1. Register");
      System.out.println("2. Login");
      System.out.print("Enter your choice: ");
      int choice = sc.nextInt();
      sc.nextLine();
      System.out.println("-----------------------------------------------");
      switch (choice)
      {
      case 1:
         Authenticator.register();
         break;
      case 2:
         authenticated = Authenticator.authenticate();
         break;
      default:
         System.out.println("Invalid choice! Please try again.");
         break;
      }
                }

      ProductDaoImpl dao = new ProductDaoImpl();
      int choice, productId;
      String productName, productCategory;
      double productPrice;

      do
      {
         System.out.println("-----------------------------------------------");
         System.out.println("Menu");
         System.out.println("1. Create Product Table");
         System.out.println("2. Add New Product");
         System.out.println("3. Show All Products");
         System.out.println("4. Show Products Sorted by Name");
         System.out.println("5. Update Product");
```

```
            System.out.println("6. Delete Product");
            System.out.println("7. Find Product By ID");
            System.out.println("8. Find Most Expensive Product Name and Price");
            System.out.println("9. Find Cheapest Product Name and Price");
            System.out.println("10. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();
            System.out.println("-----------------------------------------------");
            switch (choice) {
               case 1:
                  dao.connect();
                  dao.createProductTable();
                  break;

               case 2:
                  dao.connect();
                  System.out.print("Enter Product ID: ");
                  productId = sc.nextInt();
                  sc.nextLine();
                  System.out.print("Enter Product Name: ");
                  productName = sc.nextLine();
                  System.out.print("Enter Product Category: ");
                  productCategory = sc.nextLine();
                  System.out.print("Enter Product Price: ");
                  productPrice = sc.nextDouble();
                  sc.nextLine();
                  Product product = new Product(productId, productName, productCategory,
productPrice);
                  dao.addProduct(product);
                  break;

               case 3:
                  dao.connect();
                  List<Product> products = dao.showAllProducts();
                  if(products.isEmpty())
                  {
                     System.out.println("No entries found in the product table");
                  }
                  else
                  {
                     for (Product p : products)
                     {
                        System.out.println(p);
                     }
                  }
                  break;

               case 4:
                  dao.connect();
```

```
            List<Product> sortedProducts = dao.showProductsSortedByName();
            if(sortedProducts.isEmpty())
            {
               System.out.println("No entries found in the product table");
            }
            else
            {
               for (Product p : sortedProducts)
               {
                  System.out.println(p);
               }
            }
            break;

         case 5:
            dao.connect();
            System.out.print("Enter Product ID to update: ");
            int updateProductId = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter New Product Name: ");
            String newProductName = sc.nextLine();
            System.out.print("Enter New Product Category: ");
            String newProductCategory = sc.nextLine();
            System.out.print("Enter New Product Price: ");
            double newProductPrice = sc.nextDouble();
            sc.nextLine();
            Product updatedProduct = new Product(updateProductId, newProductName,
newProductCategory, newProductPrice);
            dao.updateProduct(updatedProduct);
            break;

         case 6:
            dao.connect();
            System.out.print("Enter Product ID to delete: ");
            int deleteProductId = sc.nextInt();
            sc.nextLine();
            dao.deleteProduct(deleteProductId);
            break;

         case 7:
            dao.connect();
            System.out.print("Enter Product ID to find: ");
            int findProductId = sc.nextInt();
            sc.nextLine();
            Product foundProduct = dao.findProductByIdUsingProc(findProductId);
            System.out.println(foundProduct != null ? foundProduct : "Product not found");
            break;

         case 8:
            dao.connect();
```

```
                double mostExpensivePrice = dao.findMostExpensiveProductUsingFunc();
                String mostExpensiveProductName =
dao.findMostExpensiveProductNameUsingFunc();
                System.out.println("Most Expensive Product is " + mostExpensiveProductName
+ " and its price is Rs." + mostExpensivePrice);
                break;

            case 9:
                dao.connect();
                String cheapestProductName = dao.findCheapestProductNameUsingFunc();
                double cheapestPrice = dao.findCheapestProductUsingFunc();
                System.out.println("Cheapest Product is " + cheapestProductName + " and its
price is Rs." + cheapestPrice);
                break;

            case 10:
                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice! Please try again.");
                break;
        }
    }
    while (choice != 10);

    sc.close();
  }
}
```

sql-

```
SHOW DATABASES;
CREATE DATABASE productdb;
USE productdb;
SELECT * FROM PRODUCTS;
```

StoredProcedure: findProductById -

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `findProductById`(IN productId
INT)
BEGIN
        SELECT * FROM PRODUCTS WHERE id = productId;
END
```

StoredFunction: findCheapestProduct –

```
CREATE DEFINER=`root`@`localhost` FUNCTION `findCheapestProduct`() RETURNS
double
  READS SQL DATA
BEGIN
        DECLARE minPrice DOUBLE;
  SELECT MIN(price) INTO minPrice FROM PRODUCTS;
```

```
   RETURN minPrice;
END
```

StoredFunction: findCheapestProductName –

```
CREATE DEFINER=`root`@`localhost` FUNCTION `findCheapestProductName`()
RETURNS varchar(50) CHARSET utf8mb4
   READS SQL DATA
BEGIN
        DECLARE productName VARCHAR(50);
   SELECT name INTO productName FROM PRODUCTS ORDER BY price ASC LIMIT 1;
   RETURN productName;
END
```

StoredFunction: findMostExpensiveProduct -

```
CREATE DEFINER=`root`@`localhost` FUNCTION `findMostExpensiveProduct`()
RETURNS double
   READS SQL DATA
BEGIN
        DECLARE maxPrice DOUBLE;
   SELECT MAX(price) INTO maxPrice FROM PRODUCTS;
   RETURN maxPrice;
END
```

StoredFunction: findMostExpensiveProductName –

```
CREATE DEFINER=`root`@`localhost` FUNCTION `findMostExpensiveProductName`()
RETURNS varchar(50) CHARSET utf8mb4
   READS SQL DATA
BEGIN
        DECLARE productName VARCHAR(50);
   SELECT name INTO productName FROM PRODUCTS ORDER BY price DESC LIMIT
1;
   RETURN productName;
END
```

Output:

MySQLWorkbench

Result Grid | Filter Rows: Search

id	name	category	price
101	Mobile	Electronics	15000
102	Headphones	Electronics	5000
201	Apple	Grocery	200
202	Milk	Grocery	75
NULL	NULL	NULL	NULL

Terminal

```
------------------------------------------------
User Authentication
1. Register
2. Login
Enter your choice: 1
------------------------------------------------
Enter new username: admin01
Enter new password: admin@01
User registered successfully!
------------------------------------------------
User Authentication
1. Register
2. Login
Enter your choice: 2
------------------------------------------------
Enter username: admin01
Enter password: admin@01
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 1
------------------------------------------------
Connected to the database!
Product table created!
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 2
------------------------------------------------
Connected to the database!
Enter Product ID: 101
Enter Product Name: Apple
Enter Product Category: Grocery
Enter Product Price: 150
1 record added to products table successfully!
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
```

```
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 2
------------------------------------------------
Connected to the database!
Enter Product ID: 102
Enter Product Name: Mango
Enter Product Category: Grocery
Enter Product Price: 500
1 record added to products table successfully!
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 2
------------------------------------------------
Connected to the database!
Enter Product ID: 103
Enter Product Name: Cheery
Enter Product Category: Grocery
Enter Product Price: 100
1 record added to products table successfully!
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 1
------------------------------------------------
Connected to the database!
Product table created!
------------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
```

```
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 2
-----------------------------------------------
Connected to the database!
Enter Product ID: 201
Enter Product Name: Mobile
Enter Product Category: Electronics
Enter Product Price: 15000
1 record added to products table successfully!
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 2
-----------------------------------------------
Connected to the database!
Enter Product ID: 202
Enter Product Name: Laptop
Enter Product Category: Electronics
Enter Product Price: 100000
1 record added to products table successfully!
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 3
-----------------------------------------------
Connected to the database!
Product [id=101, name=Apple, category=Grocery, price=150.0]
Product [id=102, name=Mango, category=Grocery, price=500.0]
Product [id=103, name=Cheery, category=Grocery, price=100.0]
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
Product [id=202, name=Laptop, category=Electronics, price=100000.0]
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
```

```
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 4
--------------------------------------------
Connected to the database!
Product [id=101, name=Apple, category=Grocery, price=150.0]
Product [id=103, name=Cheery, category=Grocery, price=100.0]
Product [id=202, name=Laptop, category=Electronics, price=100000.0]
Product [id=102, name=Mango, category=Grocery, price=500.0]
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
--------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 5
--------------------------------------------
Connected to the database!
Enter Product ID to update: 103
Enter New Product Name: Cheery
Enter New Product Category: Grocery
Enter New Product Price: 75
1 record updated successfully!
--------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 3
--------------------------------------------
Connected to the database!
Product [id=101, name=Apple, category=Grocery, price=150.0]
Product [id=102, name=Mango, category=Grocery, price=500.0]
Product [id=103, name=Cheery, category=Grocery, price=75.0]
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
Product [id=202, name=Laptop, category=Electronics, price=100000.0]
--------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
```

```
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 6
-----------------------------------------------
Connected to the database!
Enter Product ID to delete: 103
1 record deleted successfully!
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 3
-----------------------------------------------
Connected to the database!
Product [id=101, name=Apple, category=Grocery, price=150.0]
Product [id=102, name=Mango, category=Grocery, price=500.0]
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
Product [id=202, name=Laptop, category=Electronics, price=100000.0]
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 7
-----------------------------------------------
Connected to the database!
Enter Product ID to find: 201
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
```

```
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 7
-----------------------------------------------
Connected to the database!
Enter Product ID to find: 201
Product [id=201, name=Mobile, category=Electronics, price=15000.0]
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 8
-----------------------------------------------
Connected to the database!
Most Expensive Product is Laptop and its price is Rs.100000.0
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 9
-----------------------------------------------
Connected to the database!
Cheapest Product is Apple and its price is Rs.150.0
-----------------------------------------------
Menu
1. Create Product Table
2. Add New Product
3. Show All Products
4. Show Products Sorted by Name
5. Update Product
6. Delete Product
7. Find Product By ID
8. Find Most Expensive Product Name and Price
9. Find Cheapest Product Name and Price
10. Exit
Enter your choice: 10
-----------------------------------------------
Exiting...
```