

```

1  package client;
2
3  import model.Product;
4  import repos.ProductRepo;
5  import service.ProductService;
6
7  import java.util.Scanner;
8
9  public class Client {
10     public static void main(String[] args) {
11
12         int id;
13
14         // object to access methods of ProductRepo
15         ProductRepo productRepo = new ProductRepo();
16         Scanner sc = new Scanner(System.in);
17
18         // // Purchase product
19         // for(Product p: productRepo.getALLProducts()) {
20         //     System.out.println(p);
21         // }
22         // System.out.print("\nEnter id of the product you want to buy: ");
23         // id = sc.nextInt();
24         // System.out.print("\nEnter quantity: ");
25         // int quantity = sc.nextInt();
26         // double bill_amt = ProductService.getBillAmount(id, quantity);
27         // System.out.println("Bill amount: " + bill_amt);
28         // System.out.println("Have you paid the bill (yes/no)? :");
29         // sc.nextLine();
30         // String bill_payment = sc.nextLine();
31         // if(bill_payment.equalsIgnoreCase("yes")) {
32         //     if(ProductService.purchase(id, quantity)) {
33         //         System.out.println("Purchase Complete.");
34         //     }
35         //     else {
36         //         System.out.println("Purchase could not be processed.");
37         //     }
38         // }
39
40         // SEARCH BY ID
41
42         System.out.println("Enter product id to search: ");
43         id = sc.nextInt();
44
45         Product product = productRepo.getProductById(id);
46
47         if(product == null) {
48             System.out.println("No such product found.");
49         }
50         else {
51             System.out.println(product);
52         }
53

```

```
54
55 // ADD PRODUCT
56 product = new Product();
57 System.out.println("Enter product id, title, price, quantity to save: ");
58 product.setId(sc.nextInt());
59 sc.nextLine();
60 product.setTitle(sc.nextLine());
61 product.setPrice(sc.nextDouble());
62 product.setQuantity(sc.nextInt());
63
64 System.out.println(product);
65 }
66 }
67
```

```

1 package model;
2
3 public class Product {
4     private int id;
5     private String title;
6     private double price;
7     private int quantity;
8
9     @Override
10    public String toString() {
11        return "Product{" +
12            "id=" + id +
13            ", title=" + title + "\" +
14            ", price=" + price +
15            ", quantity=" + quantity +
16            "}";
17    }
18
19    public Product() {
20    }
21
22    public Product(int id, String title, double price, int quantity) {
23        this.id = id;
24        this.title = title;
25        this.price = price;
26        this.quantity = quantity;
27    }
28
29    public int getId() {
30        return id;
31    }
32    public void setId(int id) {
33        this.id = id;
34    }
35    public String getTitle() {
36        return title;
37    }
38    public void setTitle(String title) {
39        this.title = title;
40    }
41    public double getPrice() {
42        return price;
43    }
44    public void setPrice(double price) {
45        this.price = price;
46    }
47    public int getQuantity() {
48        return quantity;
49    }
50    public void setQuantity(int quantity) {
51        this.quantity = quantity;
52    }
53 }

```

```

1 package repos;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 // Singleton Class
8 public class DBUtil {
9
10     private static Connection connection = null;
11     // Private constructor for singleton class
12     private DBUtil() {}
13
14     public static Connection getConnection() throws SQLException {
15         if(connection == null) {
16             // String username = "<your-username>";
17             String username = "<your-username>";
18             String password = "<your-password>";
19             String url = "jdbc:mysql://localhost:3306/<your-database>";
20             connection = DriverManager.getConnection(url, username, password);
21         }
22         return connection;
23     }
24 }

```

```

1  package repos;
2
3  import model.Product;
4
5  import java.sql.*;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  public class ProductRepo{
10     private Product convertRowToProduct(ResultSet resultSet) throws
        SQLException {
11         Product product = new Product();
12         product.setId(resultSet.getInt("id"));
13         product.setTitle(resultSet.getString("title"));
14         product.setPrice(resultSet.getDouble("price"));
15         product.setQuantity(resultSet.getInt("quantity"));
16         return product;
17     }
18     public List<Product> getALIProducts() {
19         List<Product> products = new ArrayList<>();
20         try(Connection connection = DBUtil.getConnection();
21             PreparedStatement pst = connection.prepareStatement(
        ProductQueries.GET_ALL_PRODUCTS)) {
22             ResultSet resultSet = pst.executeQuery();
23             while(resultSet.next()) {
24                 Product product = convertRowToProduct(resultSet);
25                 products.add(product);
26             }
27             resultSet.close();
28         }
29         catch (SQLException ex){
30             System.out.println("Problem:" + ex.getMessage());
31         }
32
33         return products;
34     }
35     public Product getProductById(int productId) {
36         Product product = null;
37         try(Connection connection = DBUtil.getConnection();
38             PreparedStatement pst = connection.prepareStatement(
        ProductQueries.GET_PRODUCT_BY_ID)) {
39
40             System.out.println("flag1");
41             // replacing ? in the statement
42             pst.setInt(1, productId);
43
44             System.out.println("flag2");
45             ResultSet resultSet = pst.executeQuery();
46
47             System.out.println("flag3");
48             while(resultSet.next()) {
49                 product = convertRowToProduct(resultSet);
50             }

```

```

51         resultSet.close();
52     } catch (SQLException e) {
53         System.out.println("Problem in getting products. " + e.getMessage());
54     }
55     return product;
56 }
57 public boolean save(Product product) {
58     try(Connection connection = DBUtil.getConnection();
59         PreparedStatement pst = connection.prepareStatement(
60         ProductQueries.ADD_PRODUCT)) {
61         pst.setInt(1, product.getId());
62         pst.setString(2, product.getTitle());
63         pst.setDouble(3, product.getPrice());
64         pst.setInt(4, product.getQuantity());
65
66         // executeUpdate() returns number of rows affected.
67         return pst.executeUpdate() == 1;
68
69     } catch (SQLException e) {
70         throw new RuntimeException("Problem in getting products.");
71     }
72 }
73 public boolean updateProductPrice(int productId, double newPrice) {
74     try(Connection connection = DBUtil.getConnection();
75         PreparedStatement pst = connection.prepareStatement(
76         ProductQueries.UPDATE_PRICE)) {
77         pst.setDouble(1, newPrice);
78         pst.setInt(2, productId);
79         // executeUpdate() returns number of rows affected.
80         return pst.executeUpdate() == 1;
81     } catch (SQLException e) {
82         throw new RuntimeException("Problem in getting products.");
83     }
84 }
85 public boolean updateProductQuantity(int productId, int newQuantity) {
86     try(Connection connection = DBUtil.getConnection();
87         PreparedStatement pst = connection.prepareStatement(
88         ProductQueries.UPDATE_QUANTITY)) {
89         pst.setDouble(1, newQuantity);
90         pst.setInt(2, productId);
91         // executeUpdate() returns number of rows affected.
92         return pst.executeUpdate() == 1;
93     } catch (SQLException e) {
94         throw new RuntimeException("Problem in getting products.");
95     }
96 }
97 public boolean remove(int productId) {
98     try(Connection connection = DBUtil.getConnection();
99         PreparedStatement pst = connection.prepareStatement(
100         ProductQueries.REMOVE_PRODUCT)) {
101         pst.setInt(1, productId);
102         // executeUpdate() returns number of rows affected.

```

```
100         return pst.executeUpdate() == 1;
101     } catch (SQLException e) {
102         throw new RuntimeException("Problem in getting products.");
103     }
104 }
105 }
106
```

```
1 package repos;
2
3 public class ProductQueries {
4     // add new product query
5     public static final String ADD_PRODUCT = "insert into products values
6     (?, ?, ?, ?)";
7     // read all products
8     public static final String GET_ALL_PRODUCTS = "select * from products";
9     // read single product
10    public static final String GET_PRODUCT_BY_ID = "select * from products
11    where id = ?";
12    // update price
13    public static final String UPDATE_PRICE = "update products set price = ?
14    where id = ?";
15    // update quantity
16    public static final String UPDATE_QUANTITY = "update products set
17    quantity = ? where id = ?";
18    // remove product
19    public static final String REMOVE_PRODUCT = "delete from products
20    where id = ?";
21 }
```



```

1 package service;
2
3 import model.Product;
4 import repos.DBUtil;
5 import repos.ProductRepo;
6
7 public class ProductService {
8
9     private static boolean checkQuantity(int id, int quantity){
10         ProductRepo productRepo = new ProductRepo();
11         Product product = productRepo.getProductById(id);
12         return product.getQuantity() <= quantity;
13     }
14
15
16     public static double getBillAmount(int id, int quantity) {
17         // getting the requested product
18         ProductRepo productRepo = new ProductRepo();
19         Product product = productRepo.getProductById(id);
20
21         // return -1 if quantity in stock is not enough
22         // if(!checkQuantity(id, quantity))
23         //     return -1;
24
25         return product.getPrice() * quantity;
26     }
27
28     public static boolean purchase(int id, int quantity) {
29         ProductRepo productRepo = new ProductRepo();
30         Product product = productRepo.getProductById(id);
31         // if(!checkQuantity(id, quantity))
32         //     return false;
33
34         return productRepo.updateProductQuantity(id, (product.getQuantity() -
35             quantity));
36     }
37
38
39 }

```

```
1 package service;  
2  
3 public class CountriesService {  
4 }  
5
```

```
1 package service;  
2  
3 public class EmployeesService {  
4 }  
5
```