```java
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Queue;
import java.util.Stack;
import java.sql.Date;
import java.sql.Time; // If using java.sql.Time



import javax.swing.JOptionPane;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class PharmacyManagementSystem {
    private Connection conn;
    private Map<String, Drug> drugMap;
    private Map<String, Supplier> supplierMap;
    private Map<String, Customer> customerMap;
    private Queue<Purchase> purchaseQueue;
    private Stack<Drug> drugStack;

    public PharmacyManagementSystem() {
        drugMap = new LinkedHashMap<>();
        supplierMap = new LinkedHashMap<>();
        customerMap = new LinkedHashMap<>();
        purchaseQueue = new LinkedList<>();
        drugStack = new Stack<>();

        connectToDatabase();
    }

    private void connectToDatabase() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver
```

```
41 ");
42             conn = DriverManager.getConnection("
   jdbc:mysql://localhost:3306/pharmacy", "root", "
   password");
43         } catch (ClassNotFoundException |
   SQLException e) {
44             JOptionPane.showMessageDialog(null, "
   Error connecting to database: " + e.getMessage());
45             System.exit(1);
46         }
47     }
48
49     public void addDrug(Drug drug) {
50         if (drugMap.containsKey(drug.getDrugCode
   ())) {
51             JOptionPane.showMessageDialog(null, "
   Drug already exists");
52             return;
53         }
54         drugMap.put(drug.getDrugCode(), drug);
55         try (PreparedStatement pstmt = conn.
   prepareStatement("INSERT INTO drugs (drug_code,
   name, description) VALUES (?,?,?)")) {
56             pstmt.setString(1, drug.getDrugCode());
57             pstmt.setString(2, drug.getName());
58             pstmt.setString(3, drug.getDescription
   ());
59             pstmt.executeUpdate();
60         } catch (SQLException e) {
61             JOptionPane.showMessageDialog(null, "
   Error adding drug: " + e.getMessage());
62         }
63     }
64
65     public void searchForDrug(String drugCode) {
66         Drug drug = drugMap.get(drugCode);
67         if (drug!= null) {
68             JOptionPane.showMessageDialog(null, "
   Drug found: " + drug.getName());
69         } else {
70             JOptionPane.showMessageDialog(null, "
```

```
70 Drug not found");
71         }
72     }
73 
74     public void viewAllDrugs() {
75         StringBuilder sb = new StringBuilder();
76         for (Drug drug : drugMap.values()) {
77             sb.append(drug.getName()).append("\n"
   );
78         }
79         JOptionPane.showMessageDialog(null, sb.
   toString());
80     }
81 
82     public void viewPurchaseHistory(String
   drugCode) {
83         List<Purchase> purchases = new ArrayList
   <>();
84         try (PreparedStatement pstmt = conn.
   prepareStatement("SELECT * FROM purchases WHERE
   drug_code =?")) {
85             pstmt.setString(1, drugCode);
86             try (ResultSet rs = pstmt.executeQuery
   ()) {
87                 while (rs.next()) {
88                     Purchase purchase = new
   Purchase(rs.getString("drug_code"), rs.getString("
   customer_name"), rs.getDate("purchase_date"), rs.
   getTime("purchase_time"), rs.getDouble("amount"));
89                     purchases.add(purchase);
90                 }
91             }
92         } catch (SQLException e) {
93             JOptionPane.showMessageDialog(null, "
   Error viewing purchase history: " + e.getMessage
   ());
94         }
95         purchases.sort((p1, p2) -> p1.
   getPurchaseDate().compareTo(p2.getPurchaseDate
   ()));
96         StringBuilder sb = new StringBuilder();
```

```
 97          for (Purchase purchase : purchases) {
 98              sb.append(purchase.toString()).append(
    "\n");
 99          }
100          JOptionPane.showMessageDialog(null, sb.
    toString());
101      }
102
103      public void addSupplier(Supplier supplier) {
104          if (supplierMap.containsKey(supplier.
    getSupplierCode())) {
105              JOptionPane.showMessageDialog(null, "
    Supplier already exists");
106              return;
107          }
108          supplierMap.put(supplier.getSupplierCode
    (), supplier);
109          try (PreparedStatement pstmt = conn.
    prepareStatement("INSERT INTO suppliers (
    supplier_code, name, location) VALUES (?,?,?)")) {
110              pstmt.setString(1, supplier.
    getSupplierCode());
111              pstmt.setString(2, supplier.getName
    ());
112              pstmt.setString(3, supplier.
    getLocation());
113              pstmt.executeUpdate();
114          } catch (SQLException e) {
115              JOptionPane.showMessageDialog(null, "
    Error adding supplier: " + e.getMessage());
116          }
117      }
118
119      public void linkDrugToSupplier(String drugCode
    , String supplierCode) {
120          Drug drug = drugMap.get(drugCode);
121          Supplier supplier = supplierMap.get(
    supplierCode);
122          if (drug!= null && supplier!= null) {
123              drug.addSupplier(supplier);
124              try (PreparedStatement pstmt = conn.
```

```
124 prepareStatement("INSERT INTO drug_suppliers (
    drug_code, supplier_code) VALUES (?,?)")) {
125                 pstmt.setString(1, drugCode);
126                 pstmt.setString(2, supplierCode);
127                 pstmt.executeUpdate();
128             } catch (SQLException e) {
129                 JOptionPane.showMessageDialog(null
    , "Error linking drug to supplier: " + e.
    getMessage());
130             }
131         }
132     }
133 
134     public void addCustomer(Customer customer) {
135         if (customerMap.containsKey(customer.
    getCustomerCode())) {
136             JOptionPane.showMessageDialog(null, "
    Customer already exists");
137             return;
138         }
139         customerMap.put(customer.getCustomerCode
    (), customer);
140         try (PreparedStatement pstmt = conn.
    prepareStatement("INSERT INTO customers (
    customer_code, name) VALUES (?,?)")) {
141             pstmt.setString(1, customer.
    getCustomerCode());
142             pstmt.setString(2, customer.getName
    ());
143             pstmt.executeUpdate();
144         } catch (SQLException e) {
145             JOptionPane.showMessageDialog(null, "
    Error adding customer: " + e.getMessage());
146         }
147     }
148 
149     public void makePurchase(String drugCode,
    String customerCode, double amount) {
150         Drug drug = drugMap.get(drugCode);
151         Customer customer = customerMap.get(
    customerCode);
```

```
152         if (drug != null && customer != null) {
153             long currentTimeMillis = System.
    currentTimeMillis();
154             Time currentTime = new Time(
    currentTimeMillis);
155 
156             // Create a java.sql.Date object (
    current date)
157             java.util.Date utilDate = new java.
    util.Date();
158             java.sql.Date sqlDate = new java.sql.
    Date(utilDate.getTime());
159 
160             Purchase purchase = new Purchase(
    drugCode, customer.getName(), sqlDate, currentTime
    , amount);
161             purchaseQueue.add(purchase);
162 
163             try (PreparedStatement pstmt = conn.
    prepareStatement("INSERT INTO purchases (drug_code
    , customer_name, purchase_date, purchase_time,
    amount) VALUES (?,?,?,?,?)")) {
164                 pstmt.setString(1, drugCode);
165                 pstmt.setString(2, customer.
    getName());
166                 pstmt.setDate(3, sqlDate);
167                 pstmt.setTime(4, currentTime);
168                 pstmt.setDouble(5, amount);
169                 pstmt.executeUpdate();
170             } catch (SQLException e) {
171                 JOptionPane.showMessageDialog(null
    , "Error making purchase: " + e.getMessage());
172             }
173         }
174     }
175 
176 
177 
178     public void generateReport() {
179         // Generate report using data structures
    and algorithms
```

```
180         //...
181     }
182
183     public void maintainStockBalance() {
184         // Maintain stock balance by checking the
    quantity of each drug
185         // and alerting the user if the stock is
    too low or too high
186         //...
187     }
188
189     public static void main(String[] args) {
190         PharmacyManagementSystem system = new
    PharmacyManagementSystem();
191         // Add drugs, suppliers, and customers
192         //...
193         // Search for drugs, view purchase history
    , and generate reports
194         //...
195     }
196 }
197
198 class Drug {
199     private String drugCode;
200     private String name;
201     private String description;
202     private List<Supplier> suppliers;
203
204     public Drug(String drugCode, String name,
    String description) {
205         this.drugCode = drugCode;
206         this.name = name;
207         this.description = description;
208         suppliers = new ArrayList<>();
209     }
210
211     public String getDrugCode() {
212         return drugCode;
213     }
214
215     public String getName() {
```

```
216         return name;
217     }
218
219     public String getDescription() {
220         return description;
221     }
222
223     public void addSupplier(Supplier supplier) {
224         suppliers.add(supplier);
225     }
226
227     public List<Supplier> getSuppliers() {
228         return suppliers;
229     }
230 }
231
232 class Supplier {
233     private String supplierCode;
234     private String name;
235     private String location;
236
237     public Supplier(String supplierCode, String
    name, String location) {
238         this.supplierCode = supplierCode;
239         this.name = name;
240         this.location = location;
241     }
242
243     public String getSupplierCode() {
244         return supplierCode;
245     }
246
247     public String getName() {
248         return name;
249     }
250
251     public String getLocation() {
252         return location;
253     }
254 }
255
```

```
256 class Customer {
257     private String customerCode;
258     private String name;
259 
260     public Customer(String customerCode, String
    name) {
261         this.customerCode = customerCode;
262         this.name = name;
263     }
264 
265     public String getCustomerCode() {
266         return customerCode;
267     }
268 
269     public String getName() {
270         return name;
271     }
272 }
273 
274 class Purchase {
275     private String drugCode;
276     private String customerName;
277     private Date purchaseDate;
278     private Time purchaseTime;
279     private double amount;
280 
281     public Purchase(String drugCode, String
    customerName, Date purchaseDate, Time purchaseTime
    , double amount) {
282         this.drugCode = drugCode;
283         this.customerName = customerName;
284         this.purchaseDate = purchaseDate;
285         this.purchaseTime = purchaseTime;
286         this.amount = amount;
287     }
288 
289     public String getDrugCode() {
290         return drugCode;
291     }
292 
293     public String getCustomerName() {
```

```
294         return customerName;
295     }
296
297     public Date getPurchaseDate() {
298         return purchaseDate;
299     }
300
301     public Time getPurchaseTime() {
302         return purchaseTime;
303     }
304
305     public double getAmount() {
306         return amount;
307     }
308
309     @Override
310     public String toString() {
311         return "Purchase{" +
312                 "drugCode='" + drugCode + '\'' +
313                 ", customerName='" + customerName
    + '\'' +
314                 ", purchaseDate=" + purchaseDate +
315                 ", purchaseTime=" + purchaseTime +
316                 ", amount=" + amount +
317                 '}';
318     }
319 }
```