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

import java.util.Map;
import java.util.Scanner;
import java.util.LinkedHashMap;
import java.time.LocalDate;
import java.util.List;
import java.util.ArrayList;

/*
Introductory JDBC examples based loosely on the BAKERY dataset from CSC 365 labs.

-- MySQL setup:
drop table if exists hp_goods, hp_customers, hp_items, hp_receipts;
create table hp_goods as select * from BAKERY.goods;
create table hp_customers as select * from BAKERY.customers;
create table hp_items as select * from BAKERY.items;
create table hp_receipts as select * from BAKERY.receipts;

-- Shell init:
export CLASSPATH=$CLASSPATH:mysql-connector-java-5.1.44-bin.jar:.
export HP_JDBC_URL=jdbc:mysql://csc365fall2017.webredirect.org/<database name>?
autoReconnect=true\&useSSL=false
export HP_JDBC_USER=
export HP_JDBC_PW=
 */
public class HastyPastry {
    public static void main(String[] args) {
        try {
            HastyPastry hp = new HastyPastry();
            hp.demo5();
        } catch (SQLException e) {
            System.err.println("SQLException: " + e.getMessage());
        }
    }

    // Demo1 - Establish JDBC connection, execute DDL statement
    private void demo1() throws SQLException {

        // Step 0: Load MySQL JDBC Driver
        // No longer required as of JDBC 2.0  / Java 6
        try{
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("MySQL JDBC Driver loaded");
        } catch (ClassNotFoundException ex) {
            System.err.println("Unable to load JDBC Driver");
            System.exit(-1);
        }

        // Step 1: Establish connection to RDBMS
        try (Connection conn = DriverManager.getConnection(System.getenv("HP_JDBC_URL"),
                                                           System.getenv("HP_JDBC_USER"),
                                                           System.getenv("HP_JDBC_PW"))) {
            // Step 2: Construct SQL statement
            String sql = "ALTER TABLE hp_goods ADD COLUMN AvailUntil DATE";

            // Step 3: (omitted in this example) Start transaction

            try (Statement stmt = conn.createStatement()) {

                // Step 4: Send SQL statement to DBMS
                boolean exRes = stmt.execute(sql);

                // Step 5: Handle results
                System.out.format("Result from ALTER: %b %n", exRes);
            }
```

```
            // Step 6: (omitted in this example) Commit or rollback transaction
        }
        // Step 7: Close connection (handled by try-with-resources syntax)
    }


    // Demo2 - Establish JDBC connection, execute SELECT query, read & print result
    private void demo2() throws SQLException {

        // Step 1: Establish connection to RDBMS
        try (Connection conn = DriverManager.getConnection(System.getenv("HP_JDBC_URL"),
                                                           System.getenv("HP_JDBC_USER"),
                                                           System.getenv("HP_JDBC_PW"))) {
            // Step 2: Construct SQL statement
            String sql = "SELECT * FROM hp_goods";

            // Step 3: (omitted in this example) Start transaction

            // Step 4: Send SQL statement to DBMS
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(sql)) {

                // Step 5: Receive results
                while (rs.next()) {
                    String flavor = rs.getString("Flavor");
                    String food = rs.getString("Food");
                    float price = rs.getFloat("price");
                    System.out.format("%s %s ($%.2f) %n", flavor, food, price);
                }
            }

            // Step 6: (omitted in this example) Commit or rollback transaction
        }
        // Step 7: Close connection (handled by try-with-resources syntax)
    }


    // Demo3 - Establish JDBC connection, execute DML query (UPDATE)
    // ------------------------------------------
    // Never (ever) write database code like this!
    // ------------------------------------------
    private void demo3() throws SQLException {

        // Step 1: Establish connection to RDBMS
        try (Connection conn = DriverManager.getConnection(System.getenv("HP_JDBC_URL"),
                                                           System.getenv("HP_JDBC_USER"),
                                                           System.getenv("HP_JDBC_PW"))) {
            // Step 2: Construct SQL statement
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a flavor: ");
            String flavor = scanner.nextLine();
            System.out.format("Until what date will %s be available (YYYY-MM-DD)? ", flavor);
            String availUntilDate = scanner.nextLine();

            // ------------------------------------------
            // Never (ever) write database code like this!
            // ------------------------------------------
            String updateSql = "UPDATE hp_goods SET AvailUntil = '" + availUntilDate + "' " +
                               "WHERE Flavor = '" + flavor + "'";

            // Step 3: (omitted in this example) Start transaction

            try (Statement stmt = conn.createStatement()) {

                // Step 4: Send SQL statement to DBMS
                int rowCount = stmt.executeUpdate(updateSql);

                // Step 5: Handle results
                System.out.format("Updated %d records for %s pastries%n", rowCount, flavor);
            }

            // Step 6: (omitted in this example) Commit or rollback transaction
```

```
        }
        // Step 7: Close connection (handled implcitly by try-with-resources syntax)
    }


    // Demo4 - Establish JDBC connection, execute DML query (UPDATE) using PreparedStatement /
transaction
    private void demo4() throws SQLException {

        // Step 1: Establish connection to RDBMS
        try (Connection conn = DriverManager.getConnection(System.getenv("HP_JDBC_URL"),
                                                           System.getenv("HP_JDBC_USER"),
                                                           System.getenv("HP_JDBC_PW"))) {
            // Step 2: Construct SQL statement
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a flavor: ");
            String flavor = scanner.nextLine();
            System.out.format("Until what date will %s be available (YYYY-MM-DD)? ", flavor);
            LocalDate availDt = LocalDate.parse(scanner.nextLine());

            String updateSql = "UPDATE hp_goods SET AvailUntil = ? WHERE Flavor = ?";

            // Step 3: Start transaction
            conn.setAutoCommit(false);

            try (PreparedStatement pstmt = conn.prepareStatement(updateSql)) {

                // Step 4: Send SQL statement to DBMS
                pstmt.setDate(1, java.sql.Date.valueOf(availDt));
                pstmt.setString(2, flavor);
                int rowCount = pstmt.executeUpdate();

                // Step 5: Handle results
                System.out.format("Updated %d records for %s pastries%n", rowCount, flavor);

                // Step 6: Commit or rollback transaction
                conn.commit();
            } catch (SQLException e) {
                conn.rollback();
            }

        }
        // Step 7: Close connection (handled implcitly by try-with-resources syntax)
    }



    // Demo5 - Construct a query using PreparedStatement
    private void demo5() throws SQLException {

        // Step 1: Establish connection to RDBMS
        try (Connection conn = DriverManager.getConnection(System.getenv("HP_JDBC_URL"),
                                                           System.getenv("HP_JDBC_USER"),
                                                           System.getenv("HP_JDBC_PW"))) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Find pastries with price <=: ");
            Double price = Double.valueOf(scanner.nextLine());
            System.out.print("Filter by flavor (or 'Any'): ");
            String flavor = scanner.nextLine();

            List<Object> params = new ArrayList<Object>();
            params.add(price);
            StringBuilder sb = new StringBuilder("SELECT * FROM hp_goods WHERE price <= ?");
            if (!"any".equalsIgnoreCase(flavor)) {
                sb.append(" AND Flavor = ?");
                params.add(flavor);
            }

            try (PreparedStatement pstmt = conn.prepareStatement(sb.toString())) {
                int i = 1;
                for (Object p : params) {
                    pstmt.setObject(i++, p);
                }
```

```
                try (ResultSet rs = pstmt.executeQuery()) {
                    System.out.println("Matching Pastries:");
                    int matchCount = 0;
                    while (rs.next()) {
                        System.out.format("%s %s ($%.2f) %n", rs.getString("Flavor"),
rs.getString("Food"), rs.getDouble("price"));
                        matchCount++;
                    }
                    System.out.format("----------------------%nFound %d match%s %n", matchCount,
matchCount == 1 ? "" : "es");
                }
            }

        }
    }


}
```