// Product class representing individual products

**public** **class** Product {

**private** **int** productId;

**private** String productName;

**private** **int** quantityOnHand;

**public** Product(**int** productId, String productName, **int** quantityOnHand) {

**this**.productId = productId;

**this**.productName = productName;

**this**.quantityOnHand = quantityOnHand;

}

// Getters and setters

**public** **int** getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

**public** **int** getQuantityOnHand() {

**return** quantityOnHand;

}

**public** **void** setQuantityOnHand(**int** quantityOnHand) {

**this**.quantityOnHand = quantityOnHand;

}

}

**import** java.util.List;

**import** java.util.Random;

// Transaction class to simulate concurrent transactions

**public** **class** Transaction **extends** Thread {

**private** List<Product> inventory;

**private** Random random = **new** Random();

**public** Transaction(List<Product> inventory) {

**this**.inventory = inventory;

}

// Method to simulate purchase transaction

@Override

**public** **void** run() {

**try** {

Thread.*sleep*(random.nextInt(1000)); // Simulating some delay

**int** index = random.nextInt(inventory.size());

Product product = inventory.get(index);

**int** quantityBought = random.nextInt(10) + 1; // Buying 1 to 10 items

**synchronized** (product) { // Synchronizing on product object to avoid concurrent modifications

**int** currentQuantity = product.getQuantityOnHand();

**if** (currentQuantity >= quantityBought) {

product.setQuantityOnHand(currentQuantity - quantityBought);

System.***out***.println(Thread.*currentThread*().getName() + " bought " + quantityBought +

" units of " + product.getProductName());

} **else** {

System.***out***.println(Thread.*currentThread*().getName() + " couldn't buy " + quantityBought +

" units of " + product.getProductName() + " due to insufficient quantity");

}

}

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

// Main class for Inventory Management System

public class InventoryManagementSystem {

private List<Product> inventory = new ArrayList<>();

private Connection connection;

// Method to establish database connection

private void connectToDatabase() {

try {

// Load MySQL JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Connect to the database

String url = "jdbc:mysql://localhost:3306/inventory\_db";

String username = "your\_username";

String password = "your\_password";

connection = DriverManager.getConnection(url, username, password);

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

}

}

// Method to create product table in the database

private void createProductTable() {

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

// SQL query to create product table

String sql = "CREATE TABLE IF NOT EXISTS products (" +

"productId INT AUTO\_INCREMENT PRIMARY KEY," +

"productName VARCHAR(255) NOT NULL," +

"quantityOnHand INT NOT NULL)";

stmt.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to add a new product to the inventory and database

public void addProduct(Product product) {

inventory.add(product);

// Insert product into the database

try (PreparedStatement pstmt = connection.prepareStatement("INSERT INTO products (productName, quantityOnHand) VALUES (?, ?)")) {

pstmt.setString(1, product.getProductName());

pstmt.setInt(2, product.getQuantityOnHand());

pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to update product quantity in the inventory and database

public void updateProductQuantity(int productId, int newQuantity) {

for (Product product : inventory) {

if (product.getProductId() == productId) {

product.setQuantityOnHand(newQuantity);

// Update product quantity in the database

try (PreparedStatement pstmt = connection.prepareStatement("UPDATE products SET quantityOnHand = ? WHERE productId = ?")) {

pstmt.setInt(1, newQuantity);

pstmt.setInt(2, productId);

pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

break;

}

}

}

// Method to display current inventory

public void displayInventory() {

System.out.println("Current Inventory:");

for (Product product : inventory) {

System.out.println("Product ID: " + product.getProductId() +

", Product Name: " + product.getProductName() +

", Quantity: " + product.getQuantityOnHand());

}

}

public static void main(String[] args) {

InventoryManagementSystem ims = new InventoryManagementSystem();

ims.connectToDatabase(); // Connect to the database

ims.createProductTable(); // Create product table if not exists

// Adding some initial products to the inventory and database

ims.addProduct(new Product(1, "Laptop", 10));

ims.addProduct(new Product(2, "Mouse", 20));

ims.addProduct(new Product(3, "Keyboard", 15));

// Displaying current inventory

ims.displayInventory();

// Creating and starting transactions

Transaction transaction1 = new Transaction(ims.inventory);

Transaction transaction2 = new Transaction(ims.inventory);

transaction1.start();

transaction2.start();

}

}