JAVA FUNDAMENTALS SECTION-07 PART-1

### BANAGANAPALLI MUNNA

### 192372028

### Step 1: Modify the ProductTester Class

**a. Create the** displayInventory **Method**

Add this method after the main method in ProductTester:

public static void displayInventory(Product[] products) {

for (int i = 0; i < products.length; i++) {

System.out.println(i + ". " + products[i].getName() + ": " + products[i].getQuantity() + " units");

}

}

**b. Move the Display Code**

Move the code that displays the inventory from the main method to the displayInventory method. Update the main method to call displayInventory.

**c. Update** main **to Call** displayInventory

Replace the display code in the main method with:

displayInventory(products);

**d. Run and Test Your Code**

Make sure the code compiles and runs without issues. Verify that the inventory is displayed correctly.

**e. Create the** addToInventory **Method**

Add this method after the main method:

public static void addToInventory(Product[] products, Scanner scanner) {

int tempNumber;

String tempName;

int tempQty;

double tempPrice;

for (int i = 0; i < products.length; i++) {

System.out.println("Enter details for product " + (i + 1));

System.out.print("Product number: ");

tempNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Product name: ");

tempName = scanner.nextLine();

System.out.print("Quantity: ");

tempQty = scanner.nextInt();

System.out.print("Price: ");

tempPrice = scanner.nextDouble();

scanner.nextLine(); // Consume newline

products[i] = new Product(tempNumber, tempName, tempQty, tempPrice);

}

}

**f. Update** main **Method to Call** addToInventory

Replace the loop that adds values to the array with:

addToInventory(products, scanner);

**g. Move Local Variables to** addToInventory

Move the local variables (tempNumber, tempName, tempQty, tempPrice) to the top of the addToInventory method.

**h. Create** getNumProducts **Method**

Add this method to ProductTester:

public static int getNumProducts(Scanner scanner) {

int maxSize;

while (true) {

System.out.print("Enter the number of products: ");

try {

maxSize = scanner.nextInt();

if (maxSize > 0) {

break;

} else {

System.out.println("Number of products must be greater than 0.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a positive integer.");

scanner.next(); // Clear the invalid input

}

}

return maxSize;

}

**i. Update** main **Method to Call** getNumProducts

Replace the code that gets the number of products with:

int maxSize = getNumProducts(scanner);

Product[] products = new Product[maxSize];

**j. Run and Test Your Code**

Compile and test the updated code to ensure it works correctly.

### Step 2: Modify the Product Class

**a. Add** addToInventory **and** deductFromInventory **Methods**

Add these methods to the Product class:

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (quantity <= this.quantity) {

this.quantity -= quantity;

} else {

System.out.println("Error: Quantity to deduct exceeds current stock.");

}

}

### Step 3: Create Menu Methods in ProductTester

**a. Create** getMenuOption **Method**

Add this method to display the menu:

public static int getMenuOption(Scanner scanner) {

int menuOption = -1;

while (true) {

System.out.println("1. View Inventory");

System.out.println("2. Add Stock");

System.out.println("3. Deduct Stock");

System.out.println("4. Discontinue Product");

System.out.println("0. Exit");

System.out.print("Please enter a menu option: ");

try {

menuOption = scanner.nextInt();

if (menuOption >= 0 && menuOption <= 4) {

break;

} else {

System.out.println("Invalid option. Please choose a number between 0 and 4.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a number.");

scanner.next(); // Clear the invalid input

}

}

return menuOption;

}

**b. Create** getProductNumber **Method**

Add this method to allow user to select a product:

public static int getProductNumber(Product[] products, Scanner scanner) {

int productChoice = -1;

while (true) {

for (int i = 0; i < products.length; i++) {

System.out.println(i + ". " + products[i].getName());

}

System.out.print("Enter the product number to update: ");

try {

productChoice = scanner.nextInt();

if (productChoice >= 0 && productChoice < products.length) {

break;

} else {

System.out.println("Invalid product number. Please choose a valid number.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a valid number.");

scanner.next(); // Clear the invalid input

}

}

return productChoice;

}

**c. Create** addInventory **Method**

Add this method to add stock:

public static void addInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (true) {

System.out.print("How many products do you want to add? ");

try {

updateValue = scanner.nextInt();

if (updateValue >= 0) {

products[productChoice].addToInventory(updateValue);

break;

} else {

System.out.println("Quantity must be 0 or greater.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a non-negative integer.");

scanner.next(); // Clear the invalid input

}

}

}

**d. Create** deductInventory **Method**

Add this method to deduct stock:

public static void deductInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (true) {

System.out.print("How many products do you want to deduct? ");

try {

updateValue = scanner.nextInt();

if (updateValue >= 0 && updateValue <= products[productChoice].getQuantity()) {

products[productChoice].deductFromInventory(updateValue);

break;

} else {

System.out.println("Quantity must be between 0 and current stock.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a non-negative integer.");

scanner.next(); // Clear the invalid input

}

}

}

**e. Create** discontinueInventory **Method**

Add this method to mark a product as discontinued:

public static void discontinueInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

products[productChoice].setActive(false);

}

**f. Create** executeMenuChoice **Method**

Add this method to execute the chosen menu option:

public static void executeMenuChoice(int menuChoice, Product[] products, Scanner scanner) {

switch (menuChoice) {

case 1:

System.out.println("View Product List");

displayInventory(products);

break;

case 2:

System.out.println("Add Stock");

addInventory(products, scanner);

break;

case 3:

System.out.println("Deduct Stock");

deductInventory(products, scanner);

break;

case 4:

System.out.println("Discontinue Stock");

discontinueInventory(products, scanner);

break;

case 0:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid option.");

}

}

**g. Update** main **Method**

Update the main method to use the new functionality:

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int maxSize = getNumProducts(scanner);

Product[] products = new Product[maxSize];

addToInventory(products, scanner);

int menuChoice;

do {

menuChoice = getMenuOption(scanner);

executeMenuChoice(menuChoice, products, scanner);

} while (menuChoice != 0);

scanner.close();

}

**h. Save Your Project**

Make sure to save all your files in your project.

JAVA

import java.util.Scanner;

// Product class

class Product {

private int number;

private String name;

private int quantity;

private double price;

private boolean active;

public Product(int number, String name, int quantity, double price) {

this.number = number;

this.name = name;

this.quantity = quantity;

this.price = price;

this.active = true;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (quantity <= this.quantity) {

this.quantity -= quantity;

} else {

System.out.println("Error: Quantity to deduct exceeds current stock.");

}

}

public void setActive(boolean active) {

this.active = active;

}

public boolean isActive() {

return active;

}

}

// ProductTester class

public class ProductTester {

public static void displayInventory(Product[] products) {

for (int i = 0; i < products.length; i++) {

if (products[i].isActive()) {

System.out.println(i + ". " + products[i].getName() + ": " + products[i].getQuantity() + " units");

}

}

}

public static void addToInventory(Product[] products, Scanner scanner) {

int tempNumber;

String tempName;

int tempQty;

double tempPrice;

for (int i = 0; i < products.length; i++) {

System.out.println("Enter details for product " + (i + 1));

System.out.print("Product number: ");

tempNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Product name: ");

tempName = scanner.nextLine();

System.out.print("Quantity: ");

tempQty = scanner.nextInt();

System.out.print("Price: ");

tempPrice = scanner.nextDouble();

scanner.nextLine(); // Consume newline

products[i] = new Product(tempNumber, tempName, tempQty, tempPrice);

}

}

public static int getNumProducts(Scanner scanner) {

int maxSize;

while (true) {

System.out.print("Enter the number of products: ");

try {

maxSize = scanner.nextInt();

if (maxSize > 0) {

break;

} else {

System.out.println("Number of products must be greater than 0.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a positive integer.");

scanner.next(); // Clear the invalid input

}

}

return maxSize;

}

public static int getMenuOption(Scanner scanner) {

int menuOption = -1;

while (true) {

System.out.println("1. View Inventory");

System.out.println("2. Add Stock");

System.out.println("3. Deduct Stock");

System.out.println("4. Discontinue Product");

System.out.println("0. Exit");

System.out.print("Please enter a menu option: ");

try {

menuOption = scanner.nextInt();

if (menuOption >= 0 && menuOption <= 4) {

break;

} else {

System.out.println("Invalid option. Please choose a number between 0 and 4.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a number.");

scanner.next(); // Clear the invalid input

}

}

return menuOption;

}

public static int getProductNumber(Product[] products, Scanner scanner) {

int productChoice = -1;

while (true) {

for (int i = 0; i < products.length; i++) {

if (products[i].isActive()) {

System.out.println(i + ". " + products[i].getName());

}

}

System.out.print("Enter the product number to update: ");

try {

productChoice = scanner.nextInt();

if (productChoice >= 0 && productChoice < products.length && products[productChoice].isActive()) {

break;

} else {

System.out.println("Invalid product number. Please choose a valid number.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a valid number.");

scanner.next(); // Clear the invalid input

}

}

return productChoice;

}

public static void addInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (true) {

System.out.print("How many products do you want to add? ");

try {

updateValue = scanner.nextInt();

if (updateValue >= 0) {

products[productChoice].addToInventory(updateValue);

break;

} else {

System.out.println("Quantity must be 0 or greater.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a non-negative integer.");

scanner.next(); // Clear the invalid input

}

}

}

public static void deductInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (true) {

System.out.print("How many products do you want to deduct? ");

try {

updateValue = scanner.nextInt();

if (updateValue >= 0 && updateValue <= products[productChoice].getQuantity()) {

products[productChoice].deductFromInventory(updateValue);

break;

} else {

System.out.println("Quantity must be between 0 and current stock.");

}

} catch (Exception e) {

System.out.println("Invalid input. Please enter a non-negative integer.");

scanner.next(); // Clear the invalid input

}

}

}

public static void discontinueInventory(Product[] products, Scanner scanner) {

int productChoice = getProductNumber(products, scanner);

products[productChoice].setActive(false);

}

public static void executeMenuChoice(int menuChoice, Product[] products, Scanner scanner) {

switch (menuChoice) {

case 1:

System.out.println("View Product List");

displayInventory(products);

break;

case 2:

System.out.println("Add Stock");

addInventory(products, scanner);

break;

case 3:

System.out.println("Deduct Stock");

deductInventory(products, scanner);

break;

case 4:

System.out.println("Discontinue Stock");

discontinueInventory(products, scanner);

break;

case 0:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid option.");

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int maxSize = getNumProducts(scanner);

Product[] products = new Product[maxSize];

addToInventory(products, scanner);

int menuChoice;

do {

menuChoice = getMenuOption(scanner);

executeMenuChoice(menuChoice, products, scanner);

} while (menuChoice != 0);

scanner.close();

}

}

OUTPUT

