**Project:**

**Pusalapati Chandu**

**192372119**

**Step 1: Create Static Methods in ProductTester**

**a) Create displayInventory Method**

java

Copy code

public class ProductTester {

public static void displayInventory(Product[] products) {

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

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

}

}

// Other methods and main method will be added here

}

**b) Copy Display Code from Main Method**

Remove the code displaying the products array from the main method and call displayInventory:

java

Copy code

// Inside main method

displayInventory(products);

**c) Create addToInventory Method**

java

Copy code

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.print("Enter product number: ");

tempNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Enter product name: ");

tempName = scanner.nextLine();

System.out.print("Enter quantity: ");

tempQty = scanner.nextInt();

System.out.print("Enter price: ");

tempPrice = scanner.nextDouble();

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

}

}

**d) Modify main to Call addToInventory**

Replace the for loop in the main method with:

java

Copy code

addToInventory(products, scanner);

**e) Create getNumProducts Method**

java

Copy code

public static int getNumProducts(Scanner scanner) {

int maxSize = -1;

while (maxSize < 0) {

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

try {

maxSize = scanner.nextInt();

if (maxSize < 0) {

System.out.println("Number must be positive.");

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return maxSize;

}

**f) Call getNumProducts in main**

java

Copy code

int maxSize = getNumProducts(scanner);

**2. Update Product Class**

**a) Add addToInventory and deductFromInventory Methods**

java

Copy code

public class Product {

private int number;

private String name;

private int quantity;

private double price;

private boolean active;

// Existing constructors, getters, and setters

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (this.quantity >= quantity) {

this.quantity -= quantity;

} else {

System.out.println("Not enough stock to deduct.");

}

}

public void setActive(boolean active) {

this.active = active;

}

}

**3. Implement Menu System**

**a) Create getMenuOption Method**

java

Copy code

public static int getMenuOption(Scanner scanner) {

int option = -1;

while (option < 0 || option > 4) {

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 {

option = scanner.nextInt();

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return option;

}

**b) Create getProductNumber Method**

java

Copy code

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

int productChoice = -1;

while (productChoice < 0 || productChoice >= products.length) {

System.out.println("Select a product by number:");

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

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

}

try {

productChoice = scanner.nextInt();

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

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

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return productChoice;

}

**c) Create addInventory Method**

java

Copy code

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

int productChoice;

int updateValue = -1;

productChoice = getProductNumber(products, scanner);

while (updateValue < 0) {

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

try {

updateValue = scanner.nextInt();

if (updateValue < 0) {

System.out.println("Quantity must be non-negative.");

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

products[productChoice].addToInventory(updateValue);

}

**d) Create deductInventory Method**

java

Copy code

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

int productChoice;

int updateValue = -1;

productChoice = getProductNumber(products, scanner);

while (updateValue < 0 || updateValue > products[productChoice].getQuantity()) {

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

try {

updateValue = scanner.nextInt();

if (updateValue < 0) {

System.out.println("Quantity must be non-negative.");

} else if (updateValue > products[productChoice].getQuantity()) {

System.out.println("Cannot deduct more than available stock.");

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

products[productChoice].deductFromInventory(updateValue);

}

**e) Create discontinueInventory Method**

java

Copy code

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

int productChoice;

productChoice = getProductNumber(products, scanner);

products[productChoice].setActive(false);

}

**f) Create executeMenuChoice Method**

java

Copy code

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;

}

}

**g) Update main Method**

java

Copy code

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();

}

**FINAL CODE:**

**Here is the code for the: Product.java” and “ProductTester” classes**

**“Product.java”**

public 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;

}

// Getters

public int getNumber() {

return number;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public double getPrice() {

return price;

}

public boolean isActive() {

return active;

}

// Methods to update inventory

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (this.quantity >= quantity) {

this.quantity -= quantity;

} else {

System.out.println("Not enough stock to deduct.");

}

}

public void setActive(boolean active) {

this.active = active;

}

}

**“Product Tester.java”**

import java.util.InputMismatchException;

import java.util.Scanner;

public class ProductTester {

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();

}

public static void displayInventory(Product[] products) {

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

if (products[i] != null && products[i].isActive()) {

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

}

}

}

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.print("Enter product number: ");

tempNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Enter product name: ");

tempName = scanner.nextLine();

System.out.print("Enter quantity: ");

tempQty = scanner.nextInt();

System.out.print("Enter price: ");

tempPrice = scanner.nextDouble();

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

}

}

public static int getNumProducts(Scanner scanner) {

int maxSize = -1;

while (maxSize < 0) {

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

try {

maxSize = scanner.nextInt();

if (maxSize < 0) {

System.out.println("Number must be positive.");

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return maxSize;

}

public static int getMenuOption(Scanner scanner) {

int option = -1;

while (option < 0 || option > 4) {

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 {

option = scanner.nextInt();

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return option;

}

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

int productChoice = -1;

while (productChoice < 0 || productChoice >= products.length) {

System.out.println("Select a product by number:");

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

if (products[i] != null && products[i].isActive()) {

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

}

}

try {

productChoice = scanner.nextInt();

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

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

}

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

}

}

return productChoice;

}

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

int productChoice;

int updateValue = -1;

productChoice = getProductNumber(products, scanner);

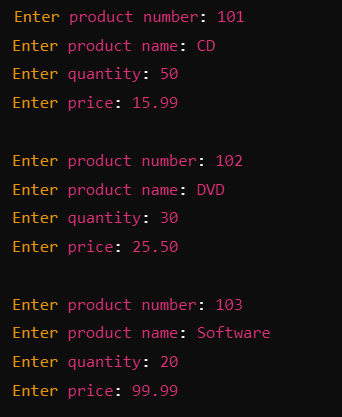
while (updateValue < 0) {

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

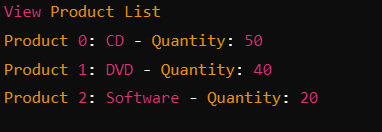
try {

updateValue = scanner

**INPUT:**

****

**OUTPUT:**

****