**Java fundamentals(module-1)**

**PROJECT-1**

**SECTION-4:**

**Inventory Project**

### ****Project Overview****

This project will progress through several sections, gradually building into a complete Java application for maintaining an inventory system. The program will be used for a range of different products like CDs, DVDs, software, etc. Each section will build upon the last to meet both the old and new requirements. All parts will be included in a package called inventory.

### ****Step 1: Identify Products****

For the inventory system, the following six products have been identified:

1. **Product Name:** USB Drive
   1. **Price:** $12.99
   2. **Units in Stock:** 100
   3. **Item Number:** 101
2. **Product Name:** Headphones
   1. **Price:** $49.99
   2. **Units in Stock:** 50
   3. **Item Number:** 102
3. **Product Name:** External Hard Drive
   1. **Price:** $89.99
   2. **Units in Stock:** 25
   3. **Item Number:** 103
4. **Product Name:** Wireless Mouse
   1. **Price:** $25.99
   2. **Units in Stock:** 75
   3. **Item Number:** 104
5. **Product Name:** Keyboard
   1. **Price:** $29.99
   2. **Units in Stock:** 80
   3. **Item Number:** 105
6. **Product Name:** Monitor
   1. **Price:** $199.99
   2. **Units in Stock:** 30
   3. **Item Number:** 106

### ****Step 2: Determine Data Types****

| **Attribute** | **Sample Data** | **Data Type** |
| --- | --- | --- |
| Name of the product | USB Drive | String |
| Price | 12.99 | double |
| Number of units in stock | 100 | int |
| Item number | 101 | int |

### ****Step 3: Create the Project****

* Project Name: inventory

### ****Step 4: Create the**** Product ****Class****

java

Copy code

package inventory;

/\*\*

\* This class represents a product in the inventory system.

\*/public class Product {

// Instance field declarations

private int itemNumber;

private String name;

private int quantity;

private double price;

/\*\*

\* Default constructor that initializes the fields to their default values.

\* Purpose: To allow the creation of a Product object with default values.

\*/

public Product() {

this.itemNumber = 0;

this.name = "";

this.quantity = 0;

this.price = 0.0;

}

/\*\*

\* Parameterized constructor to initialize the product with specific values.

\* @param number the item number

\* @param name the name of the product

\* @param qty the quantity in stock

\* @param price the price of each unit

\* Purpose: To create a Product object with specified values.

\*/

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

this.itemNumber = number;

this.name = name;

this.quantity = qty;

this.price = price;

}

// Getter and setter methods

/\*\*

\* Gets the item number.

\* @return the item number

\*/

public int getItemNumber() {

return itemNumber;

}

/\*\*

\* Sets the item number.

\* @param itemNumber the item number to set

\*/

public void setItemNumber(int itemNumber) {

this.itemNumber = itemNumber;

}

/\*\*

\* Gets the name of the product.

\* @return the name of the product

\*/

public String getName() {

return name;

}

/\*\*

\* Sets the name of the product.

\* @param name the name of the product to set

\*/

public void setName(String name) {

this.name = name;

}

/\*\*

\* Gets the quantity in stock.

\* @return the quantity in stock

\*/

public int getQuantity() {

return quantity;

}

/\*\*

\* Sets the quantity in stock.

\* @param quantity the quantity to set

\*/

public void setQuantity(int quantity) {

this.quantity = quantity;

}

/\*\*

\* Gets the price of each unit.

\* @return the price of each unit

\*/

public double getPrice() {

return price;

}

/\*\*

\* Sets the price of each unit.

\* @param price the price to set

\*/

public void setPrice(double price) {

this.price = price;

}

/\*\*

\* Returns a string representation of the product.

\* @return a description of the product

\*/

@Override

public String toString() {

return "Item Number : " + itemNumber + "\n" +

"Name : " + name + "\n" +

"Quantity in stock: " + quantity + "\n" +

"Price : " + price;

}

}

### ****Step 10: Create the**** ProductTester ****Class****

java

Copy code

package inventory;

public class ProductTester {

public static void main(String[] args) {

// Create and initialize six Product objects

// Using default constructor

Product product1 = new Product();

Product product2 = new Product();

// Using parameterized constructor

Product product3 = new Product(101, "USB Drive", 100, 12.99);

Product product4 = new Product(102, "Headphones", 50, 49.99);

Product product5 = new Product(103, "External Hard Drive", 25, 89.99);

Product product6 = new Product(104, "Wireless Mouse", 75, 25.99);

// Display details of each product

System.out.println(product1.toString());

System.out.println(product2.toString());

System.out.println(product3.toString());

System.out.println(product4.toString());

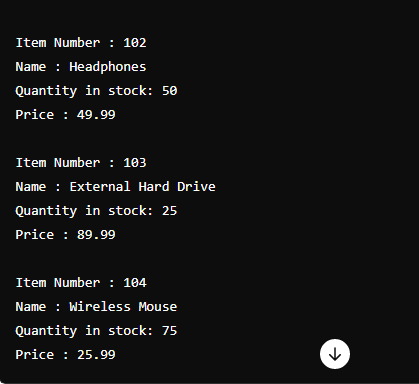
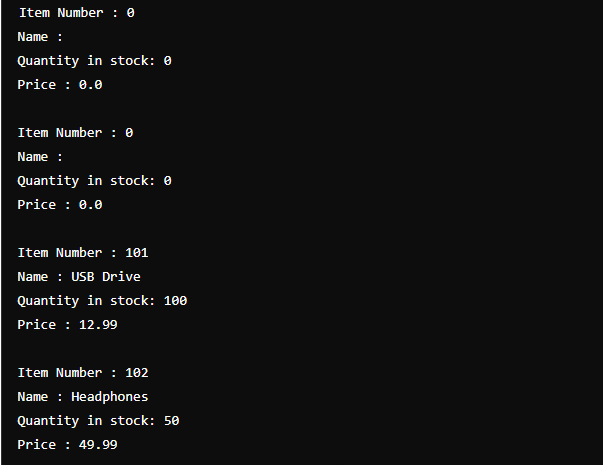
System.out.println(product5.toString());

System.out.println(product6.toString());

}

}

### ****Step 12: Display Product Details****

When you run the ProductTester class, it will create six Product objects and display their details.

**PROJECT-2**

**SECTION-5:**

### Project Overview

The project involves modifying the ProductTester class to interact with the user, updating the Product class to include a Boolean field active, and enhancing the toString() method to display product information in a user-friendly format. Additionally, you will create a method to calculate the inventory value.

### Step-by-Step Solution

* **Product Class Modifications**

**Add a Boolean Instance Field**  
In the Product class, add a Boolean field named active with a default value of true.

* **PROGRAM:**

public class Product {

private int itemNumber;

private String name;

private int quantity;

private double price;

private boolean active = true;

// Constructor, Getters, and Setters

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

// Existing methods...

// Update toString method

@Override

public String toString() {

String status = active ? "Active" : "Discontinued";

return "Item Number: " + itemNumber + "\n" +

"Name: " + name + "\n" +

"Quantity in stock: " + quantity + "\n" +

"Price: " + price + "\n" +

"Stock Value: " + getInventoryValue() + "\n" +

"Product Status: " + status;

}

// Method to calculate inventory value

public double getInventoryValue() {

return price \* quantity;

}

}

**ProductTester Class Modification:**

* **Add Scanner for User Input**  
  Modify the ProductTester class to include a Scanner object and gather input from the user.
* **PROGRAM:**

import java.util.Scanner;

public class ProductTester {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

// Variables for product attributes

int tempNumber;

String tempName;

int tempQty;

double tempPrice;

// Input for p1

System.out.print("Enter item number: ");

tempNumber = in.nextInt();

in.nextLine(); // Clear the buffer

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

tempName = in.nextLine();

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

tempQty = in.nextInt();

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

tempPrice = in.nextDouble();

Product p1 = new Product(tempNumber, tempName, tempQty, tempPrice);

// Input for p2

in.nextLine(); // Clear the buffer

System.out.print("Enter item number: ");

tempNumber = in.nextInt();

in.nextLine(); // Clear the buffer

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

tempName = in.nextLine();

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

tempQty = in.nextInt();

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

tempPrice = in.nextDouble();

Product p2 = new Product(tempNumber, tempName, tempQty, tempPrice);

// Set p2 as discontinued

p2.setActive(false);

// Display p1 and p2 details

System.out.println(p1);

System.out.println(p2);

in.close();

}

}

**COMBAINED PROGRAM:**

import java.util.Scanner;

class Product {

private int itemNumber;

private String name;

private int quantity;

private double price;

private boolean active = true;

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

this.itemNumber = itemNumber;

this.name = name;

this.quantity = quantity;

this.price = price;

}

public int getItemNumber() {

return itemNumber;

}

public void setItemNumber(int itemNumber) {

this.itemNumber = itemNumber;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

public double getInventoryValue() {

return price \* quantity;

}

@Override

public String toString() {

String status = active ? "Active" : "Discontinued";

return "Item Number: " + itemNumber + "\n" +

"Name: " + name + "\n" +

"Quantity in stock: " + quantity + "\n" +

"Price: " + price + "\n" +

"Stock Value: " + getInventoryValue() + "\n" +

"Product Status: " + status;

}

}

public class ProductTester {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int tempNumber;

String tempName;

int tempQty;

double tempPrice;

System.out.print("Enter item number: ");

tempNumber = in.nextInt();

in.nextLine(); // Clear the buffer

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

tempName = in.nextLine();

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

tempQty = in.nextInt();

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

tempPrice = in.nextDouble();

Product p1 = new Product(tempNumber, tempName, tempQty, tempPrice);

in.nextLine(); // Clear the buffer

System.out.print("Enter item number: ");

tempNumber = in.nextInt();

in.nextLine(); // Clear the buffer

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

tempName = in.nextLine();

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

tempQty = in.nextInt();

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

tempPrice = in.nextDouble();

Product p2 = new Product(tempNumber, tempName, tempQty, tempPrice);

p2.setActive(false);

System.out.println("\nProduct 1 Details:");

System.out.println(p1);

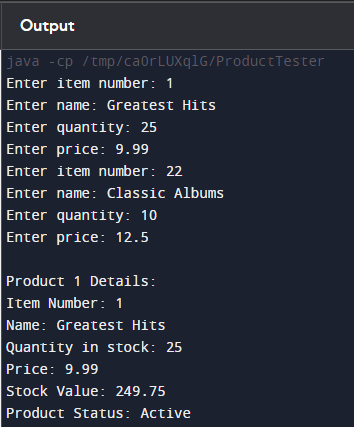
System.out.println("\nProduct 2 Details:");

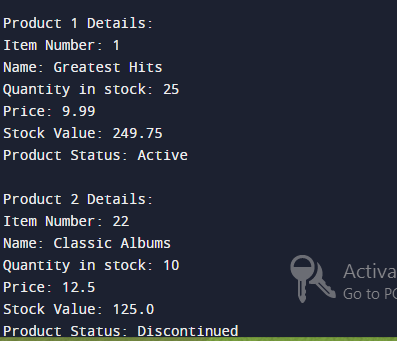
System.out.println(p2);

in.close();

}

}





**PROJECT-3**

**SECTION-6:**

Adding Products to the Inventory:

Prompt the User for Input

Open the inventory program that was updated in Section 5. Ask the user to enter the number of products they wish to add. Accept a positive integer and handle the value of zero

PROGRAME:

import java.util.Scanner;

public class ProductTester {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int maxSize = -1;

do {

try {

System.out.println("Enter the number of products you would like to add");

System.out.println("Enter 0 (zero) if you do not wish to add products");

maxSize = scanner.nextInt();

if (maxSize < 0) {

System.out.println("Incorrect Value entered");

}

} catch (Exception e) {

System.out.println(e);

System.out.println("Incorrect data type entered!");

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

}

} while (maxSize < 0);

if (maxSize == 0) {

System.out.println("No products required!");

} else {

Product[] products = new Product[maxSize];

for (int i = 0; i < maxSize; i++) {

scanner.nextLine(); // Clear the input buffer

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

String name = scanner.nextLine();

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

int quantity = scanner.nextInt();

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

double price = scanner.nextDouble();

System.out.print("Enter product item number: ");

int itemNumber = scanner.nextInt();

products[i] = new Product(name, quantity, price, itemNumber);

}

for (Product product : products) {

System.out.println(product);

}

}

scanner.close();

}

}

class Product {

private String name;

private int quantity;

private double price;

private int itemNumber;

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

this.name = name;

this.quantity = quantity;

this.price = price;

this.itemNumber = itemNumber;

}

@Override

public String toString() {

return "Product{" +

"name='" + name + '\'' +

", quantity=" + quantity +

", price=" + price +

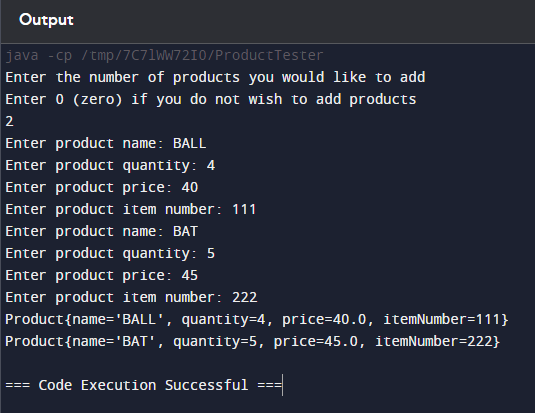
", itemNumber=" + itemNumber +

'}';

}

}

.**OUTPUT:**



**PROJECT-4**

**SECTION-7(PART-1):**

### Project Structure

plaintext

Copy code

inventory/

│

├── Product.java

└── ProductTester.java

### 1. ****Product Class****

This class represents a product in the inventory.

#### Product.java

java

Copy code

package inventory;

public class Product {

private String name;

private int quantity;

private double price;

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

this.name = name;

this.quantity = quantity;

this.price = price;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public double getPrice() {

return price;

}

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (quantity <= this.quantity) {

this.quantity -= quantity;

} else {

System.out.println("Insufficient stock to deduct!");

}

}

@Override

public String toString() {

return "Product [Name=" + name + ", Quantity=" + quantity + ", Price=" + price + "]";

}

}

### 2. ****ProductTester Class****

This class handles user interaction and manages the product inventory.

#### ProductTester.java

java

Copy code

package inventory;

import java.util.Scanner;

public class ProductTester {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Step 1: Get the number of products

int maxSize = getNumProducts(scanner);

// Step 2: Create an array to store products

Product[] products = new Product[maxSize];

// Step 3: Add products to inventory

addToInventory(products, scanner);

// Step 4: User interface menu

int choice;

do {

choice = getMenuOption(scanner);

switch (choice) {

case 1:

displayInventory(products);

break;

case 2:

addInventory(products, scanner);

break;

case 3:

deductInventory(products, scanner);

break;

case 4:

discontinueProduct(products, scanner);

break;

case 0:

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

break;

default:

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

}

} while (choice != 0);

scanner.close();

}

public static void displayInventory(Product[] products) {

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

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

if (products[i] != null) {

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

}

}

}

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

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

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

String name = scanner.nextLine();

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

int quantity = scanner.nextInt();

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

double price = scanner.nextDouble();

scanner.nextLine(); // Consume the newline

products[i] = new Product(name, quantity, price);

}

}

public static int getNumProducts(Scanner scanner) {

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

return scanner.nextInt();

}

public static int getMenuOption(Scanner scanner) {

int choice = -1;

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

System.out.println("Menu:");

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 {

choice = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a number between 0 and 4.");

}

}

return choice;

}

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

int productChoice = -1;

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

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

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

if (products[i] != null) {

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

}

}

try {

productChoice = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a valid product number.");

}

}

return productChoice;

}

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

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (updateValue < 0) {

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

try {

updateValue = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a positive number.");

}

}

products[productChoice].addToInventory(updateValue);

}

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

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

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

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

try {

updateValue = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

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

}

}

products[productChoice].deductFromInventory(updateValue);

}

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

int productChoice = getProductNumber(products, scanner);

products[productChoice] = null;

System.out.println("Product discontinued.");

}

}

### Project Explanation

* **Product Class**: Contains attributes (name, quantity, price) and methods to add/deduct inventory.
* **ProductTester Class**: Handles the program's main flow:
  + getNumProducts: Gets the number of products from the user.
  + addToInventory: Adds products to the array.
  + displayInventory: Displays the list of products.
  + getMenuOption: Displays a menu and returns the user’s choice.
  + getProductNumber: Gets a product number for updates.
  + addInventory: Adds stock to a selected product.
  + deductInventory: Deducts stock from a selected product.
  + discontinueProduct: Removes a product from the inventory.

**COMBAIN CODE:**

import java.util.Scanner;

public class InventoryProgram {

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 choice;

do {

choice = getMenuOption(scanner);

switch (choice) {

case 1:

displayInventory(products);

break;

case 2:

addInventory(products, scanner);

break;

case 3:

deductInventory(products, scanner);

break;

case 4:

discontinueProduct(products, scanner);

break;

case 0:

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

break;

default:

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

}

} while (choice != 0);

scanner.close();

}

public static void displayInventory(Product[] products) {

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

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

if (products[i] != null) {

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

}

}

}

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

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

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

String name = scanner.nextLine();

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

int quantity = scanner.nextInt();

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

double price = scanner.nextDouble();

scanner.nextLine(); // Consume the newline

products[i] = new Product(name, quantity, price);

}

}

public static int getNumProducts(Scanner scanner) {

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

return scanner.nextInt();

}

public static int getMenuOption(Scanner scanner) {

int choice = -1;

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

System.out.println("Menu:");

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 {

choice = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a number between 0 and 4.");

}

}

return choice;

}

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

int productChoice = -1;

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

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

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

if (products[i] != null) {

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

}

}

try {

productChoice = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a valid product number.");

}

}

return productChoice;

}

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

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

while (updateValue < 0) {

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

try {

updateValue = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input, please enter a positive number.");

}

}

products[productChoice].addToInventory(updateValue);

}

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

int productChoice = getProductNumber(products, scanner);

int updateValue = -1;

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

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

try {

updateValue = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

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

}

}

products[productChoice].deductFromInventory(updateValue);

}

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

int productChoice = getProductNumber(products, scanner);

products[productChoice] = null;

System.out.println("Product discontinued.");

}

// Inner Product Class

static class Product {

private String name;

private int quantity;

private double price;

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

this.name = name;

this.quantity = quantity;

this.price = price;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public double getPrice() {

return price;

}

public void addToInventory(int quantity) {

this.quantity += quantity;

}

public void deductFromInventory(int quantity) {

if (quantity <= this.quantity) {

this.quantity -= quantity;

} else {

System.out.println("Insufficient stock to deduct!");

}

}

@Override

public String toString() {

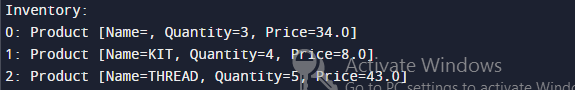
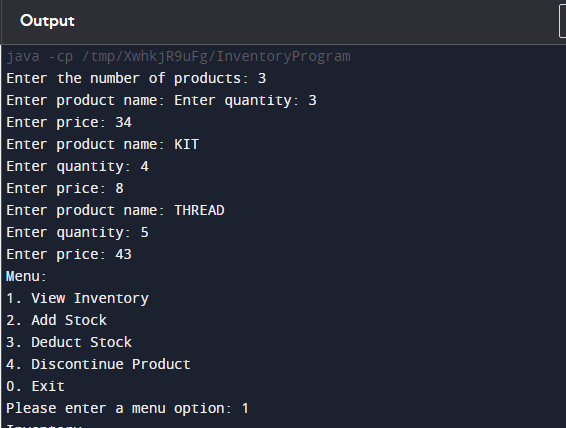
return "Product [Name=" + name + ", Quantity=" + quantity + ", Price=" + price + "]";

}

}

}

**OUTPUT:**



.

**PROJECT-5**

**SECTION-7(PART-2):**

### 1. ****DVD Sample Data Table****

| **Attribute** | **Sample Data** |
| --- | --- |
| Name of product | Daredevil |
| Price | 8.99 |
| Quantity in stock | 50 |
| Item number | 1 |
| Length (minutes) | 99 |
| Age Rating | 15 |
| Film Studio | 20th Century Fox |

### 2. ****CD Sample Data Table****

| **Attribute** | **Sample Data** |
| --- | --- |
| Name of product | Dreams we never lost |
| Price | 7.99 |
| Quantity in stock | 50 |
| Item number | 2 |
| Artist | Tidelines |
| Number of songs | 14 |
| Label | Tide Lines Music |

### 3. ****Implementation in Java****

#### ****Product Class****

This class should already exist from the previous sections, containing instance fields like name, price, quantity, and itemNumber.

java

Copy code

public class Product {

private String name;

private double price;

private int quantity;

private int itemNumber;

private boolean status; // Active or Inactive

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

this.name = name;

this.price = price;

this.quantity = quantity;

this.itemNumber = itemNumber;

this.status = true; // Assuming all products start as active

}

public double calculateInventoryValue() {

return price \* quantity;

}

@Override

public String toString() {

return "Item Number: " + itemNumber +

"\nName: " + name +

"\nQuantity in stock: " + quantity +

"\nPrice: " + price +

"\nStock Value: " + calculateInventoryValue() +

"\nProduct Status: " + (status ? "Active" : "Inactive");

}

// Getters and Setters for each field

}

#### ****DVD Subclass****

java

Copy code

public class DVD extends Product {

private int length;

private int ageRating;

private String filmStudio;

public DVD(String name, double price, int quantity, int itemNumber, int length, int ageRating, String filmStudio) {

super(name, price, quantity, itemNumber);

this.length = length;

this.ageRating = ageRating;

this.filmStudio = filmStudio;

}

@Override

public double calculateInventoryValue() {

return super.calculateInventoryValue() \* 1.05; // Adding 5% restocking fee

}

@Override

public String toString() {

return super.toString() +

"\nMovie Length: " + length +

"\nAge Rating: " + ageRating +

"\nFilm Studio: " + filmStudio;

}

// Getters and Setters for DVD specific fields

}

#### ****CD Subclass****

java

Copy code

public class CD extends Product {

private String artist;

private int numberOfSongs;

private String label;

public CD(String name, double price, int quantity, int itemNumber, String artist, int numberOfSongs, String label) {

super(name, price, quantity, itemNumber);

this.artist = artist;

this.numberOfSongs = numberOfSongs;

this.label = label;

}

@Override

public String toString() {

return super.toString() +

"\nArtist: " + artist +

"\nSongs on Album: " + numberOfSongs +

"\nRecord Label: " + label;

}

// Getters and Setters for CD specific fields

}

### 4. ****Modifying**** ProductTester ****Class****

#### ****Original Method:**** addToInventory

java

Copy code

public void addToInventory(Product[] inventory) {

// Existing code to add general products

}

#### ****New Method:**** addCDToInventory

java

Copy code

public void addCDToInventory(Product[] inventory) {

Scanner scanner = new Scanner(System.in);

System.out.print("Please enter the CD name: ");

String name = scanner.nextLine();

System.out.print("Please enter the price: ");

double price = scanner.nextDouble();

System.out.print("Please enter the quantity in stock: ");

int quantity = scanner.nextInt();

System.out.print("Please enter the item number: ");

int itemNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Please enter the artist: ");

String artist = scanner.nextLine();

System.out.print("Please enter the number of songs: ");

int numberOfSongs = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Please enter the record label: ");

String label = scanner.nextLine();

CD cd = new CD(name, price, quantity, itemNumber, artist, numberOfSongs, label);

// Add cd object to inventory array

}

**COMBAINED PROGRAME:**

import java.util.Scanner;

// Product class (Super class)

class Product {

private String name;

private double price;

private int quantity;

private int itemNumber;

private boolean status; // Active or Inactive

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

this.name = name;

this.price = price;

this.quantity = quantity;

this.itemNumber = itemNumber;

this.status = true; // Assuming all products start as active

}

public double calculateInventoryValue() {

return price \* quantity;

}

@Override

public String toString() {

return "Item Number: " + itemNumber +

"\nName: " + name +

"\nQuantity in stock: " + quantity +

"\nPrice: " + price +

"\nStock Value: " + calculateInventoryValue() +

"\nProduct Status: " + (status ? "Active" : "Inactive");

}

// Getters and Setters for each field

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public int getItemNumber() {

return itemNumber;

}

public void setItemNumber(int itemNumber) {

this.itemNumber = itemNumber;

}

public boolean isStatus() {

return status;

}

public void setStatus(boolean status) {

this.status = status;

}

}

// DVD subclass

class DVD extends Product {

private int length;

private int ageRating;

private String filmStudio;

public DVD(String name, double price, int quantity, int itemNumber, int length, int ageRating, String filmStudio) {

super(name, price, quantity, itemNumber);

this.length = length;

this.ageRating = ageRating;

this.filmStudio = filmStudio;

}

@Override

public double calculateInventoryValue() {

return super.calculateInventoryValue() \* 1.05; // Adding 5% restocking fee

}

@Override

public String toString() {

return super.toString() +

"\nMovie Length: " + length +

"\nAge Rating: " + ageRating +

"\nFilm Studio: " + filmStudio;

}

// Getters and Setters for DVD specific fields

public int getLength() {

return length;

}

public void setLength(int length) {

this.length = length;

}

public int getAgeRating() {

return ageRating;

}

public void setAgeRating(int ageRating) {

this.ageRating = ageRating;

}

public String getFilmStudio() {

return filmStudio;

}

public void setFilmStudio(String filmStudio) {

this.filmStudio = filmStudio;

}

}

// CD subclass

class CD extends Product {

private String artist;

private int numberOfSongs;

private String label;

public CD(String name, double price, int quantity, int itemNumber, String artist, int numberOfSongs, String label) {

super(name, price, quantity, itemNumber);

this.artist = artist;

this.numberOfSongs = numberOfSongs;

this.label = label;

}

@Override

public String toString() {

return super.toString() +

"\nArtist: " + artist +

"\nSongs on Album: " + numberOfSongs +

"\nRecord Label: " + label;

}

// Getters and Setters for CD specific fields

public String getArtist() {

return artist;

}

public void setArtist(String artist) {

this.artist = artist;

}

public int getNumberOfSongs() {

return numberOfSongs;

}

public void setNumberOfSongs(int numberOfSongs) {

this.numberOfSongs = numberOfSongs;

}

public String getLabel() {

return label;

}

public void setLabel(String label) {

this.label = label;

}

}

// ProductTester class

public class ProductTester {

public static void main(String[] args) {

// Array to store products

Product[] inventory = new Product[2];

// Add a DVD to inventory

inventory[0] = addDVDToInventory();

// Add a CD to inventory

inventory[1] = addCDToInventory();

// Display the inventory

for (Product product : inventory) {

System.out.println(product);

System.out.println();

}

}

// Method to add DVD to inventory

public static DVD addDVDToInventory() {

Scanner scanner = new Scanner(System.in);

System.out.print("Please enter the DVD name: ");

String name = scanner.nextLine();

System.out.print("Please enter the price: ");

double price = scanner.nextDouble();

System.out.print("Please enter the quantity in stock: ");

int quantity = scanner.nextInt();

System.out.print("Please enter the item number: ");

int itemNumber = scanner.nextInt();

System.out.print("Please enter the movie length (in minutes): ");

int length = scanner.nextInt();

System.out.print("Please enter the age rating: ");

int ageRating = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Please enter the film studio: ");

String filmStudio = scanner.nextLine();

return new DVD(name, price, quantity, itemNumber, length, ageRating, filmStudio);

}

// Method to add CD to inventory

public static CD addCDToInventory() {

Scanner scanner = new Scanner(System.in);

System.out.print("Please enter the CD name: ");

String name = scanner.nextLine();

System.out.print("Please enter the price: ");

double price = scanner.nextDouble();

System.out.print("Please enter the quantity in stock: ");

int quantity = scanner.nextInt();

System.out.print("Please enter the item number: ");

int itemNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Please enter the artist: ");

String artist = scanner.nextLine();

System.out.print("Please enter the number of songs: ");

int numberOfSongs = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Please enter the record label: ");

String label = scanner.nextLine();

return new CD(name, price, quantity, itemNumber, artist, numberOfSongs, label);

}

}

**OUTPUT:**

