Java Fundamentals
Section 7 Part 2: Creating an Inventory Project
Project

Overview

This project will progress with you throughout Sections 4, 5, 6, and 7 of the course. After each section there will be more to add until it
builds into a complete Java application to maintain Inventory. For each part, build upon the last part so that both the old and new
requirements are met. Include all parts in a package called inventory.
Create an inventory program that can be used for a range of different products (cds, dvds, software, etc.).
Topic(s):
• Working with subclasses (Section 7.4)
 using extends
 using super()
 Overriding methods from a superclass
• Updating the user interface (Section 4.2)
Scenario
Word has spread about your inventory software and you have been approached by a company that sells exclusively CDs and DVDs.
They would like you to customise your software so that it can store the length, age rating and film studio for DVDs as well as artist,
number of songs and label for their CDs.
1. Open the inventory program that was updated in Section 7 Part 1: Creating an inventory Project
a. Before you begin with your programming it can be useful to update your design tables. The following tables show
what fields belong to the Product class ( ) and what fields are specific to a DVD ( ).
i. Complete a sample Data table for a list of DVD movies of your choice.
DVD
Attribute Sample Data
Name of the product. Daredevil
Price. 8.99
Quantity of units in stock. 50
Item number. 1
Length (minutes) 99
Age Rating 15
Film Studio 20th Century Fox

ii. Complete a sample Data table for a list of CDs of your choice.
CD
Attribute Sample Data
Name of the product. Dreams we never lost
Price. 7.99

Quantity of units in stock. 50
Item number. 2
Artist Tidelines
Number of songs 14
label Tide Lines Music
2. You are going to implement inheritance by creating subclasses of the Product class.
a. Create a subclass of the Product class called DVD that has additional instance fields to store the movie length in
minutes, the age rating and the Film Studio that released the movie.
b. Create a single constructor that accepts values for every instance field for both the DVD and Product classes. Use
the super() call to the constructor in Product passing the required parameters.
c. Create getters and setters for the DVD instance fields.
d. Follow exactly the same process to create a subclass of Product named CD. Create the instance fields, constructor
and getter and setters.
3. In the DVD subclass, override the method to calculate the value of the inventory of a DVD with the same name as that method
previously created in the product class. The DVD subclass method should also add a 5% restocking fee to the value of the
inventory of that product. You will need to get the values for price and quantity in stock from the superclass.
4. You are now going to define the output for both the DVD and CD objects.
a. Override the toString() method from the Product class so that all information about new subclass objects (DVDs) can
be printed to the output console. Use the getter methods for the values instead of referencing the fields directly, this
enforces using the local version of the methods if they exist instead of the methods at the superclass. Your output
should look like the following:
Item Number : 1
Name : Daredevil
Movie Length : 99
Age Rating : 15
Film Studio : 20th Century Fox
Quantity in stock: 50
Price : 8.99
Stock Value : 471.975
Product Status : Active
b. Do the same in the CD class so that you produce the following output:
Item Number : 2
Name : Dreams we never lost
Artist : Tidelines
Songs on Album : 14
Record label : Tide Lines Music
Quantity in stock: 50
Price : 7.99
Stock Value : 399.5
Product Status : Active
5. Modify the ProductTester class so that you populate the products array with either

CD or DVD objects.
a. Copy the addToInventory method and paste it directly under the original. Rename the method to addCDToInventory.
b. Add additional temporary variables to include the instance fields that you added to the CD class.
c. Update the prompts to ask the user for the information in the following order:

To complete this project, we'll go step-by-step to create and update the inventory program as described. Here's how we can approach each part of the project:

Step 1: Design Tables for DVD and CD

DVD Sample Data Table
Attribute	Sample Data
Name	Daredevil
Price	8.99
Quantity	50
Item Number	1
Length (minutes)	99
Age Rating	15
Film Studio	20th Century Fox

CD Sample Data Table
Attribute	Sample Data
Name	Dreams we never lost
Price	7.99
Quantity	50
Item Number	2
Artist	Tidelines
Number of songs	14
Label	Tide Lines Music

Step 2: Create Subclasses DVD and CD

Product Class (Superclass)
```java
public class Product {
    private String name;
    private double price;
    private int quantity;
    private int itemNumber;
    private String status;
```

```
    public Product(String name, double price, int quantity, int itemNumber) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
        this.itemNumber = itemNumber;
        this.status = "Active";
    }

    // Getters and setters for Product fields
    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 String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }

    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;
    }
}
```

DVD Class (Subclass)
```java
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;
    }
```

```
    // Getters and setters for DVD 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; }

    @Override
    public double calculateInventoryValue() {
        return super.calculateInventoryValue() * 1.05; // Add 5% restocking fee
    }

    @Override
    public String toString() {
        return "Item Number: " + getItemNumber() +
                "\nName: " + getName() +
                "\nMovie Length: " + length +
                "\nAge Rating: " + ageRating +
                "\nFilm Studio: " + filmStudio +
                "\nQuantity in stock: " + getQuantity() +
                "\nPrice: " + getPrice() +
                "\nStock Value: " + calculateInventoryValue() +
                "\nProduct Status: " + getStatus();
    }
}
```

CD Class (Subclass)
```java
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;
    }

    // Getters and setters for CD 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; }

    @Override
    public String toString() {
        return "Item Number: " + getItemNumber() +
                "\nName: " + getName() +
                "\nArtist: " + artist +
                "\nSongs on Album: " + numberOfSongs +
                "\nRecord label: " + label +
                "\nQuantity in stock: " + getQuantity() +
                "\nPrice: " + getPrice() +
                "\nStock Value: " + calculateInventoryValue() +
                "\nProduct Status: " + getStatus();
    }
}
```

Step 3: Update ProductTester Class

ProductTester Class
```java
import java.util.Scanner;

public class ProductTester {
    private Product[] products;
    private int productCount;

    public ProductTester(int size) {
        products = new Product[size];
        productCount = 0;
    }

    public void addToInventory(String name, double price, int quantity, int
itemNumber) {
        products[productCount++] = new Product(name, price, quantity, itemNumber);
    }

    public void addDVDToInventory(String name, double price, int quantity, int
itemNumber, int length, int ageRating, String filmStudio) {
        products[productCount++] = new DVD(name, price, quantity, itemNumber,
length, ageRating, filmStudio);
    }

    public void addCDToInventory(String name, double price, int quantity, int
itemNumber, String artist, int numberOfSongs, String label) {
        products[productCount++] = new CD(name, price, quantity, itemNumber, artist,
numberOfSongs, label);
    }
```

```
    public void displayInventory() {
        for (int i = 0; i < productCount; i++) {
            System.out.println(products[i]);
            System.out.println();
        }
    }

    public static void main(String[] args) {
        ProductTester inventory = new ProductTester(10);
        Scanner scanner = new Scanner(System.in);

        // Sample data for testing
        inventory.addDVDToInventory("Daredevil", 8.99, 50, 1, 99, 15, "20th Century
Fox");
        inventory.addCDToInventory("Dreams we never lost", 7.99, 50, 2, "Tidelines",
14, "Tide Lines Music");

        inventory.displayInventory();
    }
}
```

Explanation

- **Product Class**: The superclass `Product` contains common attributes and methods for all products.
- **DVD Class**: The `DVD` subclass extends `Product`, adding specific attributes for DVDs and overriding methods where necessary.
- **CD Class**: The `CD` subclass extends `Product`, adding specific attributes for CDs and overriding methods where necessary.
- **ProductTester Class**: The main class for testing and demonstrating the inventory system. It creates instances of `DVD` and `CD` and displays their details.

This approach ensures that we are adhering to the principles of inheritance and polymorphism while creating a flexible and extendable inventory system.