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:

```
Please enter the CD name:
```

```
Please enter the artist name:
Please enter the record label name:
Please enter the number of songs:
Please enter the quantity of stock for this product:
Please enter the price for this product:
Please enter the item number:
```

You may have to clear the input buffer before you ask for any values.

d. The last line in the method creates a product object in the products array, you will need to update this.
   i. Currently the position in the array is identified with the variable i. As this method is no longer in the for loop it does not recognise i as a variable. To resolve this add i as a parameter at the end of the method signature.

   ii. You will now use polymorphism to create a CD object instead of a generic product object. Update the code to identify CD as the object type after the **= new** statement.

   iii. You get an error when you do this as the argument list does not match the parameter list. Update the parameters so that they match the parameter list for the CD constructor.

e. Follow the same process to create a DVD object. The prompts should be displayed in the following order:

```
Please enter the DVD name:
Please enter the film studio name:
Please enter the age rating:
Please enter the length in minutes:
Please enter the quantity of stock for this product:
Please enter the price for this product:
Please enter the item number:
```

6. You now need to update the code in the addInventory method to allow the user to select to add a CD or DVD.

   a. Remove all of the existing local variables and replace them with a single variable named stockChoice that will store an integer value and be initialized to minus 1.

   b. Prompt the user for a value for stockChoice by allowing them to choose between CD or DVD objects by displaying the following menu:

```
1: CD
2: DVD
Please enter the product type:
```

   c. If the user enters an integer value less than 1 or greater than 2 then the following error message should be displayed:

```
Only numbers 1 or 2 allowed!
```

   d. The user should be re-prompted until a valid input is provided and all incorrect input should be handled.

   e. After the while statement add an appropriate condition statement that will allow the user to ass a CD if they enter 1 or a DVD if they enter 2. Use the appropriate add method that you have previously created.

7. Run and test your code.

8. Do not allow the user to add stock to a discontinued product line.

   Update the addToInventory method in the **Product** class to stop the adding of stock to a discontinued line.

9. Run and test your code.

10. Save your project.