CS1083 Assignment # 3 - Fall 2024

Due: Wednesday, 2 October before 4:30 pm in the Desire2Learn dropbox. (See submission instructions below).

The purpose of this lab is:

- Implement Comparable
- Use Comparable to perform searching

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during an extra help session. If your question is not answered during an extra help session, you may contact your course professor.

Background Scenario

You are hired to create a tracking system for a vintage music store that sells three different types of goods: Records, Cassettes, and the occasional DVD. You are to keep in mind efficiency and code reusability when completing the following assignment.

Purchasable Interface

This interface should be applied to all goods sold at the store. Any purchasable item should have the following behaviours:

1. It can access the title of the item
2. It can calculate the selling price of the item

Records and Cassettes

Records and Cassettes are both Audio Items *share* many attributes and behaviours. Both should store the title, the artist, the initial price, and the year of release. Only one difference exists between records and cassettes:

> The selling price of a record = (initial price) * ((2024 - year) / 4.0) while the selling price of a cassette = (initial price) + ((initial price) / ((2024 - year) / 6.0)).

A toString() should exist and is the same for both which returns a String in the following format (cost should be selling price):

> Title (Artist) Cost: $XX.XX

DVD

Since the store plans to sell very few DVDs, the only information they wish to store is the title and the selling price (for DVDs, the initial and selling price are the same). Accessors must be available for each as does a toString() method which prints in the following format:

Title Cost: $XX.XX

CompareTo

All items are compared based on their title first (alphabetic) and then their selling price (lowest to highest). Should return 0 if they are equal, 1 if **`this`** is greater than the parameter object and -1 if **`this`** is less than the parameter object. The search method described below **must** use compareTo.

Catalogue

Next, you must write a catalogue class. This class should have only 2 instance variables: (1) an ArrayList that stores all items for sale and (2) the amount of money the store has. The only parameter in the constructor should be the amount of money the store is starting out with as they should not have any items. As the store buys or sells items, the amount of money they have should increase or decrease appropriately. The following methods should be present:

1. **`sellItem(Item i)`** – The store sells this item and gains profit. Return false if the item is not in the store. This should use the search method below.
2. **`buyItem(Item i)`** – The store buys this item and loses profit – should buy at initial price, not selling price. Return false if the store does not enough money to buy the item.
3. **`searchItemLinear(Item i)`** – search the store for the item, return the index if the item is in the catalogue or -1 if not. This must use **linear search and you must write it yourself**.
4. **`printCatalogue(Item i)`** – prints all items in the catalogue.

5. **`searchItemBinary(Item i)`** – search the store for the item, return the index if the item is in the catalogue or -1 if not. This must use **binary search and you must write it yourself**.

Driver Program

Write a driver to test the classes you have written. Make sure your driver meets all the required test cases below:

1. Add 5 items to the catalogue.
    a. Some should share titles, but with different prices. Should be at least one DVD, Record, and Cassette.
2. Remove items from a catalogue until it is empty.
3. Try to remove an item that is not in the catalogue.
4. Try to add an item to the catalogue when the store does not have enough money to purchase it.
5. Print the catalogue.

Your electronic submission (submitted via Desire2Learn) will consist of two files. Name your files YourName-fileName.extension, e.g. JohnSmith-as3.zip, JohnSmith-as3.pdf:

1. A single pdf file containing a listing of the source code for the **`Purchasable`** interface, as well as any other classes you have written. Also include your **driver.**
2. A zip file containing all your Java classes.