Due Date: 2/16/2024, 11:59pm

CMP_SC 3330 – Object-oriented Programming
Homework 3

This assignment aims to implement an inventory management system for a media product shop using Java Object-Oriented Programming principles. The assignment involves designing classes for different types of media products and implementing a singleton class for managing the shop's inventory.

Class Definition:

MediaProduct Class:

- Implement a base class ***MediaProduct*** with protected attributes/fields *title*(**String**), *price*(**double**), year(**int**) and *genre*(**Genre**). The *genre* should be represented using an enum.
- Create subclasses ***VinylRecordProduct***, ***CDRecordProduct***, and ***TapeRecordProduct***, each representing a different type of media product. Ensure these classes inherit from ***MediaProduct*** and contain a constructor that uses the super keyword to initialize the attributes.
- Implement copy constructors for each media product to prevent information leaks.

Genre Enum:

- The genre are:
  - ROCK,
  - POP,
  - JAZZ,
  - CLASSICAL,
  - HIP_HOP,
  - ELECTRONIC,
  - CHILDREN

StockManagerSingleton Class:

- Implement a singleton class ***StockManagerSingleton*** for managing the inventory of the shop.
- The ***StockManagerSingleton*** class should read the initial inventory from a CSV file (***inventory.csv***) during initialization, update existing items, add new items, remove items, and save the updated inventory back to the CSV file.
- The ***StockManagerSingleton*** class should have a String field called *inventoryFilePath,* which is initialized to the relative path to the ***inventory.csv*** file, including the file name. However, make sure that the field does not leak any information and is closed for modification.

Program Requirements:

Stock Management Methods:

- `public boolean initializeStock():`
  - Reads the initial inventory data from a CSV file located at *inventoryFilePath*. (**Hint:** Consider using the split() method for tokenization.)
  - Parses the CSV file to create media product objects and adds them to the inventory.

    - Returns **true** if the initialization is successful, **false** otherwise (file does not exist, or file empty).
- `public boolean updateItemPrice(MediaProduct product, double newPrice):`
    - Updates the price of the given media product to the newPrice.
    - Returns **true** if the update is successful, **false** otherwise
- `public boolean addItem(MediaProduct product):`
    - Adds a new media product to the inventory.
    - Returns **true** if the addition is successful, **false** otherwise.
- `public boolean removeItem(MediaProduct product):`
    - Removes the given media product from the inventory.
    - Returns **true** if the removal is successful, **false** otherwise.
- `public boolean saveStock():`
    - Saves the updated inventory back to the CSV file located at *inventoryFilePath*.
    - Overwrites the existing file with the updated inventory data.
    - Returns **true** if the saving is successful, **false** otherwise (file does not exist, or file empty).
- `public ArrayList<MediaProduct> getMediaProductBelowPrice(int maxPrice):`
    - Gets the media products that are below the given *maxPrice.*
    - This creates a new ArrayList of media products that is below the *maxPrice*. Beware of not leaking any information.
- `public void printListOfMediaProduct(ArrayList<MediaProduct> productList):`
    - Prints the given media product list.
- `public ArrayList<VinylRecordProduct> getVinylRecordList(ArrayList<MediaProduct> productList):`
    - Gets the media products as an ArrayList.
    - This creates a new ArrayList of VinylRecordProduct that filters the vinyl records and returns the ArrayList. Beware of not leaking any information.
- `public ArrayList<CDRecordProduct> getCDRecordsList(ArrayList<MediaProduct> productList):`
    - Gets the media products as an ArrayList.
    - This creates a new ArrayList of CDRecordProduct that filters the CD records and returns the ArrayList. Beware of not leaking any information.
- `public ArrayList<TapeRecordProduct> getTapeRecordList(ArrayList<MediaProduct> productList):`
    - Gets the media products as an ArrayList.
    - This creates a new ArrayList of TapeRecordProduct that filters the tape records and returns the ArrayList. Beware of not leaking any information.

Submission Guidelines:

- Each team is required to create a GitHub repository for the project.

- The repository should include all the required Java files (Main.java, MediaProduct.java, VinylRecordProduct.java, CDRecordProduct.java, TapeRecordProduct, Genre.java, and StockManagerSingleton.java) and any other necessary files to run the program.
- Team members are expected to contribute equally to the project.
- Each team member should make meaningful contributions, and commit messages must be descriptive and related to the changes made. Your grades will be affected by your commits.
- The GitHub repository should demonstrate good version control practices, with commits logically organized and documenting the evolution of the code.
- Make sure to include a README.md file providing clear instructions on how to run the program, any dependencies, and a brief explanation of the project.
- Verify that the repository is accessible and properly organized, allowing anyone to clone and run the program without additional configuration.
- Your program must use the classes with described methods, given prototypes and signatures exactly. You are allowed to implement additional helper methods and classes.
- Late submission between 0hrs < late <= 24hrs will lose half of the grade. After 24 hours, submissions will receive a grade of 0 for the assignment.
- **Not following the submission guidelines will result in a penalty on your grades.**

Note:

- Ensure that your program handles cases where the file is not found or if there are any issues during file reading.
- Make use of the concepts you've learned, such as constructors, getter/setter methods, static fields/methods, and the toString() method.
- Test your program with different scenarios, including cases where the object is not found and the update is unsuccessful.