CMSC131 Fall 2021 (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/)

Project #8, Media Rental Manager (Due Thu, Dec 2, 11:55 pm / Sun, Dec 5, 11:55 pm)

Objectives

To practice design and inheritance.

For this project you can discuss the project with classmates, but you may not exchange code nor write code together. Your goal is to learn so you can be successful in internships and class exams. You do not need to include the names of people your discuss the project with.

This project has been used in previous installments of the course. Some students may find implementations for this project online. If you use any such implementation you will be violating academic integrity rules and an appropriate academic case will be submitted which may result in an XF in the course.

Overview

To implement a simplified version of a system that allows people to rent movies and music albums (similar to Netflix) and receive them via mail. For this project you will define all the classes/interfaces needed (yes it is your design :)).

This project has two deadlines:

- **Thu, Dec 2, 11:55 pm** - Your code must pass the first 2 public tests. This is the only requirement for this deadline. We will not grade the code for style. This first part is worth **0.5%** of your course grade (NOT **0.5%** of this project grade). You can still submit late for this part.
- **Sun, Dec 5, 11:55 pm** - Final deadline for the project. You can still submit late (as usual).

For this project you don't need to use inheritance (using Java extends keyword). Actually your final exam will not include extending one class from another and concepts related to that process (additional information about the exam will be posted soon). For this project what you need are classes and interfaces. For those classes that are related, you can define an interface that both classes implement. For example, in a Game system you can have an interface called **Game** and classes **Tetris** and **Checkers** that implement the Game interface. Now you can create an **ArrayList<Game>** and combine both Tetris and Checkers objects in the ArrayList. To tell which kind actual object you have, you use instanceof (see instanceof examples in the lecture code). A Game variable (e.g., Game g) is polymorphic, as it can have the address of a Tetris object or a Checkers object. Only Game methods can be called by using g. If you know that g actually points to a Tetris object (which you can determine using instanceof), you can cast g to a Tetris object and call methods that belong to the Tetris class. It is OK to use inheritance for this project.

Using Collections.sort when classes implemennt a common interface can be confusing at first. The following example illustrates how we can sort games GameSortingExample.zip (GameSortingExample.zip).

Grading

- (65%) Public Tests

- (10%) Release Tests
- (15%) Design that includes at least three classes / interfaces (not including the MediaRentalManager class nor the MediaRentalManagerInt interface)
- (10%) ArrayList Requirement

Code Distribution

The project's code distribution is available at MediaRentalManager.zip (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/prot/projects/zipFiles/MediaRentalManager.zip). Download it and import it as you have imported our class examples. The code distribution provides you with the following:

- A package named **mediaRentalManager** - In this package you will provide the implementation for the system. We have provided an interface(MediaRentalManagerInt (doc/mediaRentalManager/MediaRentalManagerInt.html)) that specifies the functionality your system is expected to implement. The complete javadoc for the project can be found at javadoc (doc/index.html).
- A package named **tests** - Includes the public tests and the shell file for your student tests.

Specifications

You must define a class named **MediaRentalManager** (we did not provide it) that implements the **MediaRentalManagerInt** interface functionality. You must define classes that support the functionality specified by the interface. The following specifications are associated with the project:

1. You should study the public tests and the output (files with .txt extension under the expectedResults folder) to familiarize yourself with the expected system functionality.
2. Define a class named **MediaRentalManager**. Feel free to add any instance variables you understand are needed or any private methods. Do not add any public methods (beyond the ones specified in the MediaRentalManagerInt interface). Defining static constants is fine, but static variables could make the public/release tests fail as each test will remember previous results.
3. The media rental system keeps track of customers and media (movies and music albums). A customer has a name, address, a plan and two lists (queues). One queue represent the media the customer is interested in receiving and the second one represents the media already received (rented) by the customer. There are two plans a customer can have: UNLIMITED and LIMITED. UNLIMITED allows a customer to receive as many media as they want; LIMITED restricts the media to a default value of 2 (this value can be change via a manager method). A movie has a title, a number of copies available and a rating (e.g., "PG"). An album has a title, number of copies available, an artist and the songs that are part of the album.
4. You must define and use at least three classes / interfaces (not including MediaRentalManager class nor the MediaRentalManagerInt interface) as part of your design. **These clases must support the functionality of the system, otherwise you will not receive any credit.**
5. The database for your system needs to be represented using two ArrayList objects. One ArrayList will represent the customers present in the database; the second will represent the media (movies and albums). This is the ArrayList requirement provided in the **Grading** section.
6. You don't need to provide any javadoc nor documentation for the classes/methods you will provide.
7. Regarding the searchMedia method: the songs parameter represents a substring (fragment) or the full list of songs associated with the album. If the full list is provided, you can assume commas will be part of the string. **Hint:** you may want to consider using the indexOf method of the String class.
8. Feel free to use Collections.sort to sort your data.
9. **Although you are not require to write student tests you are encourage to do so.** Your student tests will not be graded.
10. Do not use Java maps nor sets in this project.

11. Not all the details associated with the project can be fully specified in this description. The sooner you start working on the project, the sooner you will be able to address any doubts you may have.
12. **Do not implement your own sorting algorithm.**
13. **Please do not post any information in Piazza about which classes/interfaces you are defining. Designing the solution is very important for this project (one of the main goals).**

Style

Verify your project satisfies style requirements as defined at Java Style Guide (http://www.cs.umd.edu/~nelson/classes/resources/javastyleguide/).

Submission

Submit your project as usual.

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the University's Office of Student Conduct (https://www.studentconduct.umd.edu/).

Web Accessibility (https://www.umd.edu/web-accessibility/)