Coursework: SCUPI+, A Java Application for Film Query

CS 04450 Data Structure, Department of Computer Science, SCUPI

Spring 2024

This coursework sheet explains the work in details. Please read the instructions carefully and follow them step-by-step. For submission instructions, please read the Sec. 4. If you have any queries regarding the understanding of the coursework sheet, please contact the TAs or the course leader. **Due on: 23:59 PM, Wednesday, June 5th.**

1 Introduction

A developer of a new Java application has asked for your help in storing a large amount of film data efficiently. The application, called *SCUPI+*, is used to present data and fun facts about films, the cast and crew who worked on them, and some ratings the developer has gathered in there free time. However, because the developer hasn't taken the module, they don't want to design how the data is stored.

Therefore, this coursework and the task that the developer has left to you, is to design one or more data structures that can efficiently store and search through the data. The data consists of 3 separate files:

- Movie Metadata: the data about the films, including there ID number, title, length, overview etc.
- Credits: the data about who stared in and produced the films.
- Ratings: the data about what different users thought about the films (rated out of 5 stars), and when the user rated the film.

To help out, the developer of *SCUPI+* has provided classes for each of these. Each class has been populated with functions with JavaDoc preambles that need to be filled in by you. As well as this, the developer has also tried to implement the `MyArrayList` data structure into a 4th dataset (called `Keywords`), to show you where to store your data structures and how they can be incorporated into the pre-made classes. Finally, the developer has left instructions for you, which include how to build, run and test you code; and the file structure of the application (see Sec. 3).

Therefore, your task is to implement the functions within the `Movies`, `Credits` and `Ratings` classes through the use of your own data structures.

2 Guidance

First, don't panic! Have a read through the documentation provided in Sec. 3. This explains how to build and run the application. This can be done without writing anything, so make sure you can do that first.

Then you can have a look at the comments and functions found in the `Movies`, `Credits` and `Ratings` classes. The location of these is described in Sec. 3.5.2. Each of the functions you need to implement has a comment above it, describing what it should do. It also lists each of the parameters for the function (lines starting with `@param`), and what the function should return (lines starting with `@return`).

When you are ready to start coding, We would recommend starting off with the `Rating` class first. This is because it is smallest of the 3 required, and is also one of the simplest. When you have completed a function, you can test it using the test suit described in Sec. 3.5.3. More details about where the code for the tests are can be found in Sec. 3.4.

3 SCUPI+

SCUPI+ is a small Java application that pulls in data from a collection of Comma Separated Value (CSV) files. It is designed to have a lightweight user interface (UI), so that users can inspect and query the data. The application also has a testing suit connected to it, to ensure all the functions work as expected. The functions called in the *SCUPI+* UI are the same as those called in the testing, so if the tests work, the UI will also work.

3.1 Required Software

For the *SCUPI+* to compile and run, Java 21 is required, make sure you download this specific version of Java. Whilst a newer version of Java can be utilised, other parts of the application will also have to be updated and this has not been tested. Although you can always have a try with your own version, it is highly recommended you download and use Java 21.

3.2 Building SCUPI+

To compile the code, simply run the command shown in the table below in the working directory (the one with `src` folder in it).

Linux/DCS System	**MacOS**	**Windows**
`./gradlew build`	`./gradlew build`	`./gradlew.bat build`

3.3 Running the SCUPI+ Application

To run the application, simply run the command shown in the table below in the working directory (the one with `src` folder in it).

Linux/DCS System	**MacOS**	**Windows**
`./gradlew run`	`./gradlew run`	`./gradlew.bat run`

This command will also compile the code, in case any files have been changed. When this is done, a window will appear with the UI for the application. The terminal will not be able to be used at this time. Instead it will print anything required from the program. To stop the application, simply close the window or press `CTRL+C` at the same time in the terminal.

3.4 Running the SCUPI+ Test Suit

To run the tests, simply run the command shown in the table below in the working directory (the one with `src` folder in it).

Linux/DCS System	**MacOS**	**Windows**
`./gradlew test`	`./gradlew test`	`./gradlew.bat test`

This command will also compile the code, in case any files have been changed. When ran, this will produce the output from each test function. It will also produce a webpage of the results, which can be found in `build/reports/tests/test/index.html`

3.5 SCUPI+ File Structure

Every effort has been made to keep the file structure simple and clean, whilst maintaining good coding practices. In the following subsections, a brief description of each of the key directories is given, along with its contents and what you need to worry about in them.

3.5.1 `data/`

This directory stores all the data files that are pulled into the application. There are 4 .csv files in this directory, 1 for each of the datasets described in Sec. 1. Each line in these files is a different entry, with values being separated by commas (hence the name Comma Separated Values). You do not need to add, edit or remove anything from this directory for your coursework. More details on how these files are structured can be found in Sec. 3.6.

3.5.2 `src/main/`

This directory stores all the Java code for the application. As such, there are a number of directories and files in this directory, each of which are required for the application and/or the UI to function. To make things simpler, there are 3 key directories that will be useful for you:

- `java/interfaces/`: stores the interface classes for the data sets. You do not need to add, edit or remove anything from this directory, but it may be useful to read through.
- `java/stores/`: stores the classes for the data sets. This is where the `Keywords`, `Movies`, `Credits` and `Ratings` from Sec. 1 are located, the latter 3 of which are the classes you need to complete. Therefore, you should only need to edit the following files:
  - `Movies.java`: stores and queries all the data about the films. The code in this file relies on the `Company` and `Genre` classes.
  - `Credits.java`: stores and queries all the data about who stared in and worked on the films. The code in this file relies on the `CastCredit`, `CrewCredit` and `Person` classes.
  - `Ratings.java`: stores and queries all the data about the ratings given to films.
- `java/structures/`: stores the classes for your data structures. As an example, a array list `MyArrayList` has been provided there. Any classes you add in here can be accessed by the classes in the stores directory (assuming the classes you add are public). You may add any files you wish to this directory, but `MyArrayList.java` and `IList.java` <u>should not be altered or removed</u>, as these are relied on for `Keywords`.

3.5.3 `src/test/`

This directory stores all the code that related solely to the JUnit tests. As such, there is a Java file for each of the stores you need to implement. You do not need to add, edit or remove anything from this directory for your coursework.

3.6 Data used for SCUPI+

All of the data used by the *SCUPI+* application can be found in the `data` directory. Each file in this directory contains a large collection of values, separated by commas (hence the CSV file type). Therefore, each of these can be opened by your favourite spreadsheet program. Most of these values are integers or floating point values, but some are strings. In the cases of strings, double quotation marks (") are used at the beginning and end of the value. Where multiple elements could exist in that value, a JSON object has been used. You do not need to parse these files, *SCUPI+* will do that for you in the `LoadData` class. The data generated by the `LoadData` class is passed to the corresponding data store class (`Movies`, `Credits`, `Ratings` and `Keywords`) using the `add` function.

To make development easier, we have provided only 1000 films present in the data. This means that there are 1000 entries in the credits data set, and 1000 entries in the keywords data set. However, some films may not have any cast and/or crew (that information may not have been released yet, or it is unknown), some films don't have keywords and some films may not have ratings. In these cases, an empty list of the required classes will be provided the `add` function.

3.6.1 Key Stats

Films		1000
Credits	Film Entries	1000
	Unique Cast	11483
	Unique Crew	9256
Ratings		17625
Keywords	Film Entires	1000
	Unique Keywords	2159

3.6.2 Movies Metadata

The following is a list all of the data stored about a film using the column names from the CSV file, in the same order they are in the CSV file. Blue fields are ones that are added through the `add` function in the `Movies` class.

- `adult`: a boolean representing whether the film is an adult film.
- `belongs_to_collection`: a JSON object that stores all the details about the collection a film is part of. This is added to the film using the `addToCollection` function in the `Movies` class. If the film is part of a collection, the collection will contain a collection ID, a collection name, a poster URL related to the collection and a backdrop URL related to the collection.
- `budget`: a long integer that stores the budget of the film in US Dollars. If the budget is not known, then the budget is set to 0. Therefore, this will always be greater than or equal to 0.
- `genres`: a JSON list that contain all the genres the films is part of. Each genre is represented as a key-value pair, where the key is represented as an ID number, and the value is represented as a string. *SCUPI+* passes this as an array of `Genre` objects.

- `homepage`: a string representing a URL of the homepage of the film. If the film has no homepage, then this string is left empty.
- `tmdb_id`: an integer representing the ID of the film. This is used to link this film to other pieces of data in other data sets.
- `imdb_id`: a string representing the unique part of the IMDb URL for a given film. This is added using the `setIMDB` function in the `Movies` class.
- `original_language`: a 2-character string representing the ISO 639 language that the film was originally produced in.
- `original_title`: a string representing the original title of the film. This may be the same as the `title` field, but is not always the case.
- `overview`: a string representing the an overview of the film.
- `popularity`: a floating point value that represents the relative popularity of the film. This value is always greater than or equal to 0. This data is added by the `setPopularity` function in the `Movies` class.
- `poster_path`: a string representing the unique part of a URL for the film poster. Not all films have a poster available. In these cases, an empty string is given.
- `production_companies`: a JSON list that stores the production countries for a film. Each entry in the JSON list has a key value pair, where the key is the ID of the company, and the value is the name of the company. *SCUPI+* parses each list element into a `Company` object. This object is the added using the `addProductionCompany` in the `Movies` class.
- `production_countries`: a JSON list that stores the production countries for a film. Each entry in the JSON list has a key value pair, where the key is the ISO 3166 2-character string, and the value is the country name. *SCUPI+* parses only handles the key, and uses a function to match this to the country name. This string is added using the `addProductionCountry` in the `Movies` class.
- `release_date`: a long integer representing the number of seconds from 1st January 1970 when the film was released. *SCUPI+* passes this into a Java Calendar object.
- `revenue`: a long integer representing the amount of money made by the film in US Dollars. If the revenue of the film is not known, then the revenue is set to 0. Therefore, this will always be greater than or equal to 0.
- `runtime`: a floating point value representing the number of minutes the film takes to play. If the runtime is not know, then the runtime is set to 0. Therefore, this will always be greater than or equal to 0.
- `spoken_languages`: a JSON list that stores all the languages that the film is available in. This is stored as a list of key-value pairs, where the key is the 2 -character ISO 639 code, and the value is the language name. *SCUPI+* parses these as an array of keys stored as strings.
- `status`: a string representing the current state of the film.
- `tagline`: a string representing the poster tagline of the film. A film is not guaranteed to have a tagline. In these cases, an empty string is presented.
- `title`: a string representing the English title of the film.
- `video`: a boolean representing whether the film is a "direct-to-video" film.

- `vote_average`: a floating point value representing an average score as given by a those on IMDb at the time the data was collected. As such, it is not used in the Review dataset. The score will always be between 0 and 10. This data is added using the `setVote` function in the `Movies` class.

- `vote_count`: an integer representing the number of votes on IMDb at the time the data was collected, to calculate the score for `vote_average`. As such, it is not used in the Review dataset. This will always be greater than or equal to 0. This data is added using the `setVote` function in the `Movies` class.

3.6.3 Credits

The following is a list all of the data stored about the cast and crew of a film using the column names from the CSV file, in the same order they are in the CSV file. All these fields are used by *SCUPI+*:

- `cast`: a JSON list that contains all the cast for a particular film. In the JSON list, each cast member has details that relate to there role in the film and themselves. *SCUPI+* passes this into an array of `Cast` objects, with as many fields populated as possible.

- `crew`: a JSON list that contains all the crew for a particular film. In the JSON list, each crew member has details that relate to there role in the film and themselves. *SCUPI+* passes this into an array of `Crew` objects, with as many fields populated as possible.

- `tmdb_id`: an integer representing the film ID. The values for this directly correlates to the `id` field in the movies data set.

3.6.4 Ratings

The following is a list all of the data stored about the ratings for a film using the column names from the CSV file, in the same order they are in the CSV file. Blue fields are ones that are actually used by *SCUPI+*:

- `userId`: an integer representing the user ID. The value of this is greater than 0.

- `movieLensId`: an integer representing the MovieLens ID. This is not used in this application, so can be disregarded.

- `tmdbId`: an integer representing the film ID. The values for this directly correlates to the `id` field in the movies data set.

- `rating`: a floating point value representing the rating between 0 and 5 inclusive.

- `timestamp`: a long integer representing the number of seconds from 1st January 1970 when the rating was made. *SCUPI+* passes this into a Java Calendar object.

3.6.5 Keywords

The following is a list all of the data stored about the keywords for a film using the column names from the CSV file, in the same order they are in the CSV file. All these fields are used by *SCUPI+*:

- `tmdb_id`: an integer representing the film ID. The values for this directly correlates to the `id` field in the movies data set.

- `keywords`: a JSON list that contains all the keywords relating to a given film. Each keyword is represented as a key-value pair, where the key is represented as an ID number, and the value is represented as a string. *SCUPI+* passes this into an array of `Keyword` objects.

4 Submission

You should submit one **.zip** file, containing the following files:

- **(50 marks)** Three data store files for marking the unit tests:
  - `src/main/java/stores/Movies.java`
  - `src/main/java/stores/Credits.java`
  - `src/main/java/stores/Ratings.java`

  Also, **submit any data structure** files that has been created by you (**DO NOT** submit the `MyArrayList` we provided). Please note that when using these data structures, please place them under the directory `src/main/java/structures`, as what we will do when running your program.
- **(50 marks)** A **PDF report** ($\leq$ 1500 words) discussing the data structure(s) you have implemented for the 3 data stores. More specifically:
  - **(20 marks)** Justify your choice of the data structure(s) among so many other data structures.
  - **(20 marks)** Discuss how you use the data structure(s) to build the required operations in the 3 data stores.
  - **(10 marks)** An extra 10 marks are for the organisation and presentation of your report.

In the end, please don't forget to compress all these files into a **.zip** file, and name the **.zip** file as:

"**[CW]-[Session Number]-[Student ID]-[Your name]**"

For instance, CW-01-2023141520000-Tom.zip