Ton Duc Thang University
Faculty of Information Technology

Assignment

MÔN: **CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

MÃ MÔN: **504008**

Đề bài: Phân tích đánh giá phim của người dùng

Version**: 1.1 –** Ngày: **23/11/2023**

(Sinh viên đọc kỹ tất cả hướng dẫn trước khi làm bài)

I. Giới thiệu bài toán

Một dữ liệu về đánh giá của người dùng về các bộ phim đã được thu thập. Dữ liệu này rất lớn, để phân tích được dữ liệu, người ta muốn xây dựng một đồ thị để biểu diễn dữ liệu này. Sau đó, một số câu hỏi phân tích sẽ được đặt ra để tìm ra các kết quả thỏa điều kiện trên dữ liệu.

Các chức năng mà sinh viên phải thực hiện là:

- Đọc dữ liệu từ file và xây dựng đồ thị từ dữ liệu này.
- Viết hàm để truy vấn dữ liệu trên đồ thị theo điều kiện của đề bài.

II. Tài nguyên cung cấp

Source code được cung cấp sẵn bao gồm các file:

- File đầu vào và kết quả mong muốn:
  - Folder *data* gồm 3 file:
    - *user.csv*: chứa thông tin của người dùng.
    - *movie.csv*: chứa thông tin của bộ phim.
    - *rating.csv:* chứa đánh giá của người dùng cho bộ phim.
  - Folder *expected_output* gồm: 7 file *Req1.txt, Req2.txt, Req3.txt, Req4.txt, Req5.txt, Req6.txt, Req7.txt* chứa kết quả chạy thử bài làm của các Yêu cầu từ 1 đến 7 trong bài.
- File mã nguồn:
  - *Main.java*: chứa phương thức **main** đã gọi sẵn các phương thức.
  - *Movie.java* và *User.java*: chứa các lớp được định nghĩa sẵn để tạo ra đối tượng tương ứng như trong sơ đồ lớp. **File này sinh viên không được chỉnh sửa**.
  - *RatingManagement.java*: chứa lớp **RatingManagement** được định nghĩa sẵn thuộc tính *rating* để chứa danh sách các các lượt đánh giá phim trong hệ thống, thuộc tính *movies* chứa danh sách các phim trong hệ thống, và *users* chứa danh sách các người dùng hệ thống. Phương thức khởi tạo và phương thức get được cung cấp sẵn. Sinh viên

sẽ hiện thực thêm vào các phương thức còn trống trong file này và không chỉnh sửa phương thức có sẵn.

III. Trình tự thực hiện bài

- Sinh viên tải file tài nguyên được cung cấp sẵn.
- Đọc hiểu đề bài và các file cung cấp sẵn, sau đó tạo file Rating.java, hiện thực lớp **Rating** và lập trình thêm vào lớp **RatingManagement** theo sơ đồ lớp và yêu cầu mô tả trong các phần tiếp theo.
- Sau khi hoàn thành các lớp trên, sinh viên có thể biên dịch và chạy với phương thức **main** trong file *Main.java* đã gọi sẵn các phương thức để kiểm thử các yêu cầu tương ứng. Ví dụ:
    - Để chạy từng yêu cầu, sau khi biên dịch sinh viên chạy file Main với tham số là yêu cầu cần chạy. Ví dụ chạy Yêu cầu 1:

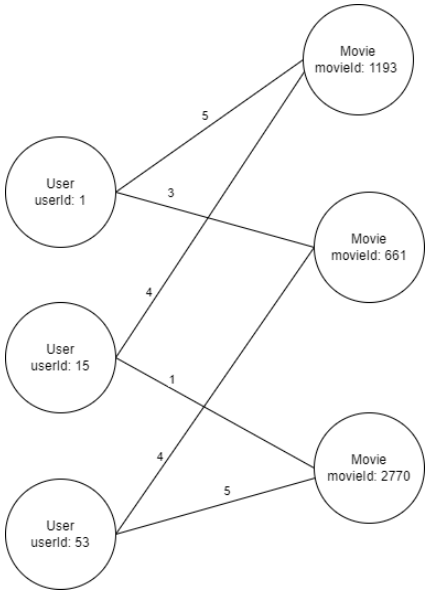
      java Main 1
    - Để chạy cùng lúc nhiều yêu cầu, sau khi biên dịch sinh viên chạy file main với tham số là các yêu cầu cần chạy. Ví dụ chạy cùng lúc Yêu cầu 1, 2, 3:

      java Main 1 2 3

    Sinh viên thực hiện các yêu cầu ở mục dưới và so sánh kết quả của mình với kết quả trong folder *expected_output* đã được cung cấp sẵn.
- Đối với các yêu cầu sinh viên không làm được vui lòng không xóa và phải đảm bảo chương trình hoạt động được với phương thức main trong *Main.java* được cung cấp sẵn.
- Đường link Google Drive gửi đề này cho sinh viên sẽ chứa kèm một file *version.txt*, sinh viên nên thường xuyên vào xem file này, nếu thấy có nội dung mới thì nên đọc kĩ mô tả và tải lại đề mới nhất. File này được dùng để thông báo nội dung chỉnh sửa và ngày chỉnh sửa trong trường hợp đề bị sai hoặc lỗi.

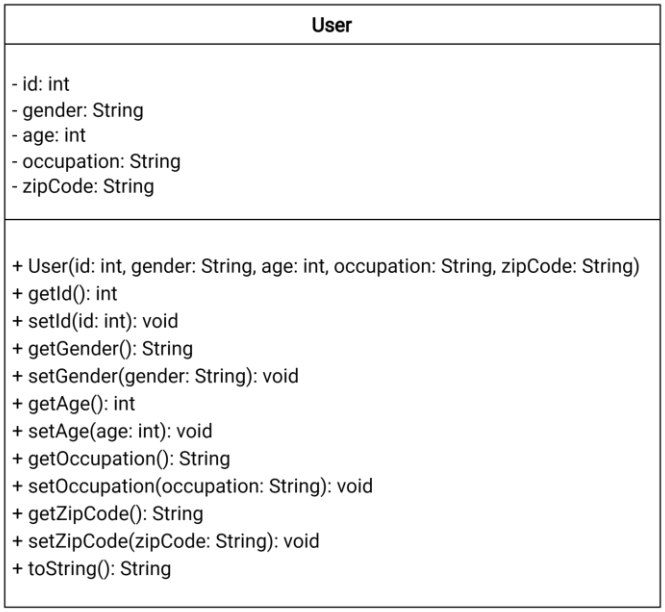
IV. Mô tả bài toán

Dữ liệu đánh giá phim của người dùng sẽ được thể hiện dưới dạng đồ thị hai phía (Bipartite Graph) bao gồm một tập người dùng và một tập các bộ phim được đánh giá. Ở bài tập này, sinh viên sẽ xây dựng đồ thị và lưu trữ dưới dạng Danh sách cạnh (Edge List). Tổng số lượng đỉnh của đồ thị là gần 10.000 đỉnh và hơn 1.000.000 cạnh (số lượng chính xác sinh viên sẽ được biết sau khi hiện thực thành công Yêu cầu 1).

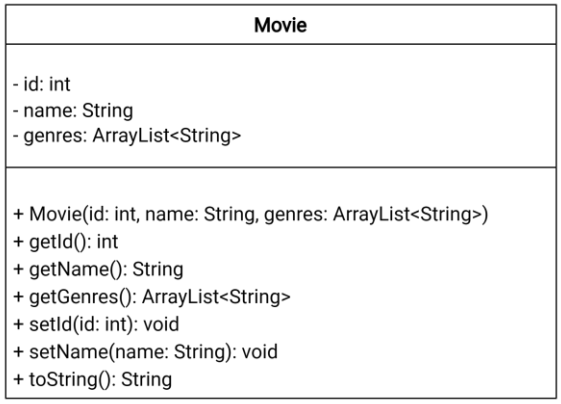


V. Mô tả các lớp trong bài

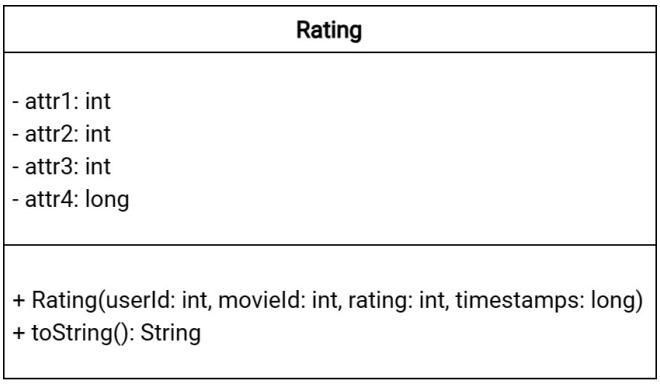
- Mô tả các lớp như sau:
  - Tất cả các thuộc tính của lớp đều phải định nghĩa với access modifer là *private*.



  - Lớp **User** – Người dùng **(Sinh viên không chỉnh sửa lớp này)**
    - Gồm 05 thuộc tính:
      - id: int – mã số người dùng
      - gender: String – giới tính
      - age: int – nhóm tuổi
      - occupation: String – nghề nghiệp
      - zipCode: String – zip code
    - Phương thức:
      - Phương thức khởi tạo đầy đủ tham số theo thứ tự của 5 thuộc tính trên.
      - Các phương thức getter/setter được hiện thực đầy đủ của cả 5 thuộc tính.
      - Phương thức **toString()**: theo format User[mã số người dùng, giới tính, nhóm tuổi, nghề nghiệp, zip code]



- Lớp **Movie** – Phim **(Sinh viên không chỉnh sửa lớp này)**
  - Gồm 03 thuộc tính:
    - id: int – mã số phim.
    - name: String – tên phim
    - genres: ArrayList<String> – danh sách thể loại của phim
  - Phương thức:
    - Phương thức khởi tạo đầy đủ tham số theo thứ tự của 3 thuộc tính trên.
    - Các phương thức getter/setter được hiện thực cho các thuộc tính..
    - Phương thức **toString()**: theo format Movie[mã số phim, tên phim, danh sách thể loại của phim]



- Lớp **Rating** - Dữ liệu đánh giá cho bộ phim của người dùng (Sinh viên hiện thực lớp này)
  - Gồm 04 thuộc tính:
    - Thuộc tính 1: int – mã người dùng
    - Thuộc tính 2: int – mã phim
    - Thuộc tính 3: int – điểm đánh giá

- Thuộc tính 4: long – dấu thời gian đánh giá phim (dấu thời gian càng lớn thể hiện ngày gửi đánh giá càng mới)

Sinh viên tự đặt tên các thuộc tính.

- Phương thức:
  - Phương thức khởi tạo đầy đủ tham số theo thứ tự của 4 thuộc tính trên theo thứ tự (mã người dùng, mã phim, điểm đánh giá, dấu thời gian).
  - Phương thức getter/setter tương ứng với các thuộc tính trên.
  - Phương thức **toString()**: theo format Rating[mã người dùng, mã phim, điểm đánh giá, dấu thời gian]

RatingManagement
- ratings: ArrayList<Rating> - movies: ArrayList<Movie> - users: ArrayList<User>
+ MovieManagement(moviePath: String, ratingPath: String, userPath: String) - loadEdgeList(ratingPath: String): ArrayList<Rating> - loadMovies(moviePath: String): ArrayList<Movie> - loadUsers(userPath: String): ArrayList<User> + getMovies(): ArrayList<Movie> + getUsers(): ArrayList<User> + getRating(): ArrayList<Rating> + findMoviesByNameAndMatchRating(userId: int, rating: int): ArrayList<Movie> + findUsersHavingSameRatingWithUser(userId: int, movieId: int): ArrayList<User> + findMoviesNameHavingSameReputation(): ArrayList<String> + findMoviesMatchOccupationAndGender(occupation: String, gender: String, k: int, rating: int): ArrayList<String> + findMoviesByOccupationAndLessThanRating(occupation: String, k: int, rating: int): ArrayList<String> + findMoviesMatchLatestMovieOf(userId: int, rating: int, k: int): ArrayList<String>

- Lớp **RatingManagement** - Phân tích hành vi xem phim của người dùng
  - Gồm 03 thuộc tính:
    - ratings: ArrayList<Rating> – Danh sách cạnh của đồ thị
    - movies: ArrayList<Movie> – Danh sách các phim trong hệ thống
    - users: ArrayList<User> – Danh sách người dùng trong hệ thống
  - Phương thức:
    - Phương thức khởi tạo nhận vào 03 tham số là đường dẫn dẫn đến các file dữ liệu (*.*csv*)
    - Phương thức getter/setter tương ứng.
    - Có 09 phương thức tương ứng phục vụ cho *Yêu cầu từ 1 đến 7*.

VI. Mô tả file input và file output

- File input có tên *movies.csv* chứa danh sách các bộ phim với mỗi dòng ứng với thuộc tính của 01 bộ phim được cách nhau bằng dấu "," theo định dạng:

  mã số phim,tên phim,thể loại

  Lưu ý: phần dữ liệu thể loại phim được phân cách nhau bằng dấu "-". Dòng đầu tiên của file chứa tên header của mỗi cột dữ liệu.

```
MovieID,Title,Genres
1,Toy Story (1995),Animation-Children's-Comedy
2,Jumanji (1995),Adventure-Children's-Fantasy
3,Grumpier Old Men (1995),Comedy-Romance
4,Waiting to Exhale (1995),Comedy-Drama
5,Father of the Bride Part II (1995),Comedy
```

  Ví dụ dòng dữ liệu 1 là bộ phim có mã phim là *1*, tên phim là *Toy Story (1995)* và bộ phim thuộc các thể loại: *Animation*, *Children's* và *Comedy*.

- File input có tên *users.csv* chứa danh sách các người dùng trong hệ thống với mỗi dòng ứng với thuộc tính của 01 người dùng được cách nhau bằng dấu "," theo định dạng:

  mã số người dùng,giới tính,nhóm tuổi,nghề nghiệp,zipcode

  Lưu ý: giới tính nam và nữ được kí hiệu lần lượt là "M" và "F"; con số trong thuộc tính tuổi thể hiện độ tuổi bắt đầu của nhóm tuổi đó. Dòng đầu tiên của file chứa header của mỗi cột dữ liệu.

```
UserID,Gender,Age,Occupation,Zip-code
1,F,1,K-12 student,48067
2,M,56,Self-employed,70072
3,M,25,Scientist,55117
4,M,45,Executive/managerial,02460
5,M,25,Writer,55455
```

  Ví dụ dòng dữ liệu 1 là người dùng có mã người dùng là *1*, giới tính là *nữ*, thuộc nhóm tuổi *1*, nghề nghiệp là *K-12 student* và zipcode là *48067*.

- File input có tên *ratings.csv* chứa danh sách các lượt đánh giá phim của người dùng trong hệ thống với mỗi dòng ứng với thuộc tính của 01 lượt đánh giá được cách nhau bằng dấu "," theo định dạng:

  mã số người dùng,mã số người phim,số sao đánh giá,thời điểm đánh giá

  Lưu ý: người dùng chỉ chọn số sao nguyên để đánh giá (1 sao, 2 sao,...); thời điểm đánh giá ở dưới dạng timestamp; và dòng đầu tiên của file chứa header của mỗi cột dữ liệu.

```
UserID,MovieID,Rating,Timestamp
1,1193,5,978300760
1,661,3,978302109
1,914,3,978301968
1,3408,4,978300275
1,2355,5,978824291
```

Ví dụ dòng dữ liệu 1 là lượt đánh giá của người dùng có mã số người dùng là *1*, cho phim có mã số phim là *1193*, với số sao là 5, và tại thời điểm cụ thể mà người dùng đánh giá là *978300760* (quy đổi sang ngày tháng là ngày 01/01/2001, 05:12:40 AM, GMT+7).

- Thư mục *expected_output* gồm có 7 file ứng với kết quả mẫu cho các yêu cầu từ 1 đến 7. Trong đó:
    - Đối với Yêu cầu 2 và 3: Mỗi dòng trong từng file *Req2.txt* và *Req3.txt* ứng với một đối tượng, có định dạng ghi ra theo theo phương thức **toString()** của lớp **tương ứng**.
    - Đối với các yêu cầu còn lại: Mỗi dòng trong từng file *.txt* là tên phim ứng với yêu cầu của đề bài.

Lưu ý:

- Sinh viên nên đọc kỹ nội dung trong các phương thức **main** để xác định hướng hiện thực.
- Sinh viên có thể thêm một số thao tác vào phương thức **main** để kiểm các phương thức đã định nghĩa tuy nhiên đảm bảo bài làm của sinh viên **phải chạy được trên phương thức main ban đầu đã cung cấp sẵn**.
- Sinh viên có thể thêm phương thức mới để phục vụ bài làm tuy nhiên bài làm của sinh viên phải chạy được với các file *Main.java* đã được cung cấp sẵn.
- Sinh viên tuyệt đối không đặt các đường dẫn tuyệt đối khi định nghĩa phương thức liên quan đến đọc file. Nếu sinh viên tự ý đặt đường dẫn tuyệt đối dẫn đến quá trình chấm không đọc được file để chấm thì tương đương với bài làm biên dịch lỗi.
- Tuyệt đối tuân thủ theo **tên của các phương thức** được yêu cầu cụ thể trong đề bài.

VII. Yêu cầu

Ngoài những thư viện có sẵn trong các file nhận được, sinh viên được phép sử dụng các lớp trong package ***java.util*** và các lớp dùng cho nhu cầu đọc file trong package ***java.io***. Ngoài ra, sinh viên không tự ý thêm thư viện khác.

Sinh viên thực hiện bài tập lớn trên Java 11 hoặc Java 8. Sinh viên không được dùng kiểu dữ liệu *var*. Bài làm của sinh viên sẽ được chấm trên Java 11, sinh viên tự chịu trách nhiệm tất cả các lỗi phát sinh nếu dùng các phiên bản Java khác.

Sinh viên được phép định nghĩa thêm phương thức để hỗ trợ bài làm nhưng chỉ định nghĩa thêm trong các file được yêu cầu nộp. Sinh viên không tự ý thay đổi tên phương thức và tham số truyền vào của các phương thức có sẵn và các phương thức được yêu cầu trong đề bài.

Các thuộc tính được đánh số trong phần **Mô tả các lớp trong bài**, khi sinh viên định nghĩa sẽ tự đặt tên cho thuộc tính, chỉ cần đảm bảo đúng kiểu dữ liệu theo mô tả và *access modifier* của tất cả các thuộc tính phải là *private*.

Do lượng dữ liệu lớn, trong quá trình sinh viên xử lý các yêu cầu, sinh viên cần quan tâm thời gian chạy của chương trình và thực hiện tối ưu thuật toán nếu cần thiết để giảm thời gian chạy. Trong quá trình chấm, nếu bài làm của sinh viên chạy quá khoảng thời gian cho phép của chương trình chấm thì yêu cầu đó của sinh viên 0 điểm.

1. YÊU CẦU 1 (1 điểm)

Sinh viên sẽ định nghĩa các phương thức trong lớp **RatingManagement** và dùng lớp **Main** để kiểm tra kết quả bài làm.

Hiện thực phương thức:

private ArrayList<Movie> loadMovies(String moviePath)

private ArrayList<User> loadUsers(String userPath)

public ArrayList<Rating> loadEdgeList(String ratingPath)

Phương thức này sẽ đọc các file trong thư mục *data* theo đường dẫn của tham số *ratingPath, moviePath, userPath* truyền vào để đọc dữ liệu, lưu vào thuộc tính có sẵn. Trong đó, phương thức **loadEdgeList** thêm dữ liệu đánh giá vào một Edge List.

Sinh viên hiện thực phương thức trên, biên dịch file *Main.java* và chạy lệnh *java Main 1* để ghi ra kết quả file *Req1.txt* vào thư mục *output*. Khi sinh viên hiện thực thành công 03 phương thức trên thì sẽ ghi được kết quả trong file *Req1.txt* là tổng số đỉnh và tổng số cạnh của đồ thị. Sinh viên xem kết quả file này và file *Req1.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

Lưu ý: Đây là phương thức sinh viên phải định nghĩa được mới bắt đầu tính điểm cho các Yêu cầu từ 2 đến 7, nếu không đọc được file thành danh sách cạnh của đồ thị thì các yêu cầu bên dưới không được tính điểm.

Các yêu cầu bên dưới thao tác trực tiếp trên dữ liệu đã đọc từ file.

2. YÊU CẦU 2 (2 điểm)

Hiện thực phương thức

public ArrayList<Movie> findMoviesByNameAndMatchRating(int userId, int rating)

trả về danh sách các bộ phim **Movie** được người dùng có mã ***userId*** đánh giá lớn hơn hoặc bằng điểm ***rating*** truyền vào. Kết quả cuối dùng, các bộ phim phải được sắp xếp tên theo bảng chữ cái Alphabet.

Sinh viên hiện thực phương thức trên, biên dịch file *Main.java* và chạy lệnh *java Main 2* để ghi ra kết quả file *Req2.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req2.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

3. YÊU CẦU 3 (2 điểm)

Hiện thực phương thức

public ArrayList<User> findUsersHavingSameRatingWithUser(int userId, int movieId)

trả về danh sách những người dùng **User** với điều kiện những người dùng này cùng đánh giá bộ phim có mã số là ***movieId*** có cùng chính xác số điểm với số điểm mà người dùng có mã số là ***userId*** đã đánh giá cho bộ phim có mã số là ***movieId*** đó.

Sinh viên hiện thực phương thức trên, biên dịch file *Main.java* và chạy lệnh *java Main 3* để ghi ra kết quả file *Req3.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req3.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

4. YÊU CẦU 4 (2 điểm)

Hiện thực phương thức

public ArrayList<String> findMoviesNameHavingSameReputation()

Trả về danh sách tên các bộ phim được ít nhất hai người dùng bất kỳ yêu thích (định nghĩa yêu thích là được đánh giá lớn hơn 3 sao). Kết quả cuối dùng, các bộ phim phải được sắp xếp tên theo bảng chữ cái Alphabet.

Sinh viên hiện thực phương thức trên. Sinh viên biên dịch file *Main.java* và chạy lệnh *java Main 4* để ghi ra kết quả file *Req4.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req4.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

5. YÊU CẦU 5 (1 điểm)

Hiện thực phương thức

public ArrayList<String> findMoviesMatchOccupationAndGender(String occupation, String gender, int k, int rating)

trả về danh sách chứa tối đa ***k*** tên bộ phim của những người dùng có cùng ngành nghề ***occupation*** và có cùng giới tính **gender** có bằng điểm ***rating***. Kết quả cuối cùng phải được sắp xếp theo thứ tự bảng chữ cái Alphabet.

Sinh viên hiện thực phương thức trên. Sinh viên biên dịch file *Main.java* và chạy lệnh *java Main 5* để ghi ra kết quả file *Req5.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req5.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

6. YÊU CẦU 6 (1 điểm)

Hiện thực phương thức

public ArrayList<String> findMoviesByOccupationAndLessThanRating(String occupation, int k, int rating)

trả về danh sách chứa tối đa ***k*** tên bộ phim được những người dùng có cùng nghề nghiệp ***occupation*** đánh giá bé hơn số điểm ***rating***. Kết quả cuối cùng phải được sắp xếp theo thứ tự bảng chữ cái Alphabet.

Sinh viên hiện thực phương thức trên. Sinh viên biên dịch file *Main.java* và chạy lệnh *java Main 6* để ghi ra kết quả file *Req6.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req6.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

7. YÊU CẦU 7 (1 điểm)

Hiện thực phương thức

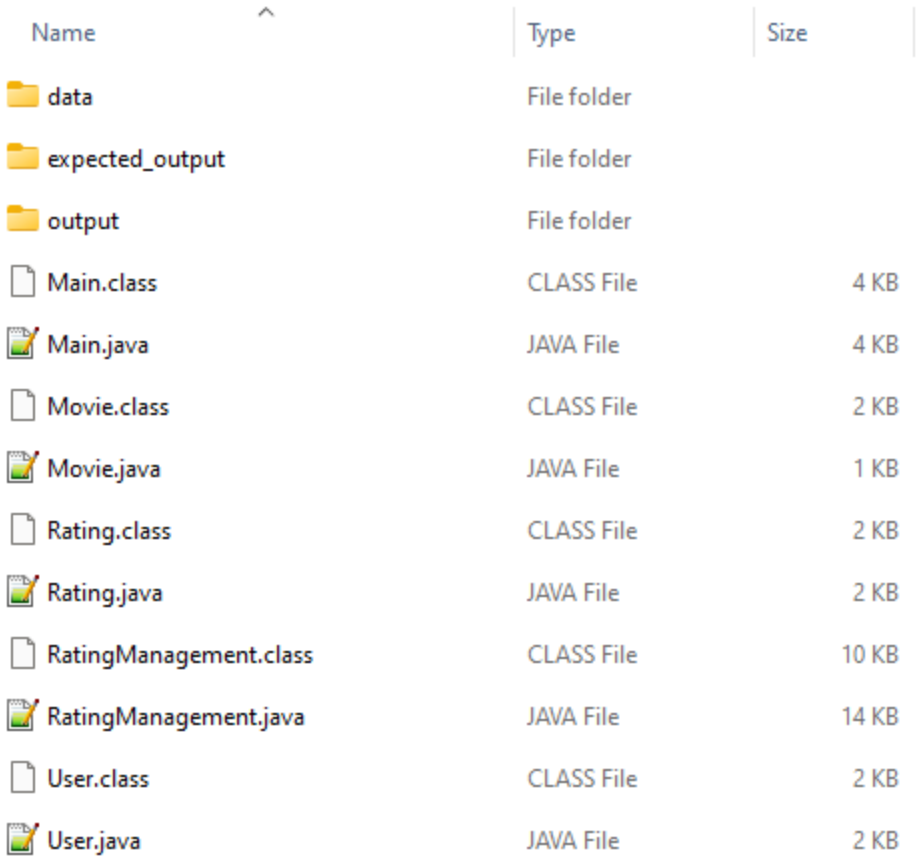
public ArrayList<String> findMoviesMatchLatestMovieOf(int userId, int rating, int k)

trả về danh sách chứa tối đa ***k*** tên bộ phim được những người dùng có cùng giới tính với người dùng có mã người dùng là ***userId*** đánh giá lớn hơn bằng điểm ***rating*** và có cùng chung ít nhất 01 thể loại với bộ phim được người dùng có ***userId*** đánh giá mới nhất (xem xét thuộc tính **dấu thời gian**) có điểm đánh giá lớn hơn hoặc bằng điểm ***rating***. Kết quả cuối cùng phải được sắp xếp theo thứ tự bảng chữ cái Alphabet.

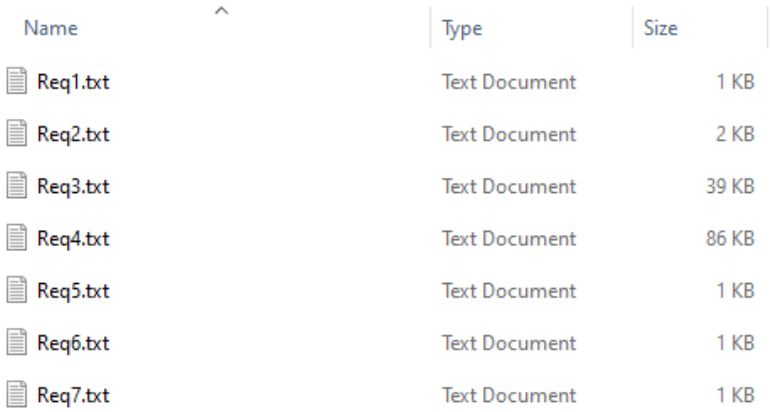
Sinh viên hiện thực phương thức trên. Sinh viên biên dịch file *Main.java* và chạy lệnh *java Main 7* để ghi ra kết quả file *Req7.txt* vào thư mục *output*. Sinh viên xem kết quả file này và file *Req7.txt* trong folder *expected_output* mẫu được cung cấp sẵn để so sánh kết quả.

VIII. Lưu ý kiểm tra trước khi nộp bài

- Nếu sinh viên không thực hiện được yêu cầu nào thì để nguyên phương thức của yêu cầu đó, TUYỆT ĐỐI KHÔNG XÓA PHƯƠNG THỨC CỦA YÊU CẦU sẽ dẫn đến lỗi khi chạy phương thức **main**. Trước khi nộp phải kiểm tra chạy được với phương thức **main** được cho sẵn.
- Tất cả các file ReqX.txt (X = {1,2,3,4,5,6,7}) được ghi ra trong thư mục *output*. Đối với sinh viên sử dụng IDE (Eclipse, Netbean, …) phải đảm bảo file chạy được bằng command prompt, đảm bảo bài làm không nằm trong package. Trường hợp bài làm đặt trong package thì sẽ bị 0 điểm cả bài.
- File kết quả ghi đúng thư mục khi chạy chương trình sẽ có dạng như sau:



- Bên trong thư mục *output*:



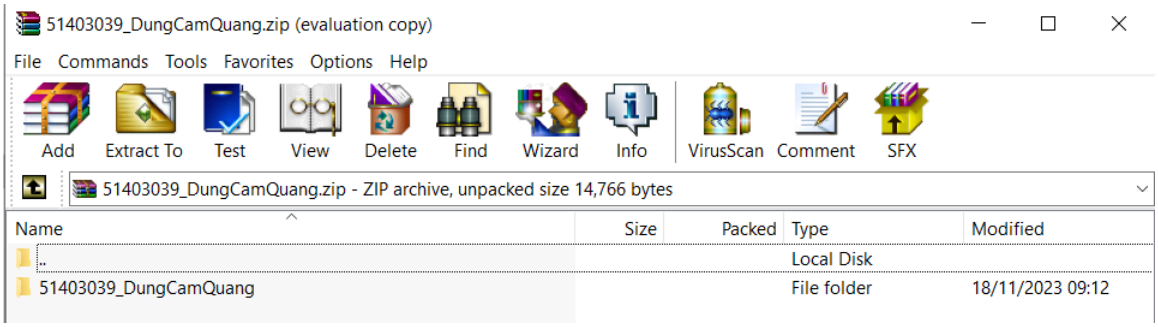
Name	Type	Size
Req1.txt	Text Document	1 KB
Req2.txt	Text Document	2 KB
Req3.txt	Text Document	39 KB
Req4.txt	Text Document	86 KB
Req5.txt	Text Document	1 KB
Req6.txt	Text Document	1 KB
Req7.txt	Text Document	1 KB

IX. Hướng dẫn nộp bài

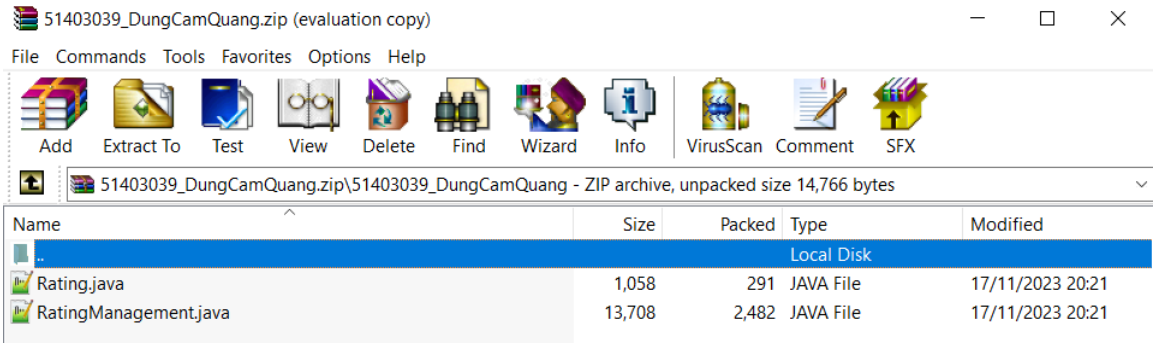
- Khi nộp bài sinh viên nộp lại file *Rating.java* và *RatingManagement.java*, **không nộp kèm bất cứ file nào khác và tuyệt đối không được sửa tên 2 file này.**
- **Sinh viên đặt 2 file bài làm vào thư mục MSSV_HoTen** (HoTen viết liền, không dấu) và nén lại với định dạng **.zip** và nộp vào mục nộp tương ứng trên hệ thống ELIT (elit.tdtu.edu.vn).
- Trường hợp làm sai yêu cầu nộp bài (đặt tên thư mục sai, không để bài làm vào thư mục khi nộp, nộp dư file, nộp thiếu file, …) thì bài làm của sinh viên sẽ bị **0 điểm**.
- File nộp đúng sẽ như sau:
  - File nén nộp bài:

51403039_DungCamQuang.zip 18/11/2023 09:13 WinRAR ZIP archive 4 KB

  - Bên trong file nén:



- Bên trong thư mục:



X. Đánh giá và quy định

- Bài làm sẽ được chấm tự động thông qua testcase (file input và output có định dạng như mẫu đã gửi kèm) do đó sinh viên tự chịu trách nhiệm nếu không thực hiện đúng theo Hướng dẫn nộp bài hoặc tự ý sửa tên các phương thức đã có sẵn dẫn đến bài làm không biên dịch được khi chấm.
- ***Tất cả các trường hợp sinh viên gán cứng đường dẫn trong quá trình đọc file đều sẽ bị 0 điểm cả bài.***
- Testcase sử dụng để chấm bài là file khác với file sinh viên đã nhận, sinh viên chỉ được điểm mỗi YÊU CẦU khi chạy ra đúng hoàn toàn kết quả theo testcase chấm của yêu cầu đó.
- Nếu bài làm của sinh viên biên dịch bị lỗi trên máy chấm thì **0 điểm cả bài**.
- **Tất cả code sẽ được kiểm tra sao chép. Mọi hành vi sao chép code trên mạng, chép bài bạn hoặc cho bạn chép bài nếu bị phát hiện đều sẽ bị điểm 0 vào điểm Quá trình 2 hoặc cấm thi cuối kì.**
- **Sinh viên tự bảo vệ mã nguồn của bài làm, vì bất cứ lý do nào mã nguồn của sinh viên lộ ra ngoài đều sẽ nhận 0 điểm.**
- **Đặc biệt, nếu sinh viên sử dụng chat GPT trong bài tập lớn này thì sinh viên sẽ bị xử lý tương tự hành vi sao chép bài.**
- Nếu bài làm của sinh viên có dấu hiệu sao chép trên mạng hoặc sao chép nhau, sinh viên sẽ được gọi lên phỏng vấn code để chứng minh bài làm là của mình.
- **Hạn chót nộp bài: 23h59 ngày 10/12/2023.**
- **Sinh viên nộp đúng vào mục bài tập của Bài tập lớn trên hệ thống ELIT.**

-- HẾT --