# Programming Exercise: Step Four

In this exercise you will continue to build on the program you wrote for the previous assignment. You will continue to make your program more efficient with a **RaterDatabase** class that is designed and implemented similarly to the MovieDatabase class you used recently—you will use both of these database classes in this assignment. You will also calculate a different kind of average movie rating, one based on weighting ratings made by raters who are more like you, or like any given rater—valuing their ratings more than raters who don't have your tastes in movies. To calculate these weighted averages you will need to calculate similarity scores for each rater to find out which raters you are more similar to than others, so you can weight ratings accordingly.

## Additional File for Assignment

For this assignment you will be given one class **RaterDatabase**, which is an efficient way to get information about raters. This class contains:

- A HashMap named **ourRaters** that maps a rater ID String to a Rater object that includes all the movie ratings made by this rater.
- A public static **initialize** method with one String parameter named **filename**. You can call this method with the name of the file used to initialize the rater database.
- A private **initialize** method with no parameters that initializes the HashMap **ourRaters** if it does not exist.
- A public static void **addRatings** method that has one String parameter named **filename**. You could alternatively call this method to add rater ratings to the database from a file.
- A public static void **addRaterRating** method that has three parameters, a String named **raterID** representing a rater ID, a String named **movieID** that represents a movie ID, and a double named **rating** that is the rating the rater **raterID** has given to the movie **movieID**. This function can be used to add one rater and their movie rating to the database. Notice that the method **addRatings** calls this method.
- A method **getRater** has one String parameter named **id**. This method returns a Rater that has this ID.
- A method **getRaters** that has no parameters. This method returns an ArrayList of Raters from the database.

- A method **size** that has no parameters. This method returns the number of raters in the database.

## Assignment

Specifically for this assignment you will do the following:

- Create a new project and copy over your classes from the last exercise.
- Create a new class named **FourthRatings**. Copy over the following methods from the class ThirdRatings and get FourthRatings to compile. Do not copy over any of the other methods. You should not copy, nor should you have any instance variables in FourthRatings—you'll use RaterDatabase and MovieDatabase static methods in place of instance variables—so where you have code with **myRaters**, you need to replace the code with calls to methods in the RaterDatabase class. The methods to copy into FourthRatings from ThirdRatings are below (you'll need to modify these after copying):
    - **getAverageByID**
    - **getAverageRatings**
    - **getAverageRatingsByFilter**
- Create a new class named **MovieRunnerSimilarRatings**. Copy the two methods **printAverageRatings** and **printAverageRatingsByYearAfterAndGenre** from MovieRunnerWithFilters to this new class and modify them to work with a FourthRatings object instead of a ThirdRatings object. You can copy more of the methods into your new Runner class, but these two should be enough to test that FourthRatings has been set up correctly. When you run these two you should get the same output you get when those methods run with the ThirdRatings object.
- In the FourthRatings class, write the following methods—two are private helper methods, and one is the method that will return movie recommendations based on similarities:
    - Write the private helper method named **dotProduct**, which has two parameters, a Rater named **me** and a Rater named **r**. This method should first translate a rating from the scale 0 to 10 to the scale -5 to 5 and return the dot product of the ratings of movies that they both rated. This method will be called by **getSimilarities**.
    - Write the private method named **getSimilarities**, which has one String parameter named **id**—this method computes a similarity rating for each rater in the RaterDatabase (except the rater with the ID given by the parameter) to see how similar they are to the Rater whose ID is the parameter to **getSimilarities**. This method returns an ArrayList of type Rating sorted by ratings from highest to

lowest rating with the highest rating first and only including those raters who have a positive similarity rating since those with negative values are not similar in any way. Note that in each Rating object the item field is a rater's ID, and the value field is the dot product comparison between that rater and the rater whose ID is the parameter to **getSimilarities**. Be sure not to use the **dotProduct** method with parameter **id** and itself!

- Write the public method named **getSimilarRatings**, which has three parameters: a String named **id** representing a rater ID, an integer named **numSimilarRaters**, and an integer named **minimalRaters**. This method should return an ArrayList of type Rating, of movies and their weighted average ratings using only the top **numSimilarRaters** with positive ratings and including only those movies that have at least **minimalRaters** ratings from those top raters. These Rating objects should be returned in sorted order by weighted average rating from largest to smallest ratings. This method is very much like the **getAverageRatings** method you have written previously. In particular this method should:
  - For every rater, get their similarity rating to the given parameter rater **id**. Include only those raters with positive similarity ratings—those that are more similar to rater **id**. Which method could you call?
  - For each movie, calculate a weighted average movie rating based on:
    - Use only the top (largest) **numSimilarRaters** raters.
    - For each of these raters, multiply their similarity rating by the rating they gave that movie. This will emphasize those raters who are closer to the rater **id**, since they have greater weights.
    - The weighted average movie rating for a particular movie is the sum of these weighted average ratings (for each rater multiply their similarity rating by their rating for the movie), divided by the total number of such ratings.
  - This method returns an ArrayList of Ratings for movies and their calculated weighted ratings, in sorted order.
- Write the public method **getSimilarRatingsByFilter**, which is similar to the **getSimilarRatings** method but has one additional Filter parameter named **filterCriteria** and uses that filter to access and rate only those movies that match the filter criteria.

- Add the following methods to the MovieRunnerSimilarRatings class.
    - Write a void method **printSimilarRatings** that has no parameters. This method creates a new FourthRatings object, reads data into the MovieDatabase and RaterDatabase, and then calls **getSimilarRatings** for a particular rater ID, a number for the top number of similar raters, and a number of minimal **rateSimilarRatings**, and then lists recommended movies and their similarity ratings. For example, using the files **ratedmoviesfull.csv** and **ratings.csv** and the rater ID 65, the number of minimal raters 5, and the number of top similar raters set to 20, the movie returned with the top rated average is “The Fault in Our Stars”.
    - Write a void method **printSimilarRatingsByGenre** that has no parameters. This method is similar to **printSimilarRatings** but also uses a genre filter and then lists recommended movies and their similarity ratings, and for each movie prints the movie and its similarity rating on one line and its genres on a separate line below it. For example, using the files **ratedmoviesfull.csv** and **ratings.csv**, the genre “Action”,  the rater ID 65, the number of minimal raters set to 5, and the number of top similar raters set to 20, the movie returned with the top rated average is “Rush”.
    - Write a void method **printSimilarRatingsByDirector** that has no parameters. This method is similar to **printSimilarRatings** but also uses a director filter and then lists recommended movies and their similarity ratings, and for each movie prints the movie and its similarity rating on one line and its directors on a separate line below it. For example, using the files **ratedmoviesfull.csv** and **ratings.csv**, the directors “Clint Eastwood,Sydney Pollack,David Cronenberg,Oliver Stone”, the rater ID 1034, the number of minimal raters set to 3, and the number of top similar raters set to 10, the movie returned with the top rated average is “Unforgiven”.
    - Write a void method **printSimilarRatingsByGenreAndMinutes** that has no parameters. This method is similar to **printSimilarRatings** but also uses a genre filter and a minutes filter and then lists recommended movies and their similarity ratings, and for each movie prints the movie, its minutes, and its similarity rating on one line and its genres on a separate line below it. For example, using the files **ratedmoviesfull.csv** and **ratings.csv**, the genre “Adventure”,  minutes

between 100 and 200 inclusive, the rater ID 65, the number of minimal raters set to 5, and the number of top similar raters set to 10, the movie returned with the top rated average is “Interstellar”.

- Write a void method **printSimilarRatingsByYearAfterAndMinutes** that has no parameters. This method is similar to **printSimilarRatings** but also uses a year-after filter and a minutes filter and then lists recommended movies and their similarity ratings, and for each movie prints the movie, its year, its minutes, and its similarity rating on one line. For example, using the files **ratedmoviesfull.csv** and **ratings.csv**, the year 2000, minutes between 80 and 100 inclusive, the rater ID 65, the number of minimal raters set to 5, and the number of top similar raters set to 10, the movie returned with the top rated average is “The Grand Budapest Hotel”.