Practice Lab: Movie Tracker

Task 1: The `MovieRecord` Class

Create a class `MovieRecord` to store movie data:

- Fields: `String title`, `String genre`, `int rating` (1-10 scale).
- Methods:
    - Constructor to initialize all fields.
    - Getters and setters.
    - `toCSV()`: returns a CSV string (`title,genre,rating`).

Task 2: Summing Ratings

Write a static method `sumRatings` that:

- Accepts an `ArrayList<MovieRecord>` and a genre.
- Returns the sum of ratings for movies in the given genre.

Task 3: Client-Server Communication

1. **Client:** Write a static method `sendMoviesToServer`:
    - Connects to a server on localhost port 9000.
    - Sends `toCSV` strings of movies from an `ArrayList<MovieRecord>` to the server.
    - Ends with "Done" and receives the count of movies sent as a string.
2. **Server:** Write a static method `receiveMoviesFromClient`:
    - Listens on port 9000.
    - Accepts connections and reads CSV movie data until "Done".
    - Returns an `ArrayList<MovieRecord>` and sends back the number of movies received.

Task 4: Mapping and Prioritizing Movies

1. Write a method `genreMap`:

    - Takes an `ArrayList<MovieRecord>` and returns a `Map<String, Integer>` where:
        - Key = genre.
        - Value = total ratings of all movies in that genre.
2. Create a `PriorityQueue<MovieRecord>`:
    - Prioritize movies by rating (higher ratings first).
    - Populate the queue with movies from an `ArrayList<MovieRecord>`.

Task 5: Multi-Threaded Movie Rating Aggregation

1. Create a class `GenreRatingAggregator`:
    - Fields: `ArrayList<MovieRecord> movieList`, `String genre`, `int totalRating`.
    - Implements `Runnable` and calculates the sum of ratings for its genre in `run()`.
    - Provides a getter for `totalRating`.
2. In `main`:
    - Create two `GenreRatingAggregator` objects for two genres.
    - Use threads to execute them concurrently.
    - Wait for threads to finish and print the ratings for both genres.