Project 8: Most Popular Movie

Due: Monday, Mar. 16 at 11:59 PM

Description: The popularity of holiday-themed films spikes in mid-November and December. To capitalize on this demand, video streaming websites like Netflix, Amazon, and Hulu collect data on which movies are watched the most during the holiday season. They use this data to decide what movies to keep and remove from their platforms.

In this project, we will simulate this data analysis by writing a program to find the most popular movie during a week in December. The viewing data is stored in a text file with a movie title on each line. Whenever someone watches a movie, its title is appended to the end of the file. This means that the data is not sorted. Movie titles appear multiple times throughout the file, and the most popular movie is the one that appears the greatest number of times.

We are providing you with starter code for this project that includes signatures for the main method and three additional methods. Your goal is to write the body of each method. If done correctly, the resulting program will read any file with the name "movie-data.txt" and output the line that appears the most.

Objectives: Your program will be graded according to the rubric below.

1. (25 points) Write the body of the method String[] readFile(String filename). This method reads a text file with a given name and returns an array that contains each line of the file.
2. (25 points) Write the body of the method void lowercase(String[] array). This method converts all capital letters in a string array to lowercase letters.
3. (25 points) Write the body of the method String findMostFrequent(String[] array). This method returns the element of the given array that appears the most times.
4. (15 points) Write the main method. This method uses the methods from objectives 1–3 to find and print the movie title that appears the most times in the file movie-data.txt.
5. (10 points) Use meaningful variable names, consistent indentation, and whitespace (blank lines and spaces) to make your code readable. Add comments where appropriate to explain the overall steps of your program.

Sample Output: Suppose the file movie-data.txt contains the following movie titles:

```
a christmas carol
Home Alone
A Christmas Carol
elf
HOME ALONE
home alone
```

When the program is run, it prints a single line:

```
Most popular movie: home alone
```

Note the following things about this example:

1. Although a movie title will have the same spelling everywhere it appears in the file, each appearance may have different capitalization.
2. The number of lines in the file is not known ahead of time. If the file is modified (e.g., by adding a movie), rerunning the program will output the new most popular movie.

Import Code into Eclipse: Refer to the Project 6 instructions for details on importing code into a new Java project. Remember that .java files (PopularMovie.java) always go in the src folder. The file movie-data.txt goes in the root project folder.

Three Helper Methods: The file PopularMovie.java contains three methods in addition to the main method. To complete the program, write the bodies of these methods so they work as described below.

1) `String[] readFile(String filename)`

Given the name of a text file, this method should return the lines of the file as elements of a perfect size string array.

To read the file, construct a Scanner object using a File object as the argument (rather than System.in). Below is an example:

```
Scanner file = new Scanner(new File(filename));
```

Once the Scanner is constructed, all the usual Scanner methods (e.g., next, nextInt, nextLine) can be used to read the contents of the file.

In order to return a perfect size array, the file needs to be read twice. Read the file once to count the number of lines. Use this number to construct an array of the correct length. Then read the file a second time to copy each line into the new array. (The Scanner must be reconstructed to read from the start of the file the second time.)

2) `void lowercase(String[] array)`

Given an array of strings, this method should convert all capital letters to lowercase letters.

Iterate through the array with a loop and change each element. See the String class API documentation for a method that will convert each uppercase letter in a string to a lowercase letter.

3) `String findMostFrequent(String[] array)`

Given an array of strings, this method should return the string that appears the most times.

This method is more challenging to implement than the other two. To get you started, consider the movies from the sample output as an example. Suppose we read these movies into an array and convert all the uppercase letters to lowercase:

index	element
0	a christmas carol
1	home alone
2	a christmas carol
3	elf
4	home alone
5	home alone

Start by sorting this array so that every occurrence of a movie title appears sequentially. (Remember that the Arrays class has a method that will do this for you.) Here is the result:

index	element
0	a christmas carol
1	a christmas carol
2	elf
3	home alone
4	home alone
5	home alone

Now iterate through the array and count the number of occurrences of each title. Use variables to store the maximum number of occurrences and the movie title that corresponds to the maximum. Below is a table that traces the values of these variables for the example:

index	element	count	max	movie title
0	a christmas carol	1	1	a christmas carol
1	a christmas carol	2	2	a christmas carol
2	elf	1	2	a christmas carol
3	home alone	1	2	a christmas carol
4	home alone	2	2	a christmas carol
5	home alone	3	3	home alone

Note that after iterating through the entire array, the variable storing the movie title (the rightmost column) is “home alone”, which is the correct output.

In order to implement this algorithm, you need to compare each element of the sorted array to the previous element. (Remember that you did something similar in Project 5.) If the elements are the same, the count increases by 1. If they are different, the count resets. If the count ever becomes larger than the maximum, update the maximum (and the corresponding movie title).

Complete the Main Method: After you have written (and hopefully tested) the methods described in the last section, write the main method to complete the program. The main method should do the following four things: (1) Read the lines of movie-data.txt into a string array. (2) Convert all the uppercase letters in the array to lowercase. (3) Find the movie title that appears most frequently in the array. (4) Print the movie title in the format shown in the sample output.

Upload Code to Zybooks: After you complete each project, you need to upload your source code to Zybooks so it can be graded. If you created the folder structure described earlier, your code is in the folder "Intro to Programming\Projects\Project 8\src". The code is contained in the file Project_8.java. Note that .java files are simply text files that contain Java code. They can be opened with any text editor (e.g., Notepad or TextEdit).

Upload your .java file to Zybooks on the Project 8 assignment page. This requires a few steps. Click the "Submit Assignment" button in the top-right corner. This will reveal a button near the bottom of the page with the text "Choose File." Click this button and browse to the location of your .java file. Finally, click the "Submit for grading" button below the "Choose on hard drive" button.

If Zybooks won't accept the file, make sure you're submitting a .java file, not a .class file. Make sure to see if the output you obtained and the output which I have mentioned match to obtain complete grade.

Submission Link:

https://learn.zybooks.com/zybook/OUCS1324Winter2020/chapter/21/section/7