CS 1331 Programming Exercise 04 – Scanner, Static Methods, and Arrays

Authors: Alyse, Sreya, Nathan, Chelsea, Jack

Problem Description

Every year, music streaming platforms release a summary of a user's listening statistics throughout the year. As a result, many other websites have emerged evaluating different aspects of listening habits. Your job is to create a program that creates and evaluates the genres of the songs within the playlist and decide how often they will be listened to based on their popularity. This assignment will test your basic knowledge of Scanner, static methods, and arrays.

Notes:

- You **must** reuse code when possible. Look for hints suggesting code reuse.
- Only create **ONE** Scanner object (when prompted in the main method) and reuse it whenever you need to read more input. Creation/use of **more than one Scanner may result in a zero for the assignment!**

Solution Description

1. Create a **class** named `Playlist`
2. Create a public **static method** named `addPlaylistInfo` that **takes in** a `Scanner` object and does not return anything. Prompts should be on the **same line** as user input. Each unique prompt should be printed on separate lines. Within this method:
    a. Print to the console "`Enter number of songs in playlist: `"
    b. Read the user's input as an `int` and assign it to an `int` named `numSongs`
    c. Print to the console "`Enter a playlist name: `"
    d. Read the user's input as a `String` and assign it to a `String` named `playlistName`. The playlist name may contain spaces, so the entire line of user input should be read in.
    e. Print the name and number of songs in the following format:

```
Playlist created successfully. Here are the details:
Number of Songs: {numSongs}
Name: {playlistName}
```

3. Create a public **static method** named `genreTally` that takes in the following two parameters: a `String[]` named `genreArray`, which represents the actual genres found within a playlist, and another `String[]` named `genres`, which represents all possible genres that can be found within a playlist. This method should return an `int[]` that represents the tally of each genre present within the playlist. Within this method:
    a. Print the following statement once: "`The following types of genres are in your playlist:`"
    b. **For each** `String` in `genres`, **loop** through the `genreArray` and keep a tally of how many songs of each genre exists within `genreArray`. Then, print out a new line with the genre and the total number of that type of genre.
        i. **HINT:** Make sure that this method works for **any** size `genreArray` it receives
        ii. For example, if `genreArray` contains [`"Rap", "Rock", "Rock", "Pop", "Pop", "Hip Hop", "Hip Hop"`] and `genres` contains [`"Rap", "Pop", "Classical", "Hip Hop", "Rock", "Lo Fi"`] you should print out:

```
Rap 1
Pop 2
Classical 0
Hip Hop 2
Rock 2
Lo Fi 0
```

c. The returned array should contain the following values: `[1, 2, 0, 2, 3, 0]`

4. Create a **public**, **static method** named `playlistScores` that takes in three `String[]` parameters named `playlist1`, `playlist2`, and `genres` in that order. This method should return nothing. Within this method:
    a. Calculate the popularity of each playlist array. To calculate the popularity of a playlist array, take the sum of each element within the array multiplied by its index.
        i. **HINT:** You already have a method that calculates the amount of each genre found within a playlist. You **must** use that method when calculating playlist popularity.
        ii. For example, the "score" of this playlist: [`"Rap", "Rap", "Rap", "Rap", "Rap", "Pop"`] using the `genres` array of [`"Rap", "Pop", "Classical", "Hip Hop", "Rock", "Lo Fi"`] would be 1 because there are 5 `"Rap"` songs which are located at index 0 in `genres` and there is 1 `"Pop"` song which is located at index 1 in `genres`. **5*0 + 1*1 = 1.**
    b. If `playlist1`'s score is greater than `playlist2`'s score, print "`The first playlist is likely to have more listeners than the second.`"
    c. If `playlist2`'s score is greater than `playlist1`'s score, print "`The second playlist is likely to have more listeners than the first.`"
    d. If the two scores are equal, print "`The two playlists are predicted to perform equally well.`"
5. Create a **main** method.
    a. Create a `String[]` named `genres` with the following values: "`Rap`", "`Pop`", "`Classical`", "`Hip Hop`", "`Rock`", and "`Lo Fi`".
    b. Declare two `String[]` variables named `playlist1` and `playlist2`, respectively.
        i. These arrays should have a length of 7 and be filled with random values from the `genres` array you already declared.
        ii. Note: you don't have to use `Math.random()`, you can pick whatever values you want.
    c. Create a **Scanner** object named `scan`.
    REMINDER: This is the **ONLY** instance of **Scanner** you should create!
    d. Print the numbers of songs on each playlist and give it a name. This information should be input from the user.
    HINT: you have already created the method that does this.
    e. Print out the "Tally" of genres found within each playlist.
    HINT: you have already created the method that does this.
    f. Compare the popularity of the playlists to see which one is likely to have more listeners.
    HINT: you have already created the method that does this.

Example Outputs

Please refer to the PE clarification thread on Ed for example.

©CS1331

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Import Restrictions

To prevent trivialization of this assignment, the **only** allowed import is `java.util.Scanner`

Checkstyle

You must run Checkstyle on your submission (To learn more about Checkstyle, check out cs1331-style-guide.pdf under CheckStyle Resources in the Modules section of Canvas.) **The Checkstyle cap for this assignment is 5 points. This means there is a maximum point deduction of 5.** If you don't have Checkstyle yet, download it from Canvas -> Modules -> CheckStyle Resources -> checkstyle-8.28.jar. Place it in the same folder as the files you want to run Checkstyle on. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off (limited by the Checkstyle cap mentioned in the Rubric section). In future PEs and Homeworks, **we will be increasing this cap**, so get into the habit of fixing these style errors early!

For additional help with Checkstyle see the CS 1331 Style Guide.

Collaboration

Only discussion of the PE at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going **too far**. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs, and this exchange should be private.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- It is expected that everyone will follow the Student-Faculty Expectations document, and the Student Code of Conduct. The professor expects a positive, respectful, and engaged academic environment inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. **No inappropriate language is to be used, and any assignment, deemed by the professor, to contain inappropriate, offensive language or threats will get a zero**. You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.

@CS1331

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Playlist.java`

Make sure you see the message stating "PE4 submitted successfully". You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission so be sure to **submit every file each time you resubmit**.

Make sure you see the message stating the assignment was submitted successfully. From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder test are provided as a courtesy to help "sanity check" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or Professor via Piazza for clarification.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

If an autograder is enabled for this assignment, you may be able to see the results of a few basic test cases on your code. Typically, tests will correspond to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- Provide basic formatting and usage validation

In other words, **the test cases on Gradescope are by no means comprehensive**. **Be sure to thoroughly test your code** by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

CS1331