Programming Exercise 03 - Scanner, Static Methods, and Arrays

Authors: Vinayak, Ruchi, Ricky

Problem Description

This assignment will assess your knowledge of Scanner, Static Methods, and Arrays in Java.

Now that you are in university, it's time to start planning what courses you are going to take next semester. However, you want to be able to take coursework that will not be too difficult so you can put an adequate amount of time into each course to obtain a certain depth of knowledge in each course's material. Fortunately, you know that Java will be able to help you out in this scenario. In this PE, you will consider the difficulty of courses you would like to undertake next semester to create a prospective list of courses that you might be able to take in your imaginary university.

Some important notes for this assignment:

I. Note that interval notation is used throughout:
   a. **`[a,b]`** means a number **x** such that **`a <= x <= b`**.
   b. **`(a,b]`** means a number **x** such that **`a <  x <= b`**.
   c. **`[a,b)`** means a number **x** such that **`a <= x <  b`**.
   d. **`(a,b)`** means a number **x** such that **`a <  x <  b`**.

II. Every command-line input must occur on a new line. The user input should also be typed on the same line. For example, asking for the schedule owner's name should look like the following:

```
Enter this schedule owner's name: George P. Burdell
```

III. You **<u>must</u>** reuse code when possible. Look for hints suggesting code reuse.

IV. **Only create <u>ONE</u> Scanner object** (when prompted in the main method) and reuse it whenever you need to read more input. **Creation or use of more than one Scanner may result in a ZERO for the assignment as it "breaks" our tests!**

Solution Description

1. Create a class named `Schedule`.
2. Create a **public static** method named `generateSchedule` that takes in two parameters: a `String[]` named `subjectList` which represents all of the possible subjects that can be taken and an `int` named `numClasses` which represents the number of classes that will be taken in the schedule. This method should return a `String[]` that represents the subjects that will be in the schedule.
   a. Create a `String[]` named `schedule` that has `numClasses` length.
   b. Iterate through `schedule` using a `for` loop. Inside the `for` loop:
      i. If the index is even, set the value of `schedule` at that index to the value of `subjectList` at a random index in the range [0,3] (use `Math.random()` to accomplish this).

    ii. If the index is odd, set the value of `schedule` at that index to the value of `subjectList` at a random index in the range [4,6] (use `Math.random()` to accomplish this).

  c. Return `schedule`.

3. Create a **public static** method named `computeDifficulties` that takes in the following two parameters: a `String[]` named `schedule`, which represents the subjects found within the schedule and another `String[]` named `subjectList`, which represents all the possible subjects that can be found within a schedule. This method should return an `int[]` that represents the difficulty of each subject within the schedule. Within this method:
    a. For each `String` in `schedule`, **loop** through the `subjectList` and generate a difficulty for each subject in `schedule`. At the **same index** that the subject is found in `schedule`, place the difficulty of each subject in the `int[]` that will be returned.
        i. To calculate the difficulty of a subject, add the index of that subject in `subjectList` and the index of that subject in `schedule`.
        ii. **HINT:** Make sure that this method works for **any** size `schedule` it receives.
    b. Return the `int[]` array.

    For example, if `subjectList` contains `["English", "History", "Math", "CS", "Science", "Engineering", "Lab"]` and `schedule` contains `["History", "CS", "English", "Math"]`, the returned array should contain the following values: `[1, 4, 2, 5]`.

    `History` has a difficulty of 1 because **0 (index in** `schedule`**) + 1 (index in** `subjectList`**) = 1** and it is stored at index 0 since it appears at index 0 in `schedule`.

4. Create a **public static** method named `compareDifficulties` that takes in three `String[]` parameters named `schedule1`, `schedule2`, and `subjectList`, in that order. This method should return nothing. Within this method:
    a. Calculate the total difficulties of `schedule1` and `schedule2`. To calculate the total difficulty, take the sum of each element within the array returned by `computeDifficulties`. You **must** call this method.
    b. If `schedule1`'s difficulty is greater than `schedule2`'s difficulty, print

    ```
    The first schedule is harder than the second schedule.
    ```

    c. If `schedule2`'s difficulty is greater than `schedule1`'s difficulty, print

    ```
    The second schedule is harder than the first schedule.
    ```

    d. If the two difficulties are equal, print

    ```
    The two schedules are equally difficult.
    ```

5. Create a **main** method.
    a. Create a Scanner object named `scan` that will read input from the console.
    REMINDER: This is the ONLY instance of Scanner you should create!
    b. Print to the console:

    ```
    Enter the number of classes:
    ```

c. Read the user's input as an `int` and assign it to an `int` named `numClasses`.
d. Print to the console:

```
Enter this schedule's owner's name:
```

e. Read the user's input as a `String` and assign it to a `String` named `owner`. The owner's name may contain spaces, so the entire line of user input should be read in (e.g., "Mary Ann").
f. Create a `String[]` named `subjectList` with the values `"English"`, `"History"`, `"Math"`, `"CS"`, `"Science"`, `"Engineering"`, and `"Lab"`, in that order.
g. Declare two `String[]` variables named `schedule1` and `schedule2`. Generate a schedule for each array that contains `numClasses` classes.
   HINT: You have already written a method that does this.
h. Print out the following to the console:

```
Schedules created successfully. Here are the details:
Owner: {owner}
Number of Classes: {numClasses}
```

i. Iterate through each of the schedule arrays and print out each subject as well as its corresponding difficulty on a new line. Ensure to print out the name of the schedule for which you are printing out the subject and the difficulty level prior to iterating through the array. For example, if you were about to start iterating through `schedule1`, you would first print out:

```
Schedule 1:
```

If we implement the same example in Step 3b, where `schedule1` in this case would contain `["History", "CS", "English", "Math"]`, and its corresponding array holding each of the difficulty values would be `[1, 4, 2, 5]`, the following would be printed out:

```
Schedule 1:
History 1
CS 4
English 2
Math 5
```

Make sure to include a new line **before** printing the title of the schedule (i.e., Schedule 1 or Schedule 2).
   HINT: You have already written a method that provides significant help.
j. Compare the difficulties of the schedules to see which schedule is more difficult.
   HINT: You have already written a method that does this.

Example Outputs

Please refer to the corresponding clarification thread on the course forum for examples.

HINT: To help debug your code, try inserting print statements in places where variables are changed.

Checkstyle

You must run checkstyle on your submission (To learn more about Checkstyle, check out cs1331-style-guide.pdf under the Checkstyle Resources module on Canvas.) **The Checkstyle cap for this assignment is 0 points.** This means there is a maximum point deduction of 0. 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 assignments 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.

Turn-In Procedure

Submission

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

- `Schedule.java`

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 test" 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 the class forum 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 assignment (submit early and often). We will only grade your latest submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. 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:

1. Prevent upload mistakes (e.g., forgetting checkstyle, non-compiling code)
2. 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.

Allowed Imports

- `java.util.Scanner`

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`

Collaboration

Only discussion of the assignment 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.

You **MAY NOT** use code generation tools to complete this assignment. This includes generative AI tools like ChatGPT.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- **Check on Ed Discussion for a note containing all official clarifications and sample outputs**

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. **<u>No inappropriate language is to be used</u>, 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.