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

Authors: Melanie, Aqdas, Stephanie, Owen, Sooraj, Rachna

Problem Description

In consideration of your first exam approaching, CS1331 is organizing study groups to encourage *academically honest* collaboration between students to better understand course concepts and review for the exam. To ensure these groups will be helpful for every student, we need your help in analyzing group dynamic! This assignment will test your basic knowledge of enums, static methods, and arrays.

Solution Description

1. Create a **class** called `StudyBuddies`
2. Create an enumeration (enum) called `Student`
   a. Add the following values to the `enum`: `PROCRASTINATOR`, `CURVE_BREAKER`, `LATECOMER`, `DAYDREAMER`, `TROUBLEMAKER`, and `QUIET_ACHIEVER`
   b. This should be in a separate java file from `StudyBuddies`
3. Inside `StudyBuddies`, create a public, **static method** called `groupFinder` that does not take in any parameters and returns a `Student` array. Within this method:
   a. Create a `Student` array of size 6 called `studentArray`
   b. At each index of `studentArray`, place a **random value** of `Student`
      i. Note that `EnumName.values()` returns an array of the values inside an enum. How can you use it along with `Math.random` to get random values of type `Student`? You should not just return `EnumName.values()`.
      ii. Each time this method is called, a different array should be returned.
   c. Return the `studentArray`
4. Inside `StudyBuddies`, create a public, **static method** called `groupAnalyzer` that takes in a variable named `studentArray` of which is an array of `Student` and does not return anything. Within this method:
   a. Print the following statement once: "`The following types of students are in your study group:`"
   b. *For each* type of `Student`, count how many are present in `studentArray`. Then, print out a new line with the string representation of each `Student` and the total number of that type of student.
      i. For example, given the array `[PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, LATECOMER]` as a parameter, you should print out

```
PROCRASTINATOR 5
CURVE_BREAKER 0
LATECOMER 1
DAYDREAMER 0
TROUBLEMAKER 0
QUIET_ACHIEVER 0
```

   c. Make sure that this method works for any size `studentArray` it receives

5. Inside `StudyBuddies`, create a public, **static method** called `groupScores` that takes in two variables named `group1` and `group2` (in that order and both of type `Student[]`) and does not return anything. Within this method:
    a. Calculate the "score" of each group by finding the ordinal position of each `Student` in the group and adding them together
        i. Note that `EnumValue.ordinal()` will return the ordinal of `EnumValue`, the position in which `EnumValue` was originally declared in the enum.
        ii. For example, the "score" of this array:
        `[PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, PROCRASTINATOR, LATECOMER]`
        would be 2 because there are 5 `PROCRASTINATOR`s which are located at position 0 in the enum, and there is 1 `LATECOMER` which is located at position 2 in the enum.
        5*0 + 1*2 = 2.
        There are a few different ways to make this calculation in the code. Feel free to pick what makes the most sense to you.
    b. If group1's score is greater than group2's score, print "`The former is predicted to perform better than the latter.`"
    c. If group2's score is greater than group1's score, print "`The latter is predicted to perform better than the former.`"
    d. If the two scores are equal, print "`The two groups are predicted to perform equally well.`"
6. Inside `StudyBuddies`, create the `main` method and within it:
    a. Declare two arrays of type `Student[]` named `group1` and `group2`, respectively.
    b. Test your static methods by using them to:
        i. Fill these groups with students
        ii. Print out the contents of each group
        iii. Compare `group1` to `group2` in terms of their group dynamics by estimating their score value

Example Outputs

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

Checkstyle

Starting from PE04, you will begin implementing 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 0 points.** This means you will not lose points for Checkstyle errors, but keep in mind this cap will be raised for future homeworks. 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 above). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Turn-In Procedure

Submission

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

- `StudyBuddies.java`
- `Student.java`

Make sure you see the message stating "PE04 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder tests are provided as a courtesy to help "sanity test" your work and you may not see all of 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 Ed 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 last 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. (We are not using checkstyle for this programming exercise, but in future HWs we will).

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.

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

You may not import anything for this homework assignment.

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.

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 all official clarifications
- It is expected that everyone will follow the Student-Faculty Expectations Agreement (see syllabus), 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.