Programming Exercise 04 – Healthy Eating

Authors: Lucas, Kenneth, Ruchi, Daniel, Ricky, Sang Yoon
Topics: Enums, Static Methods, Arrays

Problem Description

Having taken an APPH course recently, you have grown increasingly aware of your eating habits. As a result, you decide to write a program to evaluate the different food options around campus. With this tool, you will hopefully make the healthy choice!

This assignment will test your basic knowledge of enums, static methods, and arrays.

Solution Description

Healthy Eating

1. Create a **class** called `HealthyEating`
2. Create an enumeration (**enum**) called `Food`
   a. Add the following values to the enum: `JUNK_FOOD`, `DAIRY`, `GRAIN`, `PROTEIN`, `FRUIT`, and `VEGETABLE`
   b. This should be in a separate Java file from `HealthyEating`
3. Inside `HealthyEating`, create a `public`, **static** method called `mealPrep` that takes in an integer named `numFoods` and returns a `Food` array. Within this method:
   a. Create a `Food` array of size `numFoods` called `foodArray`
   b. At each index of `foodArray`, place a **random value** of `Food.`
      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 `Food`? You should not just return `EnumName.values()`.
      ii. Each time this method is called a different array should be returned.
   c. Return the `foodArray`
4. Inside `HealthyEating`, create a `public`, **static** method called `followRecipe` that takes in a `String` named `recipe` and returns a `Food` array. Within this method:
   a. A `recipe` is a string of enum names separated by spaces.
      i. For example, a `recipe` may look like:
      `"PROTEIN GRAIN FRUIT VEGETABLE"`
      and the returned `Food` array would look like:
      `[PROTEIN, GRAIN, FRUIT, VEGETABLE]`
      ii. **HINT:** The String API provides a method named `split(…)` that can help you separate the individual foods so you can place them in an appropriately-sized array, and `EnumName.valueOf(String)` returns an enum value that matches the `String`.
   b. You may assume that the provided String will only contain valid names of foods from the enum you created.

5. Inside `HealthyEating`, create a `public`, **`static`** method called `mealAnalyzer` that takes in a variable named `foodArray`, which is an array of `Food`, and does not return anything. Within this method:
   a. Print the following statement once: "`The following types of food are in your meal:`"
   b. *For each* type of `Food`, count how many are present in `foodArray`. Then, print out a new line with the string representation of each `Food` and the total number of that type of food.
      i. For example, given the following array as a parameter:
      `[JUNK_FOOD, JUNK_FOOD, JUNK_FOOD, JUNK_FOOD, VEGETABLE, GRAIN]`
      you should print out:
```
JUNK_FOOD 4
DAIRY 0
GRAIN 1
PROTEIN 0
FRUIT 0
VEGETABLE 1
```
   c. Make sure that this method works for any size `foodArray` it receives.
6. Inside `HealthyEating`, create a `public`, **`static`** method called `healthyChoice` that takes in two variables named `meal1` and `meal2` (in that order and both of type `Food[]`) and does not return anything. Within this method:
   a. Calculate the "score" of each meal by finding the ordinal position of each `Food` 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:
      `[JUNK_FOOD, JUNK_FOOD, JUNK_FOOD, JUNK_FOOD, VEGETABLE, GRAIN]`
      would be 7 because there are 4 `JUNK_FOOD`s which are located at position 0 in the enum, there is 1 `GRAIN` which is located at position 2 in the enum, and there is 1 `VEGETABLE` which is located at position 5 in the enum.

$$4 \times 0 + 1 \times 2 + 1 \times 5 = 7$$

      There are a few different ways to make this calculation. Feel free to pick what makes the most sense to you.
   b. If meal1's score is greater than meal2's score, print "`The first meal is the healthier choice with a score of <meal1 score>.`"
   c. If meal2's score is greater than meal1's score, print "`The second meal is the healthier choice with a score of <meal2 score>.`"
   d. If the two scores are equal, print "`The two meals are equally healthy with scores of <score>.`"
7. Inside `HealthyEating`, create the `main` method and within it:
   a. Declare two arrays of type `Food[]` named `meal1` and `meal2`, respectively.
   b. Test your static methods by using them to:
      i. Fill these groups with `food`

ii. Print out the contents of each `meal`
iii. Compare `meal1` to `meal2` in terms of their nutritional value
iv. Use `followRecipe()` to create a `meal1` and `meal2` that test all possible outcomes (i.e. `meal1` healthier, `meal2` healthier, same score).

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 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 number of points we would take off (limited by the Checkstyle cap). 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:

- `HealthyEating.java`
- `Food.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 tests 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 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. 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. **We reserve the right to hide any or all test cases, so you should make sure to test your code thoroughly against the assignment's requirements.**

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.

Burden of Testing

You are responsible for thoroughly testing your submission against the written requirements to ensure you have fulfilled the requirements of this assignment.

Be **very careful** to note the way in which text output is formatted and spelled. Minor discrepancies could result in failed autograder cases.

If you have questions about the requirements given, reach out to a TA or Professor via the class forum for clarification.

Allowed Imports

To prevent trivialization of the assignment, you may **not** import any classes for this 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 autograder. For that reason, do not use any of the following in your final submission:

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

Collaboration

Only discussion of the assignment at a conceptual high level is allowed. You can discuss course concepts and 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 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. **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.