Programming Exercise 3 - Randoms, Math, Enums, and Static Methods

Problem Description

"Water (stay hydrated). Earth (Nave Food). Fire (My mixtape). Air (please shower so it smells nice). Long ago, the four aspects of freshman life at Georgia Tech lived together in harmony. Then, everything changed when Georgian Tech leaked our data. Only the Sophomore by credit hours, master of all four aspects, could save us, but when the world needed her most, she graduated. A few semesters passed and my fellow TA's and I discovered a new hope, you! And although your programming skills are great, you have a programming exercise to complete before you're ready to save anyone. But I believe you can save the world."

Using Randoms, `Math`, `Enums`, and Static Methods, we will help protect Georgian Tech.

Solution Description

- Create the class
  1. Create a class called `DataLeakSimulator`
- Types of Data Enum
  1. Create an enum called `SchoolDataType`
  2. Add the following values to the enum: `STUDENTS`, `CLASSES`, `HOUSING`, `DINING` (remember to captialize them)
- NumberParser
  1. Create a static method `leakData` in `DataLeakSimulator`. It should take an int as a parameter and return a `SchoolDataType` value.
  2. The method should follow the below process for determining the type of data to leak: 1. Take the absolute value of the number (using `Math.abs`)
     2. Mod the number in step 1 by 4.
     - If the result is 0, return `STUDENTS` from `SchoolDataType`
     - If the result is 1, return `CLASSES` from `SchoolDataType`
     - If the result is 2, return `HOUSING` from `SchoolDataType`
     - If the result is 3, return `DINING` from `SchoolDataType`
- Main Method
  1. Add a main method to `DataLeakSimulator`
  2. In your main method, create a `Random` variable called myRandom and assign to it a new `Random` object.
  3. Generate a int between 0 and 25 (inclusive) using `myRandom.nextInt(26)`.
  4. Take the square root of the number generated in step 3 (using `Math.sqrt`) and cast the result to an int.
  5. Call the `leakData` method with the number generated in step 4 as a parameter. Print the result out in the format of "[data type name] data was leaked" with the resulting value.
  6. Generate an int between -10 and 10 using `Math.random()`. The general formula to do this is `(int)(Math.random() * ((Max - Min) + 1) + Min)`.
  7. Call `leakData` method with the number generated in step 6 and print out "[data type name] data was leaked" with the resulting value.

Testing your app

This is an example of running the main method from the command line.

```
--> java DataLeakSimulator
STUDENTS data was leaked
DINING data was leaked
```

Allowed Imports

To prevent trivialization of the assignment, you are *only* allowed to import java.util.Random.

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

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

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

- DataLeakSimulator.java
- SchoolDataType.java

Make sure you see the message stating "PE03 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

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: 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. Each test typically corresponds 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:

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.

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 Piazza for all official clarifications