Programming Exercise 04 – Javangers: Integer.MAX_VALUE War

Problem Description

**pans across screen and onto a throne* Jaqob the head TA surveys the gradescope submissions lacking Collaboration Statements from atop. Slowly he gets off. Grasping his 64-bit Grading Gauntlet he mutters "fine, I'll do it myself."*

When the Grading Gauntlet has the Grading stone in it, the owner can snap his fingers and make half of all submissions have a collaboration statement.

Hello, and welcome! Please make sure to read all parts carefully.

For programming exercise 04 you will be creating an enumeration (`enum`) and a `static` method utilizing arrays.

Solution Description

1. **Create an enumeration file** called `StoneType.java` – do the following in the file:

    1. Create a `public enum` called `StoneType`
    2. Add the following values to the enum: `SHINY, SMALL, BIG, GRADING` (make sure to capitalize them)

2. Create a class file called `HeadTA.java` – do the following in the file:

    1. Create a class called `HeadTA` and do the following in the class
    2. Create a `static` method with the following signature:

        ```
        giveCollabStatement(StoneType[] stones, int rows, int cols)
        ```

        It should do the following:

        a. Loop through the `stones` array, searching for `GRADING` enumeration value `StoneType`
            i. If array does not contain `GRADING`, print "`Cannot Grade`" and do not do steps b-c.
            ii. If the array does contain `GRADING`, print "`Snaps Fingers`" and do steps b-c.
        b. Initialize a 2D boolean array called `collabStatementArray` with `rows` number of rows and `cols` number of columns. This means a 2D array containing `rows` number of arrays, each with `cols` elements. Think about what the current value is of each element in the array.
        c. Looping through `collabStatementArray` row-by-row, set every other element to `true`, starting with the second element. Continue alternating from the previous row (if the previous row ends with `false`, the new row should start with `true`). Then, print out T if the element is true or F if the element is false. For a 2 row, 3 column array, `collabStatementArray` should end up with the following values:

`false`	`true`	`false`
`true`	`false`	`true`

For the same 2 row, 3 column array, `collabStatementArray` should print the following:

```
FTF
TFT
```

3. Create a `main` method in `HeadTA` to do the following:
    a. Create an array of type `StoneType` called `myStones` with length 4.
    b. Assign each of the values of `StoneType` to `myStones`. This means `myStones` should hold `SHINY, SMALL, BIG, GRADING.`
    c. Call `giveCollabStatement` and pass in `myStones` as the first parameter and 3 and 4 as `rows` and `cols`, respectively
    d. Create an array of type `StoneType` called `noStones` with length 0.
    e. Call `giveCollabStatement` and pass in `noStones` as the first parameter and 5 and 5 as `rows` and `cols`, respectively

Example Outputs

For the provided main method, the program should print the following:

```
Snaps Fingers
FTFT
FTFT
FTFT
Cannot Grade
```

Rubric

[15] **StoneType.java**

- [7] StoneType is a public enum
- [8] StoneType has the specified values

[85] **HeadTA.java**

- [60] `giveCollabStatement` is correct
    - [10] Loops through student array
    - [5] Correct output if `GRADING` not found
    - [15] Correct output if `GRADING` is found
    - [10] `collabStatementArray` is a 2D array with proper dimensions
    - [10] Alternating T/F in `collabStatementArray`
    - [10] `collabStatementArray` is printed properly
- [25] Main Method is correct
    - [5] `myStones` is an array of type `StoneType` with length 4
    - [5] `myStones` is assigned every value of `StoneType`
    - [5] `main` calls `giveCollabStatement` with parameters: `myStones, 3, 4`
    - [5] `noStones` is an array of type `StoneType` with length 0
    - [5] `main` calls `giveCollabStatement` with parameters: `noStones, 5, 5`

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes. Points will be deducted for checkstyle errors, lateness, a missing collaboration statement, and illegal imports.

Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **5** points. If you don't have checkstyle yet, download it from Canvas -> Files -> Resources in the A/B/GR section and in Modules -> General Information in the D section . Place it in the same folder as the files you want checkstyled. 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!

Allowed Imports

You may not import anything for this homework 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`

Collaboration

Collaboration Statement

To ensure that you acknowledge a 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].

Allowed Collaboration

When completing homeworks and programming exercises for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks or programming exercises
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- **approved**: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"
- **disapproved**: "Hey, it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

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

- HeadTA.java
- StoneType.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.

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:

- 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 or run 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 a note containing all official clarifications