Programming Exercise 4 - Classes, Constructor Chaining, Arrays

Purpose Statement

This programming exercise provides you some initial experience working with arrays and creating programs that are more object-oriented than our prior ones. You'll build a class and work with instance data and non-static methods. We're finally on our way toward using key o-o principles like encapsulation. This programming exercise will give you practice implementing the key concepts that we've discussed in lecture during the past week.

Problem Description

"Georgia Tech students are known for their creativity and entrepreneurship. Two GT students became disenchanted with the local pizza options and decided to open their own PizzaShop, a PizzaShop that incorporated some of the knowledge they had acquired at Georgia Tech. They wanted to use software to track pizza orders. However, their Java skills were rusty, so they decided to enlist the help of students in their favorite computer science class, CS1331."

Your job is to create a Pizza class that creates a Pizza Object for an order.

Solution Description

1. Create a class called Pizza
2. Pizza has 4 instance variables:
   - An int variable called `numSlices` that stores the number of slices to cut the pizza into
   - A String array called `toppings` that stores all the toppings as Strings
   - A boolean variable `hasCheese` that stores whether or not the pizza has cheese on it (Cheese does not count as a topping)
   - A boolean variable `isGlutenFree` that stores whether or not the pizza is gluten free.
   - The default for a regular pizza is cut into 8 slices, has 0-toppings, cheese, and is not gluten-free.
3. Pizza has 2 constructors
   - A constructor that doesn't take in any arguments and makes a pizza with the default values for a regular pizza (A pizza with zero toppings should have an empty topping array)
   - A constructor that takes in a request for how many slices, a value for cheese, a value for gluten free, and a String array of toppings (in that order)
4. Pizza has the following methods:
   - A method that returns the array of toppings and a method that sets the of toppings to a passed in array:
     - `getToppings`
     - `setToppings`
   - A method that returns the number of slices and a method that sets the number of slices to a passed in value:
     - `getNumSlices`
     - `setNumSlices`
   - A method that sets the hasCheese field to a passed in value:
     - `setHasCheese`
   - A method that sets the glutenFree field to a passed in value:
     - `setIsGlutenFree`
   - a method `printToppings` that prints out the array of toppings in the following format:

Pizza Toppings:
{*topping 1*}
{*topping 2*}
...

- a method `changeFirstTopping` that takes in a String and changes the first element in the topping array (if the array is of sufficient length) to be the String passed in. If the array is not of sufficent length, this method should do nothing. This method should not return anything.

Notes & Hints

- variable names should be exactly as specified
- getters and setters should follow java naming conventions
- constructors should take the arguments in order given
- arrays have a length value that can be used to get the number of elements they have
- arrays can also be of length 0, but there are implications of this

Allowed Imports

To prevent trivialization of the assignment, you are **not** allowed to import: - java.util.ArrayList

Please ask on Piazza about any other things you wish to import.

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:

- `Pizza.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:

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