Programming Exercise 05 – Sandwich Shop

Authors: Akul, Vince, Ryan, Tejas, Julia, Connor, Andrew

Problem Description

This programming exercise provides you some experience with creating programs that are object-oriented. You will build a class and work with instance data and non-static methods to help create sandwiches for the CS 1331 Sandwich Shop.

Solution Description

You will have one file named `GTSandwich.java` which will create a `GTSandwich` object for the sandwich artists to look at and create for the customers.

GTSandwich.java

1. Create a class called `GTSandwich`
2. `GTSandwich` has 7 private instance variables:
    - A `String` variable called `bread` that stores the type of bread the customer wants.
    - A `String` variable called `meat` that stores the meat the customer wants.
    - A `String array` variable called `extras` that stores the various vegetables, sauces, or extras the customer wants.
    - An `int` variable called `numMeat` that stores the number of meats the customer wants.
    - A `double` variable called `price` that stores the cost of the sandwich.
    - A `boolean` variable called `hasSauce` that stores whether the sandwich has the special GT house sauce.
    - A `boolean` variable called `makeCombo` that stores whether the customer wants to make the sandwich a combo.
3. `GTSandwich` has these constructors:
    - A constructor that takes in `bread`, `meat`, `extras`, `numMeat`, `price`, `hasSauce`, and `makeCombo` in this exact order. The constructor should correctly initialize the variables to the values passed in.
    - A constructor that takes in `bread`, `meat`, and `makeCombo` in this order. This constructor sets the following defaults:
        - number of meats should be 4,
        - cost should be `8.75`,
        - should have the **special house sauce**,
        - and there should be **no extras** (a sandwich that has no extras should have an empty array).
    - A zero-argument constructor which sets the following defaults:
        - `bread` is Rye,
        - choice of meat is Turkey,
        - no extras (A sandwich that has no extras should have an empty array for extras),

    - 4 slices of meat,
    - costs 8.75,
    - has the special house sauce,
    - and the sandwich is a combo in this order.
- **Must use constructor chaining when possible**.

4. The `GTSandwich` class has the following methods:

    - Accessors/Getters and Mutators/Setters **for every instance variable**. Accessors return the value of private instance variables. Mutator methods are `void`, take in a parameter of the same type as the instance variable, and sets the value for that variable.
    - The mutator for the `makeCombo` instance field, in addition to setting `makeCombo`, should increase the cost of the sandwich by 2 dollars if a `true` value is passed in and the current value of `makeCombo` is `false` (if it is `true` and a `true` value is passed in, do nothing). If a `false` value is passed in and the current value of `makeCombo` is `true`, then decrease the cost of the sandwich by 2 dollars (if the current value of `makeCombo` is `false` and a `false` value is passed in, do nothing).
    - Method called `applyCoupon` that reduces the cost of the sandwich and increases the slices of meat. This method takes in a String representing a coupon code. If the coupon code is "CS1331" and the sandwich is a combo then you can apply the discount. The cost of the sandwich is reduced by 50% and the slices of meat are doubled and `"Coupon Applied"` is printed to the console. If the sandwich is not a combo or the coupon code is not correct then this method prints out `"Coupon Not Applied"` and does not apply the coupon.
    - Method called `applyCoupon` that reduces the cost of the sandwich and makes the sandwich a combo for free. This method overloads the method described above. This method does not take in anything. If the cost is greater than or equal to 20 dollars, then the cost of the sandwich is reduced by 5 dollars and the order is made a combo (if the order is already a combo you should only apply the discount). If the discounts and changes are applied, then the method prints out `"Discount Applied"` otherwise it prints out `"Discount Not Applied"`.
    - Method called `printExtras` that prints out the extras a customer wants on their Sandwich. If the customer has no extras, then this method should print:

      ```
      The customer wants no extras.
      ```

      Otherwise, it should print:

      ```
      The customer wants these extras: {extra 1}, {extra 2}, {extra 3}...
      ```

    - A `toString` method that prints out a String representation of a Sandwich object in the following way. This method should have no parameters and return a `String`. The price should be rounded to 2 decimal places.

      ```
      GT Sandwich that costs {price} with {bread}, {meat}, {extras}, {numMeat} slices of meat, Sauce: {hasSauce}, Combo: {makeCombo}.
      ```

Example Outputs:

In your main method do the following:

```
GTSandwich houseSpecial = new GTSandwich("French", "Turkey", new String[]{"Banana Peppers", "Spinach", "Chipotle Mayo"}, 4, 10.75, true, true);
```

```
GTSandwich javaSandwich = new GTSandwich();
houseSpecial.applyCoupon("CS1331");
javaSandwich.applyCoupon();
houseSpecial.printExtras();

System.out.println(houseSpecial);
System.out.println(javaSandwich);
```

This would print the following to the console:

```
Coupon Applied
Discount Not Applied
The customer wants these extras: Banana Peppers, Spinach, Chipotle Mayo
GT Sandwich that costs 5.38 with French, Turkey, Banana Peppers, Spinach,
Chipotle Mayo, 8 slices of meat, Sauce: true, Combo: true
GT Sandwich that costs 8.75 with Rye, Turkey, 4 slices of meat, Sauce: true,
Combo: true
```

Checkstyle and Javadocs

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. Place it in the same folder as the files you want checkstyle. 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 points we would take off (limited to the amount we specified above). The Java source files we provide contain no Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Allowed Imports

To prevent trivialization of the assignment, you may not import anything.

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

No collaboration is allowed on this assignment. See syllabus for more details.

In addition, 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:

- GTSandwich.java

Make sure you see the message stating "PE05 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, 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
- Run Checkstyle on your code to avoid losing points
- 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