Homework 3 - BuzzCard Class

Introduction

In this homework, you will practice:

- implementing classes
- using constructors
- instance methods

Problem Description

You are hungry after taking an infamous CS1331 exam, so you decide to go get food from the many dining options on campus. Unfortunately, the point of sales service goes down. Luckily, you are a great Java programmer and can make one for the school!

Solution Description

Download hw3.zip (/spring2018/hw3/hw3.zip) and complete the Student and BuzzCard classes that implements the point of sales systems for the three restaurants provided: Subway, Brittain, and Burdells. We have included the skeletons for both classes. The implementation guide can be found below

Do not modify the code we give you.

BuzzCard.java

This class has the following private fields, **and associated getter and setter methods** for them:

- `mealSwipes` which is an int that represents the number of meal swipes left
- `diningDollars` which is a double that represents the amount of dining dollars left.
- `buzzFunds` which is a double that represents the amount of buzzfunds left.

Make sure to make the three instance fields private so that only your methods within the class has access to them!

This class has the following constructors:

One that takes in an int for the Meal Swipes, a double for the Dining Dollars, and a double for the BuzzFunds (in that order) and sets their corresponding instance fields.

This class has the following public methods:

- `String toString()` which returns the `String` representations of the student's in the following format:

`"Buzzcard balance with Dining Dollars: (diningDollars), BuzzFunds: (buzzFunds), Meal Swipes: (mealSwipes)"`

Student.java

This class has the following private fields, **and associated getter methods** for them:

- `card` which is a BuzzCard object that represents the student's BuzzCard.
- `name` which is a String that represents the name of the student.
- `id` which is an int that represents the gtID of the student.

This class has the following public methods:

- `void buyBrittainMealSwipes(BrittainItem item)` which will allow the student to buy items from Brittain Dining Hall using meal swipes. If the student does not have the amount needed to buy them item, print to the console `"You do not have the amount to buy this item :("`
- `void buyBrittainBuzzFunds(BrittainItem item)` which will allow the student to buy items from Brittain Dining Hall using BuzzFunds. If the student does not have the amount needed to buy them item, print to the console `"You do not have the amount to buy this item :("`
- `void buyBurdellsBuzzFunds(BurdellsItem item)` which will allow the student to buy items from Burdells using BuzzFunds. If the student does not have the amount needed to buy them item, print to the console `"You do not have the amount to buy this item :("`
- `void buySubwayDiningDollars(SubwayItem item)` which will allow the student to buy items from Subway using dining dollars. If the student does not have the amount needed to buy them item, print to the console `"You do not have the amount to buy this item :("`
- `void buySubwayBuzzFunds(SubwayItem item)` which will allow the student to buy items from Subway using BuzzFunds. If the student does not have the amount needed to buy them item, print to the console `"You do not have the amount to buy this item :("`
- For all of the above methods, decrement the appropriate value on the student's buzzcard.
- `String toString()` which returns the `String` representations of the student's information in the following format:

`"Student named (name) with ID: (id)"` + BuzzCard info from the BuzzCard toString().

This class has the following constructors:

One that takes in an a `BuzzCard` object and stores it in `card`, a `String` and stores it in `name`, and an `int` and stores it in `id`.

Grading

- [5] Properly declaring private variables for `BuzzCard` and `Student`.

- [5] Constructor for `BuzzCard`.
- [5] Constructor for `Student`.
- [5] Getter methods for `BuzzCard`.
- [5] Setter methods for `BuzzCard`.
- [10] `BuzzCard.toString()`.
- [5] Getter methods for `Student`.
- [10] `Student.buyBrittainMealSwipes()`.
- [10] `Student.buyBrittainBuzzFunds()`.
- [10] `Student.buyBurdellsBuzzFunds()`.
- [10] `Student.buySubwayDiningDollars()`.
- [10] `Student.buySubwayBuzzFunds()`.
- [10] `Student.toString()`.

Running and Testing

Creating a `Driver` class with a `main` method to create a simulation to test of all the methods would be the best course of action.

Tips and Considerations

If anything seems confusing, read through the entire description and instructions again. As always, feel free to contact your TAs, post on Piazza, or come in for office hours. In addition, here are some tips specific to this homework:

- You can assume input will be of the correct type (i.e, we will not break your code by giving you a String instead of an int), but not necessarily the correct values (i.e, if you ask for an int between 1 and 5 we might enter 9). Your code should handle these cases gracefully.
- Remember that char is actually a numeric primitive type. That means you can mix arithmetic involving chars and ints. For example:

Javadocs

Starting from this homework, you will need to write Javadoc comments and watch for checkstyle errors with your submission.

- Every class should have a class level Javadoc that includes `@author <GT Username>`.
- Every public method should have a Javadoc explaining what the method does and includes any of the following tags if applicable:
    - `@param <parameter name> <brief description of parameter>`
    - `@returns <brief description of what is returned>`
    - `@throws <Exception> <brief explanation of when the given exception is thrown>`

See the CS 1331 Style Guide (http://cs1331.gatech.edu/cs1331-style-guide.html) for details.

Checkstyle

For each of your homwork assignments we will run checkstyle and deduct one point for every checkstyle error.

For this homework the **checkstyle cap is 20**, meaning you can lose up to 20 points on this assignment due to style errors. This limit will increase with each homework.

- If you encounter trouble running checkstyle, check Piazza for a solution and/or ask a TA as soon as you can!
- You can run checkstyle on your code by using the jar file found on the course website that includes xml configuration file specifying our checks. To check the style of your coed run `java -jar checkstyle-6.2.2.jar *.java`.
- To check your Javadocs run `java -jar checkstyle-6.2.2.jar -j *.java`.
- Note that the command for checking code and the command for checking Javadocs are different. You will have to run both commands to fully test for style errors.
- Javadoc errors are the same as checkstyle errors, as in each one is worth a single point and they are counted towards the checkstyle cap.
- **You will be responsible for running checkstyle on *ALL* of your code.**
- Depending on your editor, you might be able to change some settings to make it easier to write style-compliant code. See the customization tips (http://cs1331.gatech.edu/customization-tips.html) page for more information.

Collaboration

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

_ What general strategies or algorithms you used to solve problems in the homeworks _ Parts of the homework specification 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:

OKAY: “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?”

BY NO MEANS OKAY: “Hey… the homework is due in like 20 minutes… Can I see your code? I *promise* won’t copy it directly!”

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.

Submission

- Submit your `Student.java` and `BuzzCard.java` files as attachments to the `hw3` assignment on Canvas. You can submit as many times as you want, so feel free to submit as you make substantial progress on the homework. We only grade your **last** submission, meaning we will ignore any previous submissions.
- As always, late submissions will not be accepted and non-compiling code will be given a score of 0. For this reason, we recommend submitting early and then confirming that you submitted ALL of the necessary files by re-downloading your file(s) and compiling/running them.