Homework 3

Taylor Hartman

1 Introduction

Given the skeleton of the new game Lemonade Tycoon, it is your job to implement the necessary features that are required for the game to run.

2 Problem Description

The game is to run through the command line. Game play is turn-based on a daily cycle. In one day (turn) the player has a couple of choices presented to them:

1. Menu actions
    - Check Menu
    - Add to Menu
    - Adjust Prices
2. End Day

A lot of the I/O has already been implemented for you, so make sure to look at the code and run the skeleton file to see what has already been completed for you. You only need to implement some methods and adjust some of the output. Good luck!

3 Solution Description

3.1 Menu Actions

The menu actions pertain to the drinks and ingredients that you provide at your lemonade stand. The menu itself will be a multidimensional array that holds Strings. You can make the initial size of your outer array to be any value you'd like, but the inner array should be of size 10 (you'll see why below!). Whenever a user inputs ingredients or the name of items they will be inputted as Strings. The menu actions and their behavior are as follows:

1. Checking the Menu
    (a) If a user wants to check the menu, you need to print out the menu items in a clean way. An example output might look something like below:

```
Lemonade: Lemon, Sugar, Ice, Water $0.50
Pink Lemonade: Lemon, Blueberry, Sugar, Ice $0.75
```

Note that it doesn't have to be in this exact order, you can have price first, name second, ingredients third if you like that better. Just make sure each subcategory is grouped together and it is formatted neatly!

(b) Notice that the price is displayed like an actual monetary value. Make sure your output formats this correctly.

2. Adding to the Menu

   (a) If a user wants to add to the menu, you need to gather 3 pieces of information
   - Name of the item
   - Ingredients
   - Price

   (b) After receiving input, add this item to the menu. The names of the items do not have to be in any certain order, just add them to the next available space. You can add the price in at the very end.

   (c) You need to do 2 things with the price. You need to add it to the array for the menu so that you can output it when you check the menu as shown above, but you also need to add the price of the item to the `menuPriceTotal`.

   (d) It is not ok to have multiple items by the same name. Check through the menu to make sure an item of that name does not already exist. If it does exist, print out an error message to the console. Something along the lines of "That item already exists" is perfect.

   (e) Note that names of items are not case sensitive so an item called "Pink Lemonade" is the same as "pink lemonade" is the same as "PiNK lemOnAde".

   (f) If the menu is full, you need to resize it. So if I want to add a valid menu item but the array is full, copy everything over from the first array into a new array with more room so that I can add my item. Do not worry about resizing the inner arrays for ingredients. We will not put more than 8 ingredients in Lemonade. That's a little overkill.

   (g) The first thing you should think about given the above information is: what dimensions should my array be? Hint: it will be multidimensional, but at least one of your dimensions won't look like the others.

3. Adjusting Prices

   (a) The user needs to input the name of the item they want to adjust. Remember names are not case sensitive as mentioned above.

   (b) If the item is not in the menu, print out a message to the user and return them to the base menu.

3.2 Weather

Lemonade Tycoon has a weather specification which changes how well lemonade sells. There are 4 types of weather that are present in the game. At the beginning of each day, you need to determine what the weather is and notify the user.

1. Weather is determined through a pseudo-random number generator. Draw upon our discussion of Random in class, and of course, you can always look it up in the API. This is different from Math.random(). If you use Math.random() in this assignment, you will not receive points for your implementation of Weather.

2. You need to implement weather using the provided Weather enumeration in your code. Using the Random number generator, select the type of Weather per day from the enumeration and change the level of popularity, satisfaction, and sales with your stand as prescribed by the above description. Example: If the weather is sunny out, you should change your popularity to be 1.5 times the base popularity amount.

3. The four types of weather are:

   (a) Extra Sunny: Sunny weather causes more lemonade to be sold than normal. Lemonade popularity increase by a Random integer amount between 0 and 10.

   (b) Normal: Just a regular day in the neighborhood. This does not change how much lemonade is sold in the day. Instead, restore the sales rate back to 1, the popularity back to 50, and the satisfaction back to 50.

   (c) Cloudy: Lemonade sales do not change, but satisfaction decreases by a Random amount.

   (d) Raining: Rainy weather causes the Lemonade sales rate to decrease by a Random amount. Notice that sales rate is a multiplier and should be between 0 and 1 inclusive.

3.3 Money

At the end of each day, tell the player how much money they made. The method for calculating money has already been written for you and is called `getProfit`. Look in the provided files for this method's parameters and how to use it. Other than using this method in the right place and updating its dependent variables correctly, you don't need to worry too much about how the money is being calculated.

3.4 Details, Tips, and Tricks

- You will be using `java.util.Random` and some of its various methods. Whenever you want to use the `nextInt()` method, use the method by the same name which takes in a bound: `nextInt(int bound)`. Pass in 10 as the bound. This is for ease of grading purposes and if you do not do this you might get points taken off for not following directions.

- Make sure that your satisfaction, popularity, and salesRate are reasonable amounts. Make sure the bounds on your Random number calls are appropriate. Popularity and satisfaction should always be positive numbers and salesRate should always be between 0.0 and 1.0.

- There will be lots of looping during this homework. Make sure you fully understand how to write loops that go through multi-dimensional arrays; especially ones that don't look like grids!

- Feel free to consult outside sources for help! The TAs are always here to assist, and a quick Google search for some topic or concept can be beneficial as well.

- For those of you that like to procrastinate, don't. This homework might be a little tricky even though it doesn't sound like it. I'd hate for you to realize that on the day it is due.

4 Checkstyle

Checkstyle counts for this homework. You may be deducted up to 10 points for having Checkstyle errors in your assignment. Each error found by Checkstyle is worth one point. This cap will be raised next assignment. Again, the full style guide for this course that you must adhere to can be found by clicking **here**.

Come to us in office hours or post on Piazza if you have specific questions about what Checkstyle is looking for and how to fix Checkstyle errors.

First, make sure you download the `checkstyle-6.0-all.jar` and `cs1331-checkstyle.xml` from the T-Square assignment page. Then, make sure you put *both* of those files in the same directory (folder) as the `.java` files you want to run Checkstyle on. Finally, to run Checkstyle, type the first line into your terminal while in the directory of your Java files and press enter.

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | wc -l
      2
```

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting
    audit..." | findstr /v "Audit done" | find /c /v "hashcode()"
0
```

5 Turn-in Procedure

Submit your `LemonadeTycoon.java` file on T-Square as an attachment. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

6 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email

almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   (a) It helps insure that you turn in the correct files.

   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is 8PM Thursday. Do not wait until the last minute!