CSCI 1583 Homework 4: SimBurger

Summary

We have seen in the previous homework assignments how we can write Java programs to work with numbers. For this assignment, we will see how we can use the same numerical processing operations we have been learning to simulate life in the restaurant business. The main goal is a create a virtual restaurant and maintain it as long as possible. As the player makes orders for the customers, the restaurant should become more difficult to maintain until the player fails. Making a virtual restaurant may seem like a big task, but we can simplify this by applying the iterative approach we have discussed throughout the semester. We can break it down into small, simple pieces.

Learning Objectives

- Storage Operations using variables
- Output Operations using print statements
- Input Operations using the Scanner Object
- Control Flow Operations using Selection Statements and Repetition Statements
- Iterative design using the Top-Down Step-Wise approach
- Encapsulating tasks into methods for a D.R.Y. code

At the start of the game, the player is given a restaurant to manage. Each turn, the player is presented with the vital information informing him or her how clean the kitchen is, how happy the customers are, and how much food is left. Next, a menu list of options is displayed to the player. The list includes all tasks that the player can do. The player will choose an option, and the task should be completed. As the player serves customers, the difficulty should slowly increase until the restaurant is unmanageable, and the player finally loses.

Let us break it down from the Top:
Create a game where the player is given a restaurant to run for as long as possible.

Refinement 1:

```
Declare and or initialize variables to represent restaurant attributes
Quickly introduce game to the player

while the player hasn't lost the game
        display the state of the game to the user ( how are the customers, kitchen, food stock)
        let player choose how to go about running the restaurant
        let the player know how the customer served is feeling
        check to see if the player has run the restaurant into the ground (lost the game)
Let the player know how they did
```

Everything indented in the pseudo-code is our core game loop. These are the elements that need to happen every turn until the game is over. The player should decide what to do, and there should be some consequence on the game based on their choice.

Looking at the pseudo-code, each line is still really broad. Let us further refine this problem and make some functionality decisions.

Refinement 2:

```
Initialize global int variables that will mark in-game thresholds
Initialize global int variables that will track how the player is doing
Initialize global int variables that contain the rates that decay the players success (such as how dirty the kitchen gets after every customer)
Initialize global boolean variables to signify if the has taken an order and made the order

while the player hasn't lost the game
    display the state of the game to the user ( how are the customers, kitchen, food stock)
    let player choose how to go about running the restaurant
        let the player Take an Order OR
        let the player Make an Order OR
        let the player Serve an Order OR
        let the Player Buy Food Stock OR
        let the player Clean the Kitchen OR
        let the player Waste Valuable Time OR
        let the player Quit the Game
    let the player know how the customer served is feeling
    check to see if the player has run the restaurant into the ground (lost the game)
Let the player know how they did
```

The second refinement's topics are still pretty broad, but each line in the core game loop (indented in the while loop) is a single task. It would be challenging to transform these lines into single java lines, but they are excellent candidates for methods. A method is a set of code referred to by name and can be called anywhere in the program only by utilizing its name. **A good method does one task and one task alone.** If your methods are many lines long, you can most likely break it into several methods.

The second refinement breaks the variables into different groups for a different purpose. The first group, labeled as the Threshold variables, are in-game markers. Thresholds hold the values that trigger elements in the game. A threshold variable might hold the number of points the player needs to win the game. The game can check to see if the player's points are greater than or equal to the winning threshold. If the player's points match, then they win the game. The second group of variables tracks how the player is doing. Using the above example, the tracking variable would be the player's points (the variable is checked against the threshold value). The third group of variables concerns the rate of change that occurs to the second group. A rate of change variable would be the amount added to the player's points when they make a goal. Unlike the points example, this assignment will require the rate of change variables to increase the game's difficulty instead of adding points (though you can add additional content to add points). Rates of change can be altered throughout the game (and will need to for this assignment). The last group of variables works as flags, signaling that the player has done something in the game. An example is setting a hasOrder variable to true when the player takes an order.

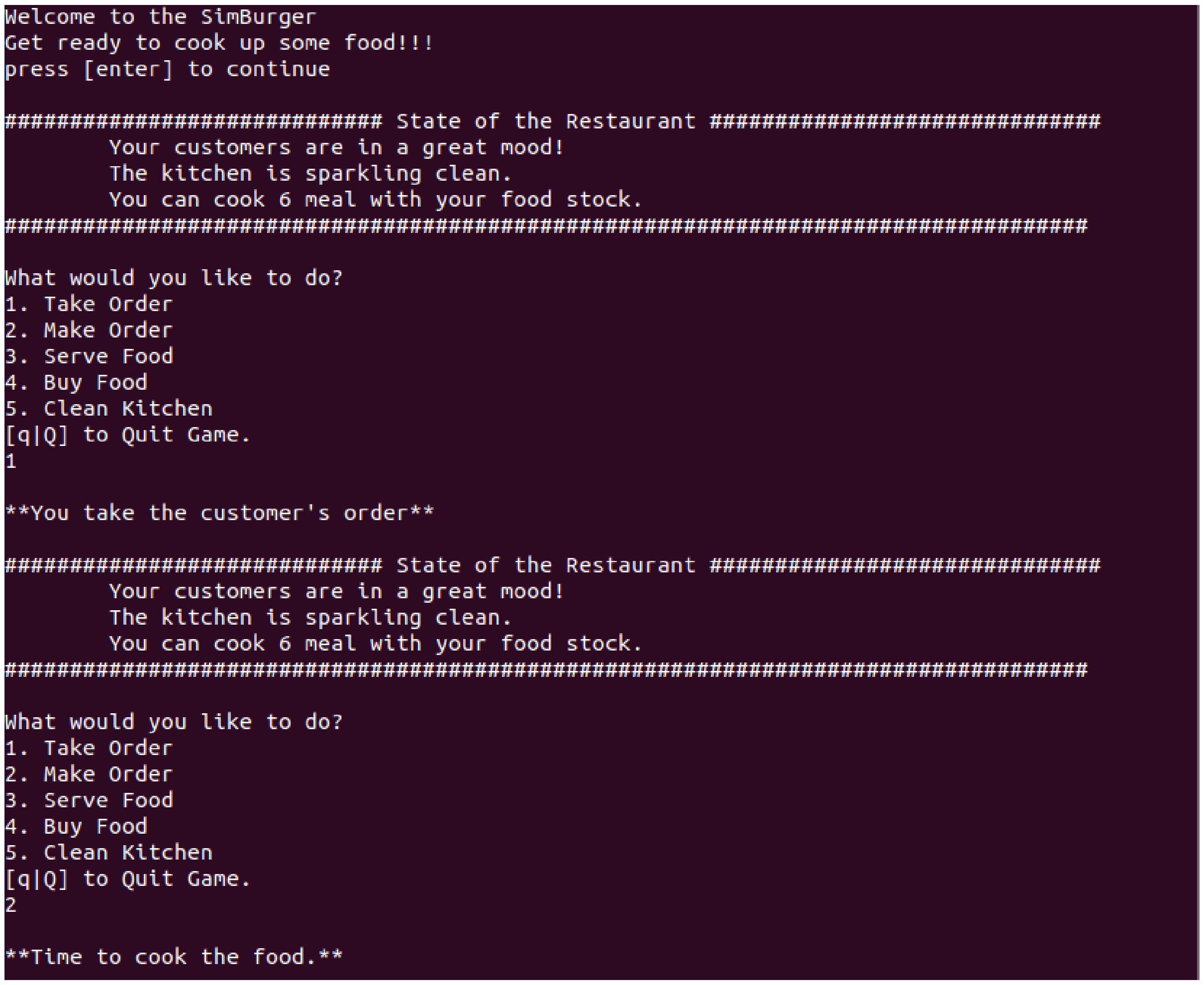
We have discussed the variables and listed the required tasks needed for the game, but the pseudo-code is still very broad and needs to be broken down further. The assignment comes with a sample file, *SimBurger.java*, that can be used for completion. It contains the next refinement along with some starter code.

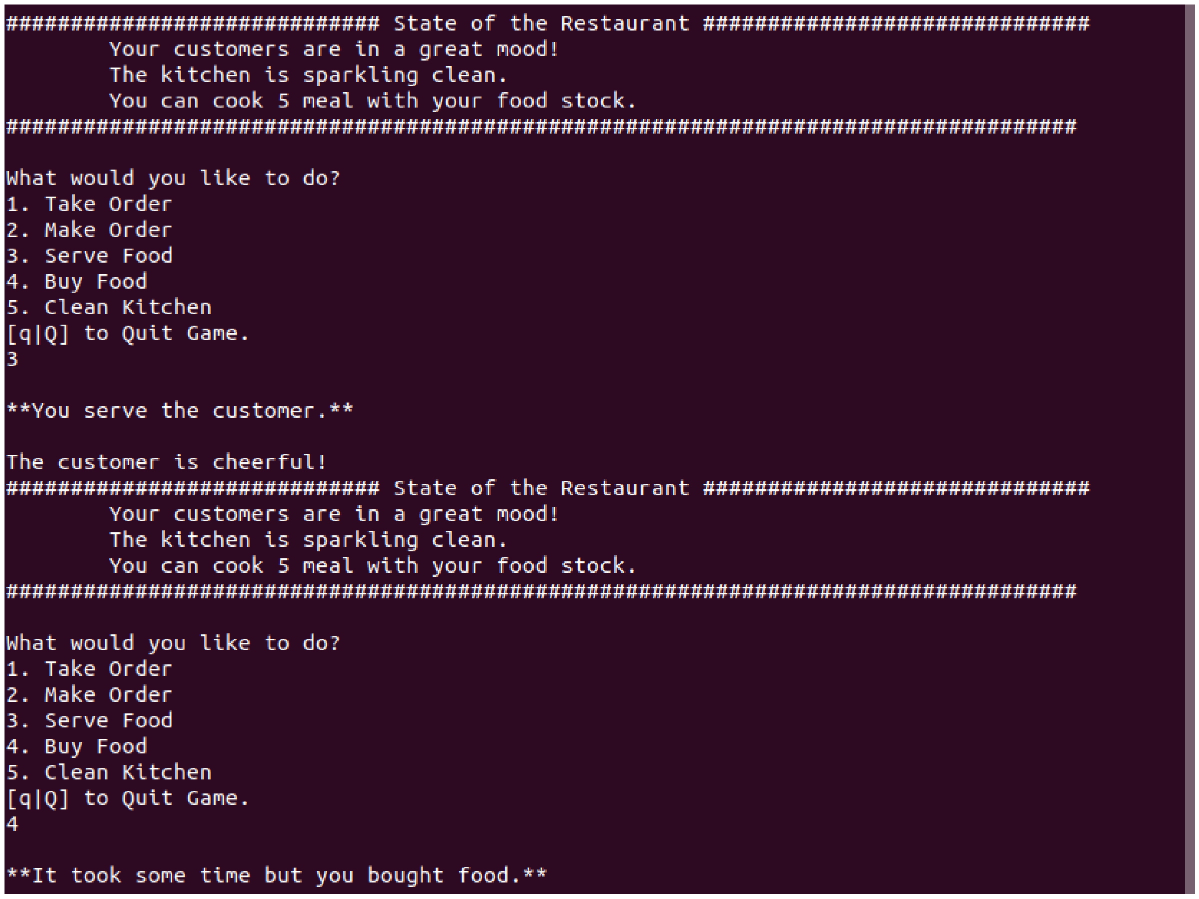
Using the Starter Code

Very rarely in industry will a developer start a project from scratch. There will almost always be a codebase that the developer will work from. The sample code provided can be thought of as the previous codebase to worked from. It is not required that you use any of the code provided (you can start from scratch if you like), but it is okay to use any or all of it.

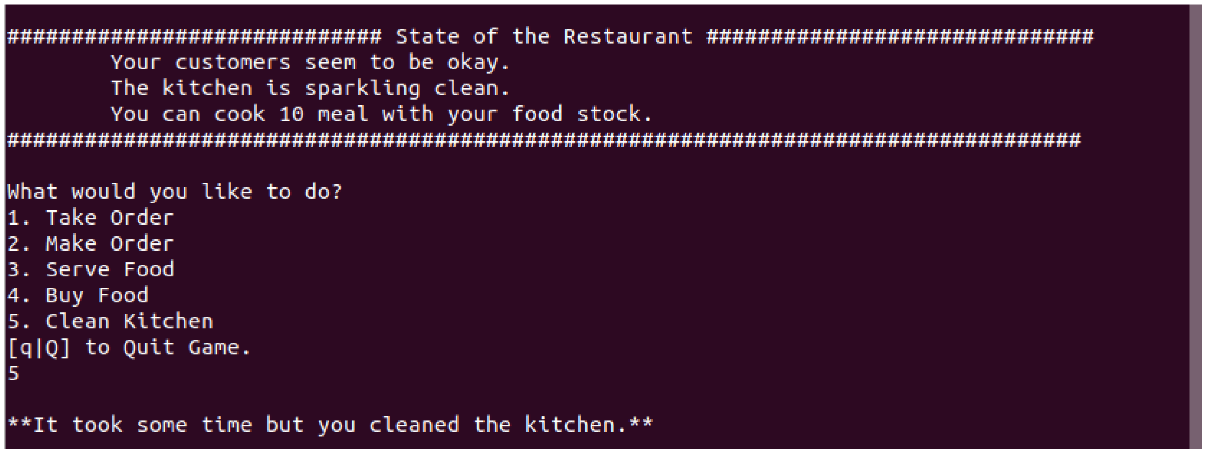
Game Flow
How the base game is broken down, the player must first take the customers' order, then cook the food, and finally serve the customer. The player cannot cook the food without taking the order and cannot serve the food without first cooking it. As the player cooks the food, the food stock depletes, and the kitchen gets dirty. Each action that the player does (besides serving the food) increases the customers' annoyance level because they have to wait. Buying food and cleaning the kitchen also takes time, which increases the customer's annoyance as well. The default game lose condition checks the customer's annoyance level to the service quality threshold and the kitchen's cleanliness to the kitchen cleanliness threshold. If either the cleanliness or annoyance level reaches the game over threshold value, the game ends, and the player is informed on how they did.

Let's look at an example output from the game.


```
Welcome to the SimBurger
Get ready to cook up some food!!!
press [enter] to continue

############################# State of the Restaurant ##############################
        Your customers are in a great mood!
        The kitchen is sparkling clean.
        You can cook 6 meal with your food stock.
###################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
1

**You take the customer's order**

############################# State of the Restaurant ##############################
        Your customers are in a great mood!
        The kitchen is sparkling clean.
        You can cook 6 meal with your food stock.
###################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
2

**Time to cook the food.**
```

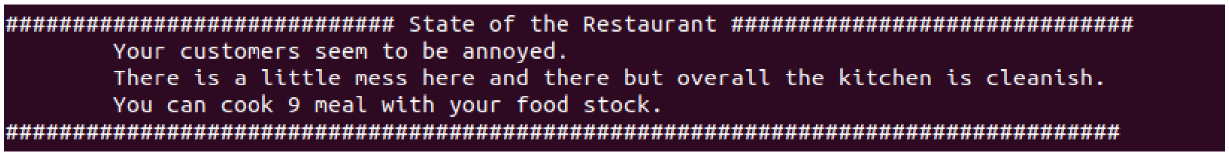


```
############################# State of the Restaurant ##############################
        Your customers are in a great mood!
        The kitchen is sparkling clean.
        You can cook 5 meal with your food stock.
####################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
3

**You serve the customer.**

The customer is cheerful!
############################# State of the Restaurant ##############################
        Your customers are in a great mood!
        The kitchen is sparkling clean.
        You can cook 5 meal with your food stock.
####################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
4

**It took some time but you bought food.**
```



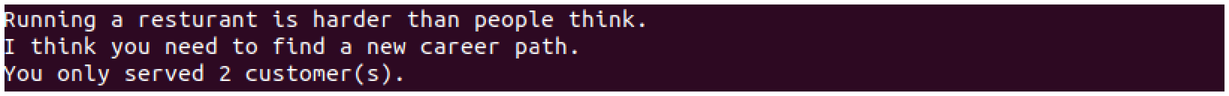
```
############################# State of the Restaurant ##############################
        Your customers seem to be okay.
        The kitchen is sparkling clean.
        You can cook 10 meal with your food stock.
####################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
5

**It took some time but you cleaned the kitchen.**
```

As the customer has to wait for their food, they become more and more annoyed. Also, as you cook, the food stock decreases, and when you buy food, the stock increases. As you cook, the kitchen gets dirty.



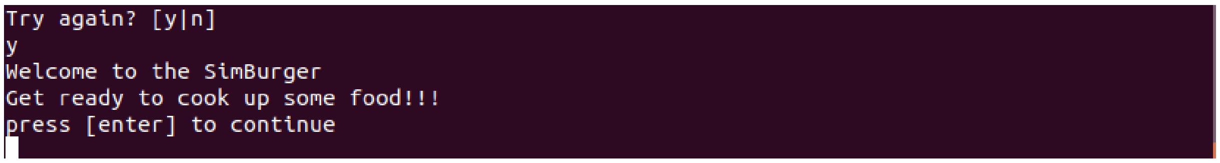
```
############################# State of the Restaurant ##############################
        Your customers seem to be annoyed.
        There is a little mess here and there but overall the kitchen is cleanish.
        You can cook 9 meal with your food stock.
####################################################################################
```

Upon a game over let the player know how they did.



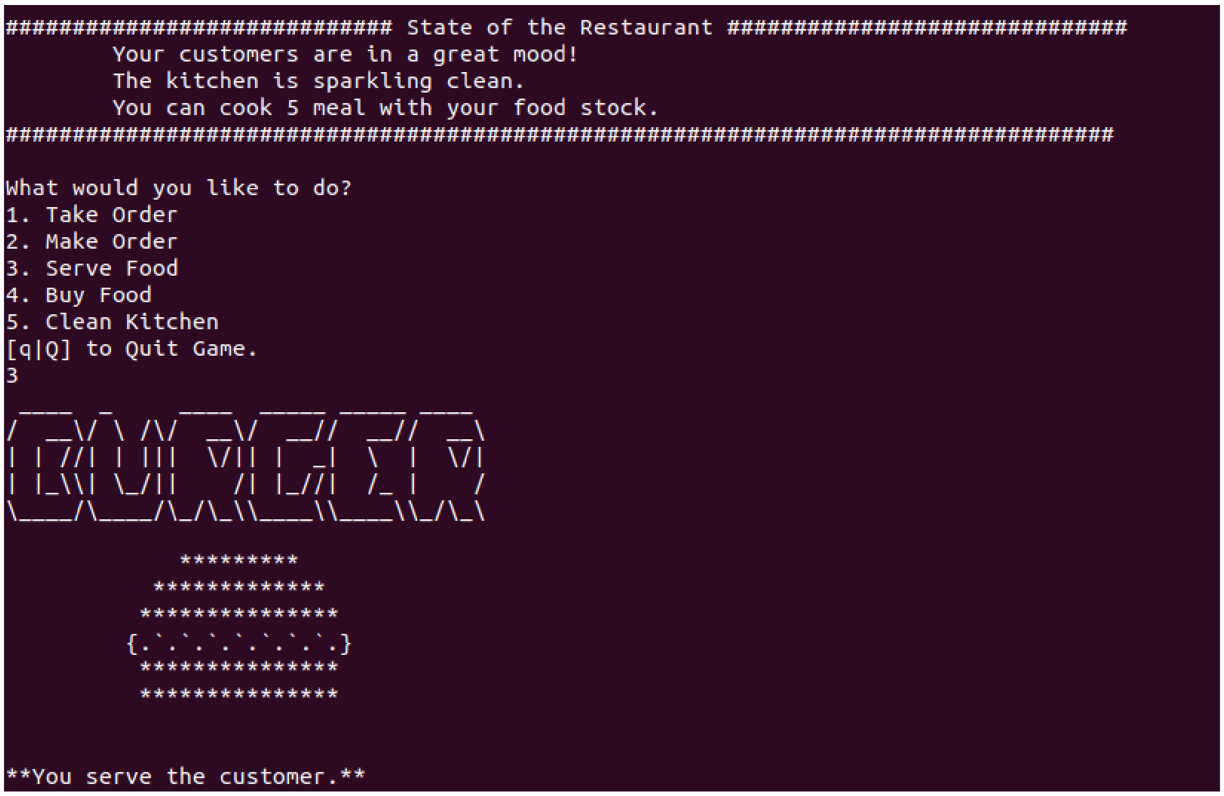
```
Running a resturant is harder than people think.
I think you need to find a new career path.
You only served 2 customer(s).
```

Bonus 1: The serial Restaurant owner 5 points
Allow the player to choose to play again after each game over. This will restart the game with a fresh start as if they are starting the game for the first time.



```
Try again? [y|n]
y
Welcome to the SimBurger
Get ready to cook up some food!!!
press [enter] to continue
```

Bonus 2: Sweet Graphics 10 points
Incorporate ASCII graphics to your game. This can be included at any stage of the game.



```
############################# State of the Restaurant ##############################
        Your customers are in a great mood!
        The kitchen is sparkling clean.
        You can cook 5 meal with your food stock.
###################################################################################

What would you like to do?
1. Take Order
2. Make Order
3. Serve Food
4. Buy Food
5. Clean Kitchen
[q|Q] to Quit Game.
3
 ____  _     ____  _____ _____ ____
/  __\/ \ /\/  __\/  __//  __//  __\
| | //| | |||  \/|| |  _|  \  |  \/|
| |_\\| \_/||    /| |_//|  /_ |    /
\____/\____/\_/\_\\____\\____\\_/\_\

             *********
           *************
          ***************
         {.`.`.`.`.`.`.`.}
          ***************
          ***************


**You serve the customer.**
```

Bonus 3: The World is your oyster!   (? points)
Take this game and make it yours. Add new features, more information to the user, or any additions that you like. If you don't like the restaurant idea, change it. If you rather open a store, do it. If you prefer to open an accounting firm, you have the power. I encourage everyone to experiment and be creative. This homework is just a sample and open for augmentation. With that said, whatever you do needs to be comparable to the sample. It must be in the same format and be equal or greater in complexity than the sample to gain all points. I will accept almost anything except for a Tamagotchi pet. **DO NOT SUBMIT A TAMAGOTCHI PET PROGRAM!!** If you have an idea and you are not sure if it is applicable, feel free to run it by me.