Assignment #2
CSCI 201 Spring 2020
5.0% of Course Grade

Title
SalEats

Topics Covered
Locks Programming
Semaphore Programming
Multi-Threaded Programming Design

Introduction
Introducing the groundbreaking, definitely original, non-USC backed nor affiliated startup, SalEats! SalEats aims to hire drivers to complete food delivery trips between USC and various restaurants in the proximity. As the Executive Officer of SalEats, I am here to inform you that the Council of SAL has decided to commission all Spring 2020 CSCI 201 students to design a cutting-edge logistics system that supports the scheduling of drivers and food deliveries.

Assignment
In this assignment, you will first read in a JSON file containing various information regarding the restaurants and their menus. The JSON format is generally the same as the one we used in the previous assignment, so feel free to reuse the parser you've built. However, you will need to alter the classes as we have modified the JSON object. Here's a sample JSON:

```
{
  "data": [
    {
      "name": "Boba Guys",
      "address": "1670 Beverly Blvd",
      "latitude": 34.063950,
      "longitude": -118.265360,
      "drivers": 2,
      "menu": [
          "strawberry matcha milk tea",
          "black sesame latte",
          "thai tea"
        ]
    },
    {
      "name": "Twinkle Brown Sugar",
      "address": "131 S Central Ave",
      "latitude": 34.048160,
      "longitude": -118.239240,
      "drivers": 3,
      "menu": [
          "brown sugar milk tea",
          "mango tango green tea",
          "rose lychee green tea",
          "brown sugar fluffy coffee",
```

```
          "sea salt iced coffee",
          "strawberry green milk tea"
        ]
    },
    {
      "name": "One Zo",
      "address": "500 N Atlantic Blvd",
      "latitude": 34.068280,
      "longitude": -118.133680,
      "drivers": 1,
      "menu": [
          "one zo boba milk tea",
          "caramel milk tea",
          "honey green tea",
          "one zo fruit tea"
        ]
    }
  ]
}
```

Aside from the JSON file, you will need to read in a second file. Below is a layout of the second text file, which contains information for the orders:

```
0, Boba Guys, strawberry matcha milk tea
0, Twinkle Brown Sugar, sea salt iced coffee
1, One Zo, caramel milk tea
5, Boba Guys, black sesame latte
18, One Zo, one zo boba milk tea
18, Twinkle Brown Sugar, strawberry green milk tea
25, Twinkle Brown Sugar, rose lychee green tea
```

- The first parameter indicates when the order is ready.
- The second parameter indicates the location that the order is coming from.
- The third parameter indicates which food is being delivered.

It takes each driver exactly one second to travel one mile, which is the equivalent of a *USAF X-51A Waverider* jet traveling at Mach 6. You can use the distance formula from `Assignment 1` to calculate how many seconds it will take for a driver to deliver food, and how many seconds to return to the restaurant. You can assume that each driver will only travel back and forth between one restaurant, and the order is not completed until the driver returns to the restaurant. The number of drivers allocated to each restaurant will be in the `assignment2.txt` file. Note that each driver will need to operate in a separate thread so other drivers can execute while one is currently out on a delivery. You will need to have a timer, which can be implemented in multiple ways, but one way is using `Thread.sleep(1000)`. The 1000 represents 1000 milliseconds, or 1 second, which is what you would work for this assignment. To make sure a driver can't be assigned more than one order, you will need to use a lock.

Just like the previous assignment, prompt the user to enter a filename and check for the file's existence and validity when the program first runs. You will also prompt the user for the latitude and longitude of their current location. For testing purposes, you may set latitude to `34.021160` and longitude to `-118.287132` (same values as `Assignment 1`). You will need to utilize locks and conditions to settle the availability of the drivers. For example, one driver cannot take two orders at the same time. The first trip must be completed before starting the second trip. The program should output when an order is picked up and when it is completed.

Example output with timestamps bolded for clarity (you do not have to bold the timestamps for your execution of the program). Your output may not match exactly because of the use of threads, but the format needs to be exactly the same for you to receive full credit on the assignment.

```
What is the name of the file containing the restaurant information?
assignment2.txt

What is the name of the file containing the schedule information?
schedule.txt

What is the latitude? 34.021160

What is the longitude? -118.287132

Starting execution of program...
[18:06:35.00] Starting delivery of strawberry matcha milk tea from
Boba Guys!
[18:06:35.00] Starting delivery of sea salt iced coffee from Twinkle
Brown Sugar!
[18:06:36.00] Starting delivery of caramel milk tea from One Zo!
[18:06:40.00] Starting delivery of black sesame latte from Boba Guys!
[18:06:41.40] Finished delivery of strawberry matcha milk tea from
Boba Guys!
[18:06:41.60] Finished delivery of sea salt iced coffee from Twinkle
Brown Sugar!
[18:06:46.40] Finished delivery of black sesame latte from Boba Guys!
[18:06:53.00] Starting delivery of strawberry green milk tea from
Twinkle Brown Sugar!
[18:06:54.80] Finished delivery of caramel milk tea from One Zo!
[18:06:54.80] Starting delivery of one zo boba milk tea from One Zo!
[18:06:59.60] Finished delivery of strawberry green milk tea from
Twinkle Brown Sugar!
[18:07:00.00] Starting delivery of rose lychee green tea from Twinkle
Brown Sugar!
[18:07:06.60] Finished delivery of rose lychee green tea from Twinkle
Brown Sugar!
[18:07:13.60] Finished delivery of one zo boba milk tea from One Zo!
All orders complete!
```

Grading Criteria

You will be graded based on the correctness of scheduling as well as the order and duration of deliveries.

File I/O (1.0%)

0.25% - JSON File I/O
0.25% - Text File I/O
0.5% - Checking for invalid user inputs

SalEats Program Execution (4.0%)

0.5% - Implementing scheduling of drivers
1.0% - Order start/completion print statements
2.5% - Locks implementation