Title

Diner Data

Topics Covered

Locks Programming
Semaphore Programming
Multi-Threaded Programming Design
JDBC

Introduction

A diner owner has asked you for help improve the efficiency of their business. When the lunch rush comes in, he would like to see how long customers wait for their orders to improve customer satisfaction, as well as what items take the longest so he can determine the bottlenecks in the production. Additionally, he would like to know the remaining quantity of each item so he can be more efficient when ordering ingredients or getting rid of less popular items.

Assignment

In this final assignment, you will first read in a JSON file containing the menu and the orders for the customers. The menu is separated into the different appliances that are used in the diner. These appliances will be the same for grading as they are for the sample JSON file provided. The menu items are subject to change though. They are broken down in this fashion because each appliance can handle a different number of tasks and take different times to complete each task. Below, you will find the layout of the JSON file.

```
{
     "Menu" : {
          "deep fryer" : [
               "french fries",
               "funnel cake",
               "corn dogs"
          ],
          "grill" : [
               "hamburger",
               "veggie burger"
          ],
          "milkshake maker" : [
               "chocolate shake",
               "strawberry shake",
               "vanilla shake"
          ],
          "drink machine" : [
               "small drink",
               "medium drink",
               "large drink",
               "coffee"
          ]
     },
     "Customers" : [
          {
               "order" : [
                    "hamburger",
                    "french fries",
                    "vanilla shake",
                    "large drink"
               ]
          },
          {
               "order" : [
                    "hamburger",
                    "veggie burger",
                    "corn dog",
                    "french fries",
                    "chocolate shake",
                    "large drink"
               ]
          }
     ]
}
```

When the program first runs, prompt the user to enter a filename. Be sure to handle whether the file exists or not, but it is assured that the testing file is a valid format. When the file is read, the customer orders must first go through the register. Use a lock so that the register only takes one order at a time. Each order should take 1.5 seconds at the register.

Once an order is completed at the register, it can start to be made in the kitchen. Save the current time when an order is completed at the register and when it is completed in the kitchen to determine how long it took to complete the order. Below is the information for each appliance including how many items it can handle at a time and how long it takes to complete each item.

Deep Fryer – 4 items, 2 seconds
Grill – 5 items, 5 seconds
Milkshake Maker – 2 items, 3 seconds
Drink Machine – 2 items, 2 seconds

During execution, print to the console when each order was made, when each order is completed, and when all orders are completed. Below is a sample of the execution. Note that this does **not** correspond to the sample JSON file, so make sure you validate your output based on the JSON file you are using.

```
18:35:02.33 Starting order 1!
18:35:03.83 Starting order 2!
18:35:05.33 Starting order 3!
18:35:05.53 Completed order 1!
18:35:06.83 Starting order 4!
18:35:07.13 Completed order 3!
18:35:08.03 Completed order 2!
18:35:08.33 Completed order 4!
All orders complete!
```

Database
You will need to create a database to hold information about the orders. So that the graders will be able to grade your program, create a file named `createDatabase.sql` in your project directory that will create the database and tables. Each execution of the above program will be stored in the database. Store each order with the amounts of each item that were ordered, and the time each order was started and completed. These tables should be updated each time the above program is executed.

Create another program named `RestaurantStatistics` that will print out the statistics of the orders by reading from the database. From the execution in the previous section, you should print out the following:

```
Execution started at 18:35:02.33.
Order 1 took 00:00:03.20 to complete.
Order 2 took 00:00:04.20 to complete.
Order 3 took 00:00:01.80 to complete.
Order 4 took 00:00:01.50 to complete.
Execution took 00:00:06.00 to complete.
Execution ended at 18:35:08.33.
```

Grading Criteria (4.0%)

Diner Program Execution (2.0%)
0.25% - JSON File I/O
0.25% - Order start/completion print statements
1.5% - Semaphores and locks implementation

Databases (2.0%)
1.0% - Database created properly with a `createDatabase.sql` file provided
1.0% - `RestaurantStatistics` program outputs results correctly