Phase 2 Document

TEAM\_I4

**Output File**

The output file is targeted to show the final data of the restaurant and its regions in an external text file which is easier to be read by the user. Objects of this class maintain a filebuf object as their internal stream buffer, which performs input/output operations on the file they are associated with (if any).

Firstly, we ask the user to input the file name in order to avoid the hard-coding problem then open this file and deal with it like how we deal with the console concerning the operators (<< for cout) & (>> for cin) but as we’re saving here so we’ll use this only (FileObjectName << \*data\*).

We added members in the restaurant and setters for them, they were essential because logically each restaurant should have information about the delivered orders, their arrival and finish times, the number of motors, the average waiting time… etc..... We also added some of these members in the region, not only for the logical reason but for easing the process of calling and making calculations too.

We chose to call the setters of these data inside the function because we call the Save function after all the events are executed and this is the part where we want to gather the data of the whole restaurant and its regions and that is simply the purpose of the save function.

**Simulation Modes**

The simulation works in 3 modes as described in the project document, here’s how they differ from each other; the interactive mode is the one with the most details, that’s because it takes a click from the user (using the waitForClick function implemented in the GUI) and also it prints the status of the current event happening in the restaurant. We excluded the printing part as a separate function and only call it in the simulation function to avoid long functions.

The step mode is similar to the interactive mode regarding the printing part but the difference is that it doesn’t take the input click from the user to proceed, this was done by removing the GUI function we mentioned previously and replaced it with function sleep(1000), it’s used to pause the system 1 second before proceeding to the next time step.

Lastly, the silent mode doesn’t require printing or displaying any of the data, it just outputs the data file so we removed all the functions related to the GUI.

**Calculation for Average Waiting/Serving Time:**

Firstly, this is a very important point to be taken into consideration to update the performance and efficiency of the restaurant. It comes by knowing the satisfaction of the customer which can be measured by: the time delay of the order, and the time they wait for it to be served. We chose to calculate both Average Waiting and Serving time in one function to have reasonable complexity for the code. Also, we've provided a setter for these times along with setters and getters for the data members that are being modified through the call of the function. This function's purpose is to calculate the number of delivered orders in all regions. Then, it adds the Waiting and Serving time for each type: VIP, Normal, and Frozen. It then edits the data member responsible for each time and updates it with the new one. Then it takes the total number of Waiting/Serving time divided by the total number of orders delivered. If the restaurant doesn't take any order, it will have a zero value. This function and its helpers (setters and getters) are implemented as member functions of class Region.

**Promotion:**

Promotion is implemented as a class that inherits from class Event. It has data members that are equivalent to the data that's read from the input file. It has a virtual Execute function that gets the order by using a function called getOrder in class Restaurant, and checks if it's found or not. It then updates the order's information, and changes its type from Normal to VIP, and adds the Extra Money from the input file to the actual money of the order. It then calls a function that's found in Restaurant called Promoted that's responsible for finding the region of the order, and calling another function in Region called Promotion. The Promotion function in region is responsible for deleting the order from the Normal Orders List as it has direct access to it. It also calculates the priority of the Order, and enqueue it in the VIP sorted list.

Auto-Promotion is another side for the promotion of an order, it basically comes directly without the interference of the customer when the waiting time and arrival's time summation exceed the promotion limit. This function is implemented as a member function is class Restaurant. It takes a time step as it's called every time step to check for the condition to be true, and loops on the four regions to find the order that should be auto-promoted, and calls the Promotion function in Region.

**Serving of Orders:**

In each time step, we call a function named ServeAllRegions that calls function ServeAll for the 4 regions. This function basically calls three other functions each one responsible for serving one type of orders in the correct sequence (VIP then Frozen then Normal)   
In each function we loop on the motorcycles list to find an Idle motorcycle. In our function, when this motorcycle is found, we change its status to be serving.  
A more efficient method of searching for an idle motorcycle is to separate lists for each motorcycle statuses. i.e a separate list for serving motorcycle and when it is assigned to an order we remove it from the idle list and insert it in the serving list. This way we can avoid extra loops thus having better complexity and efficiency

.  
We thought of that after completely finishing the entire project so we thought it was too late to edit any more knowing this the previous one  is a better way.  
  
After assigning the order to a motorcycle we remove the order from its original lost and insert it(in a sorted way in terms of the finish time) in the delivered list of orders in its region.  
  
The finish time, waiting time and serving time are calculated in the function that searches for the motorcycle such that  the serving time equals Ord->get dist() ÷ the speed of the chosen motorcycle. The waiting time is calculated by subtracting the arrival time of the order from the timestep which the order is served in.

**Updating Motorcycles’ Status:**

A function named Motorcycles is called before beginning to serve in each time step.  
This function loops on all the motorcycles to check whether the serving motorcycles have returned an became idle to be available to serve another order or not. A data member in the motorcycle (resting time) is used to know when it is time. This data member is set when the order is first assigned to the motorcycle with value equal to the order's finishing time plus its serving time (the time the motorcycle takes to return back to the restaurant after serving)  
  
We have also created another two status for the bonus part that is resting and damaged status . Each motorcycle has a data member that is: health. For motorcycles' safety , each motorcycle should rest after each serving of order it does , this means that if a motorcycle didn’t take its usual resting time it makes it more prone to damage , damaging of a motorcycle means that it will need a great deal of time resting without doing any services for its maintenance and for its health to recharge . We created a function that updates the status of the motorcycle every time a change in status occurs that updates the status when servicing from idle to in service that when a motorcycle returns it changes status from in service to resting to idle. When the motorcycle is damaged, its status is changed to damaged. This has been all testes in a document alone "RD7" to test the damage, waiting and resting times.

Edited by:

Amira Ahmed

Hana Amr Mahmoud

Kariman Mahmoud

Youmns Khale Sayed