

Cairo University Computer Engineering Department

Faculty of Engineering First Year



**Data structures & Algorithms**

**Team 7**

|  |  |  |
| --- | --- | --- |
| Name | Section | B.N. |
| Ghieath Omar Saleh Ajam | 2 | 5 |
| Mostafa Ahmed Sobhy Ahmed | 2 | 22 |
| Mostafa Mohamed Ahmed El-Gendy | 2 | 25 |
| Youssef Atef Tawfik Hafez | 2 | 39 |

**June, 2020**

**Function Name:** RemoveFromBreakList

**Member of:** class Restaurant

**Inputs:** currentTimeStep

**Returns:** void

**Called by:**

**Calls:**

* BinarySearchTree::Delete(T item, K key)
* ManageBreak::EndBreak(Restaurant\* pRest)

**Function logic description:**

It checks all the cooks’ break lists (Binary Search Tree). If there is a cook who finished his break, then it deletes it from the list and instantiates a (Manage Break) object to end his break.

**Function Name:** PrintFile

**Member of:** class Restaurant

**Inputs:** void

**Returns:** void

**Called by:** Restaurant::RunSimulation()

**Calls:**

* GUI::PrintMessage(string msg)
* GUI::GetString()
* LinkedList::Sort()

**Function logic description:**

It prints the output file in the specified format.

**Function Name:** RunSimulation

**Member of:** class Restaurant

**Inputs:** void

**Returns:** void

**Called by:** Main function

**Calls:**

**-** GUI::getGUIMode()

- Restaurant::ReadFile()

- Restaurant::Run(mode)

- Restaurant::PrintFile()

**Function logic description:**

Organizes overall flow of the program, i.e. Creates a GUI, gets mode and passes it, handles input, starts simulation and prints final output.

**Function Name:** Run

**Member of:** class Restaurant

**Inputs:** PROG\_MODE (enum)

**Returns:** void

**Called by:** Restaurant::RunSimulation()

**Calls:**

- isEmpty() (for orders and events queues)

- Restaurant::ExecuteEvents(int ts)

- Restaurant::RemoveFromBreakList(int ts)

- Restaurant::RemoveFromRestList(int ts)

- Restaurant::CompleteOrders(int ts)

- Restaurant::ManageOrders(int ts)

- Restaurant::AssignOrders(int ts)

- Restaurant::CheckInjuries(int ts)

- Restaurant::FillDrawingList(int ts)

- GUI::waitForClick()

- std::Sleep(int ms)

- GUI::PrintMessage(str msg)

**Function logic description:**

Handles flow of the program by executing events, handling breaks, rests and injuries of cooks, handles orders then passes instructions to GUI depending on program mode.

**Function Name:** ExecuteEvents

**Member of:** Restaurant

**Inputs:** int timestep (ts)

**Returns:** void

**Called by:** Restaurant::Run(mode)

**Calls:**

- peekFront of Queue

- Event::getEventTime()

- Event::Execute(pRest)

- dequeue of Queue

**Function logic description:**

Loops on events checking if upcoming event’s time has come, if so, executes it then remove it from the events queue and deletes it.

**Function Name:** GetNormalOrderFromID

**Member of:** Restaurant

**Inputs:** int ID

**Returns:** NormalOrder\*& no

**Called by:**

**-** Restaurant::CancelOrder

- Restaurant::PromoteOrder

- PromotionEvent::Execute

**Calls:**

- LinkedList<NormalOrder\*, int>::GetEntry

**Function logic description:**

Searches for a normal order by its ID then returns a pointer to it if found.

**Function Name:** CancelOrder

**Member of:** Restaurant

**Inputs:** int ID

**Returns:** void

**Called by:**

- CancellationEvent::Execute

**Calls:**

- LinkedList<NormalOrder \*, int>::Delete

**Function logic description:**

Cancels an order of given ID.

**Function Name:** PromoteOrder

**Member of:** Restaurant

**Inputs:** int ID, int time

**Returns:** void

**Called by:**

- Restaurant::ManageOrders

- PromotionEvent::Execute

**Calls:**

- Restaurant::GetNormalOrderFromID

- LinkedList<NormalOrder \*, int>::Delete-

- LinkedSortedList<VipOrder \*, int>::Insert

**Function logic description:**

Gets a pointer to the normal order of given ID then transforms it into a VIP order if found.

**Function Name:** FillDrawingList

**Member of:** Restaurant

**Inputs:** int current TimeStep

**Returns:** void

**Called by:**

- Restaurant::Run

**Calls:**

- GUI::ResetDrawingList

- Order::GetType

- OrderService::GetOrder

- OrderService::GetCook

- GUI::AddToDrawingList

- OrderService::GetStartTime

- Cook::GetType

- Cook::GetID

- Order::GetID

- GUI::PrintMessage

- GUI::UpdateInterface

**Function logic description:**

Resets drawing list then fills it again from scratch by looping through every list of orders/cooks while counting numbers of each type of orders and their service status.

**Function Name:** AddToNormalList

**Member of:** Restaurant

**Inputs:** NormalOrder\* po

**Returns:** void

**Called by:**

- ArrivalEvent::Execute

**Calls:**

- LinkedList<NormalOrder \*, int>::InsertEnd

**Function logic description:**

Takes a pointer to a normal order and adds it to the end of the list of normal orders then increments number of normal orders by 1.

**Same for:** AddToVeganList, AddToVipList

**Function Name:** FindCook

**Member of:** Restaurant

**Inputs:** Order\*

**Returns:** Cook\*

**Called by:**

- Restaurant::FindCookForUrgentOrder

- Restaurant::AssignOrders

**Calls:**

- Queue<Cook \*>::dequeue

- also uses Dynamic cast

**Function logic description:**

Searches through available cooks then returns a valid one depending on the order type.

**Function Name:** FindCookForUrgentOrder

**Member of:** Restaurant

**Inputs:** VipOrder\*

**Returns:** Cook\*

**Called by:**

- Restaurant::ManageOrders

**Calls:**

- Restaurant::FindCook

- dequeue of cooks lists

**Function logic description:**

First searches through available cooks by calling findcook function. If no cook available it searches in cooks break lists then cooks rest lists.

**Function Name:** AssignOrders

**Member of:** Restaurant

**Inputs:** int current TimeStep

**Returns:** void

**Called by:**

- Restaurant::Run

**Calls:**

- Order::GetAT

- Restaurant::FindCook

- OrderService::Serve

**Function logic description:**

Loops through lists of orders then searches for cooks to assign with found orders.

**Function Name:** CompleteOrders

**Member of:** Restaurant

**Inputs:** int current TimeStep

**Returns:** void

**Called by:**

- Restaurant::Run

**Calls:**

- BinarySearchTree<OrderService \*, int>::Delete

- OrderService::FinishOrder

**Function logic description:**

Checks for finished orders then calls FinishOrder function to move any finished order to fnished orders list.

**Function Name:** ManageOrders

**Member of:** Restaurant

**Inputs:** int current TimeStep

**Returns:** void

**Called by:**

- Restaurant::Run

**Calls:**

- Order::GetAT

- NormalOrder::GetAutoPromote

- Restaurant::PromoteOrder

- Order::GetID

- LinkedSortedList<VipOrders\*, int>::GetEntry

- Restaurant::FindCookForUrgentOrder

- VipOrder::Urgent

- OrderService::Serve

- LinkedSortedList<VipOrders\*, int>::Delete

**Function logic description:**

This function handles special events including promoting normal orders and assign cooks to urgent VIP orders.