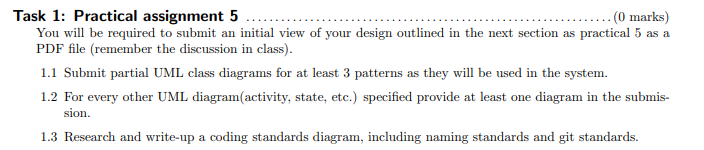
Cult of COS

TO DO :

Identify all design patterns. (minimum of 10 needed)

Decide, game or real?

Divide work.

Plan for Tuesday, 

Next meeting date.

Coding standards

GUI (RayLib)

Doxygen

Setup GitHub

PATTERNS: Cole , Ivan , Justin , Julia , Charl , Janicke

Main

Strategy for Chef types 0

( State [customer happiness / ready to order <- this is for a table ] /) 1

(Iterator [ checking table availability / searching for the reserved table ] ) 2

( Command [ (CONTROLS CUMMICATION BETWEEN MOST CLASSES for example, talks to iterator to retrieve seating location for clients {receives order from main woman], when customer pays sends to bank ens / floor, opens and closes restruant, spawns customers in background ] ) 3

MaitreD

Floor

( Chain of responsibility, for movement of order and then splitting it up so that it goes to the correct chefs ) 4

(Mediator [kitchen window talking to chefs to split up order] ) 5

(Restaurant for singleton (instantiated by main woman)) 6

(Factory method for creating customers works through main woman, calls customer object) 7

(Observer for state of customers / available seating [we are using push observers that will call on waiters, if we run into thread issues use waiters to constantly check) 8

(Decorator for table and menu items) 9

(Composite for Menu items/ section hold tables, tables holds clients) 10

REMOVED : (Template Method Opening and closing restaurant) 11

(Adapter if we cant find more)

MEETING 2:

Justin brings laptop to demo at orange lab 13:00.

Justin : General code starting (creating classes and header files)

Cole + Charl + Janicke + Julia : 2.1

Cole + Charl : 2.3 – 2.5

Janicke + Julia : 2.6 – 2.8

Ivan : Start GUI

Justin + Ivan : Document code using doxygen.

MEETING FRIDAY AFTERNOON 14:00.

EXPLINATION OF ALL DIAGRAMS:  
Restaurant : Holds the floor, controls whether or not the simulation is active, controls whether or not the restaurant is full (thus must be notified by MaitreD how many customers have been added by calling its incOccupancy function). Controls round counter, os advances the simulation.

OpenRestruant : contains all initialisation that goes with opening the restaurant if the restruant did not exist already, if it’s closed state must simple be changed to open, that is what it does.

CloseRestaurant : either deconstructs the restaurant or closes it depending on what the user wants.

Floor / Tile / Path : (Ask Julia and Ivan).

MaitreD : Command Pattern, acts as the ‘door’tile for the restaurant, spawns all customers and uses the iterator pattern to go through all the tables to find a table with enough seats to hold the party of customers, if customer has a reservation use a different iterator to take the customer directly to their table. A customer does not immediately go in when they are created rather they wait in line, the reservation is made unpon their creation. MaitreD also controls the opening and closing of the restaurant.

Table Iterator class: iterates through the tables appropriately depending on MaitreD command.

Table : Table holds the customers once seated, and when all customers are ready to order it summons a waiter to come collect the order. Has a certain waiter assigned to it.

Waiter (observer) : The waiters collect orders from the table class (each waiter has one or more tables which only they see to) once summoned by a table to collect the order they take this order to the kitchen window, once the kitchen window signals an order is complete the waiter takes the order to the appropriate table. Once a table is done eating the waiter will take the paid bill to the maitreD if the customers pay on a later date they will inform the MaitreD of this.

KitchenWindow (Mediator) : Uses the chain of responsibility built into Chef to split an order into its meals and its meals into its meal components, each chef then receives the correct meal component that they can prepare. Once all parts of an order are complete the kitchen window reassembles the order and notifies the waiter of its completion to be taken away.

Chef and sub chefs (Strategy and Chain of Responsibility) : Each chef prepares the meal component they are proficient in, remember to take note of the preptime required for each meal component as this is how many game cycles it will take to complete that component, once a meal component is done it sends it back to the kitchen window.

Menu and menu components and Plating (composite and decorator): a menu is made up of MenuItems and menuItems are made out of MealComponents. Each Menu Item should also have an option to be plated in a different manner, this will influence customer happiness positively but also add cost. What plating option a customer chooses can be made random.

Order : Collection class for all the orders that come from a table, makes transport between classes easier and allows chefs to have an id to what order a meal component belongs to.

Customer (State and Factory) : Controls the happiness and order state of the customer, happiness is from 0 – 10, a customer is considered happy between 6-10 and unhappy from 1-5, if happiness hits 0 all customers on that table must leave the restaurant, thus losing the restaurant the money from their order. The type of customer can be randomly determined by the MaitreD, a Wealthy customer always has a reservation, KarenCustomers lose happiness faster than a normal customer. Talks to table every time they choose what they want to order (increasing the ordercount when order count == customer count at that table big order is made and given to waiter), a customer must also hold a menu.