CSCI 201L Final - Programming
Fall 2014
13% of course grade

You are going to be creating a Cook employee. Waiters will now take an order from Customers. Waiters will give taken orders to the Cook for cooking. When the food is ready, Cooks will let Waiters know the food is ready. Waiters do not wait for food to be cooked, they can handle additional Customers while orders are being cooked. Waiters then "deliver" the completed food to the Customer. Customers cannot leave before they get their food. Once a Customer gets their food, they wait the random amount of time (that is already in their code), then they leave the restaurant. What Waiters currently do after a Customer leaves does not change.

It is important that you follow the steps given below to complete this exam. Each step is worth points and they are given to you in an order that will make finishing the exam easier for you.

All data sharing is to be thread safe. All thread interactions are to be completely free from race conditions.

1. Create a CookPanel class **(1%)**
    a. Create a new class called CookPanel. This class is to extend JPanel. Place this new panel to the right of all existing GUI components in the Restaurant JFrame.
    b. Make the CookPanel size to be 300 pixels wide by 500 pixels high.
    c. Change the size of the JFrame to be 800 pixels by 500 pixels high.
    d. Add a scrollable text area that all Cooks will write their messages to.

2. Waiters take orders from Customers. **(2%)**
    a. Customers must wait for their Waiter to arrive at their table so they can place their order. Orders are simple:
        - 0 – means a steak dinner
        - 1 – means a seafood dinner
        - 2 – means a vegetarian dinner

        A Customer generates a random number (0, 1, or 2), and passes this value to the Waiter after the Waiter asks for their order. You must pass the actual value for the order.
        Print a message, to the regular message window after the Waiter has a Customer order.
        The message is to say: "Waiter X has taken an order for Y from table Z.".

3. Use the provided Cook class that extends Thread and the Order class provided to you. **(2%)**
    a. Cooks share a Vector to store received orders that they have not started to cook. This is a Vector<Order>. The variable is called *ordersToCook*. The Order class has an integer for the actual order, the table number for the Customer who ordered it, and a reference to the Waiter that placed the order. Waiters will be adding orders to this list in step 4.
    b. Change the main GUI so that the number of Cooks to be created can be set – just like the number of Waiters, Tables, etc. Create the appropriate number of Cook threads when the restaurant is started. You are to have a JLabel and combo box, just like for selecting the number of Waiters.
    c. When a Cook thread is created, they are to print a message to the CookPanel that they are ready to cook. This message is to include their Cook number. The message should be something like "Cook X is ready to cook.".

4. Waiters give Order to the Cook. **(2%)**

    From step 2, Waiters have a Customer's order. They are to append this order to the *ordersToCook* vector that the Cook thread has. Don't forget to create a new Order object each time a Customer places an order (see step 3). The Cook will use the Waiter reference to alert the Waiter that the order is completed.

5. Cooks cook orders **(3%)**
    a. Cooks will check the vector of uncooked orders to see if there is something to be cooked. If not, they are to wait between 10 and 20 seconds, before checking the vector again.
    b. If there is an item to cook, the Cook removes the first item from the vector of uncooked orders and begins to "cook" the order. It is to take from 5 to 10 seconds (randomly determined) to cook any order. The Cook is to print a message to the CookPanel that says: Cook X is cooking an order for Y for table Z.".
    c. When an order is completed, the Cook is to call a method in the appropriate Waiter object to let the Waiter know the order has been cooked.
    d. The Cook is also to print a message to the CookPanel. The message is to say: "Cook X has completed an order for Y for table Z."

6. Waiters check for cooked orders and deliver to Customers **(3%)**
    a. Change the Waiter code so that they check, every so often, for completed orders. Since Waiters can have multiple tables, you need a Vector<Order> for each Waiter that the Cooks populate when an order is cooked. If the Vector is not empty, the Waiter is to "deliver" the food to the Customer. Since Customers are waiting for their food, the Waiter will have to signal the Customer. There is no data to pass, as the Customer knows what they have ordered.
    b. The Waiter is to print a message to the regular window: Waiter X has delivered order Y to table Z.".
    c. When Customers are given their food (they wake up), they are also to print a message: "Customer X has received order Y at table Z.". Customers do their typical random delay to simulate eating, then they leave.