***Thneed Inc: Phase 1***



Create a program that allows a business (Thneed Inc.) to keep track of customers and orders. There will be *Customer* objects that will indicate the name, address and phone number of the customer, along with a list of current and past orders. There will be *Order* objects which indicate which *Customer* is placing the order, and the details of the order, such as number of Thneeds, size(s) and color(s). The *Order* object should also indicate the date the order was placed and the date the order was filled (null if not yet filled). There will be a GUI which will allow the user to view a list of all orders, with the details of the currently selected order shown. If the user clicks on the Customer field for the current order, the details of the Customer should be shown. The GUI should allow the user to enter a new order, new customer, and update the date a particular order was filled. Finally, the GUI should allow the user to save the current state of the data (*Orders* and *Customers*) to a file. When the application launches, it should look for that file and populate the application with the information in that file if it exists.

**Classes and GUIs**

* *●  Customer:* A class that contains the relevant information for a particular customer, including a unique Customer ID #, name, address, phone number and list of orders associated with this customer.
* *●  Order:* A class that contains the relevant information for a particular order, including a unique Order #, a list of Thneeds ordered (quantity for a particular size & color... and order could have multiple different types of Thneeds), which customer ordered the item, the date it was ordered and the date it was filled.
* ●  File I/O:
  + ○  Upon starting the application, it should look for a flat file that has all of the  *Customer* and Order in formation and loads it into the appropriate data structures  in the program. If the file doesn’t exist, it should start with no data.
  + ○  Upon pressing a button in the GUI to save all data, *or closing the application*, the  application should write all *Customer* and *Order* data to a flat file.
* ●  GUI: There should be three parts to the GUI:
  + ○  Display all orders
    - show unfilled orders first ­ with oldest order first
    - then show filled orders ­ with most recently filled first
  + ○  A single order can be selected at a time. That order should have all pertinent information displayed on the same screen (don’t pop up a new window which has to be dismissed to view orders again).

■ Allow the user to update the filled date for the current order, and explicitly “save” it.

* ○  If the user clicks on the *Customer* associated with an order, the *Customer* data should be shown (this could be on the same screen, or in a new window). You do not need to allow the Customer data to be updated at this time.
* ○  There should be a button to add a new order. This form should allow users to select which *Customer* is placing the order, or create a new *Customer*.



○ There should be a button to write all existing data to a flat file.

Note: you can hardcode the location and name of the flat file which stores the data.

**Programmer Roles**

Each student should be lead programmer on one of the following: *Customer and Order* classes, GUI, File I/O. In addition, each student should take the lead in one of the following roles (but teams are expected to work together to review each other’s work and make sure no major mistakes are being made):

* ●  **Architect:** Work out how the classes and GUI will work together for the final program. What methods will each class need? What data will each class need? Etc...
* ●  **Tester:** Create unit tests for each class, and an overall test for the project. Make sure to test for boundary cases and malformed input.
* ●  **Document/Repo:** Ensure all code is documented and committed properly, including test code.