Homework 3

Homework 3 will be a **team project between everyone in the class.** You will work together as a team of three to complete the following deliverables over the next two weeks.

**Objectives**

* Develop familiarity with the concept of “static”
* Implement and use the Factory design pattern
* Design and implement a multi-object program, dealing with the intricacies of objects communicating with one another

**Deliverables**

* Deliverable 1: hw3\_design.pdf, csv files
  + in class Monday, Feb 26th
* Deliverable 2: Makefile, outline of all header & implementation files, main.cpp, csv files
  + Due Friday, March 1st
* Deliverable 3: Makefile, complete header & implementation files, main.cpp, output.txt, csv files, peer evaluation, design reflection, interview
  + Due Friday March 8th

# The BidToBuy Program Requirements

You will write a program that simulates eBay in CLI program called BidToBuy. You will use inheritance to define the different types of users and products in this application. Buyers can bid on products posted by the sellers and the highest bidder is determined once the bid has been closed. You have the creative freedom to design the solution for this assignment if you fulfill the following requirements.

## Program Flow:

1. Initialize application from the csv files.
2. First, the user should choose a role, that of a Buyer or Seller
3. If Seller, they should be able to:
   1. post a product for sale - should also have the option to view prices of similar products that were historically sold.
   2. read and respond to messages from buyers,
   3. check their account balance,
   4. update their user information,
   5. get an overview of their products that have been sold or are yet to be sold
   6. open/close bids on their products
4. If Buyer, they should be able to:
   1. view products for sale,
   2. place a bid on a product of their choice
   3. read, respond, send to new messages from sellers,
   4. check their account balance,
   5. update their user information,
   6. get an overview of the bids they have placed
   7. view the history of products they have bought
5. The driver (main) class will control the application interface and the flow of the program and will:
   1. Track all currently active and completed users, products and bids
   2. write new bids and bids that have closed to the product/bid information csv
   3. Implement this class as a Singleton

## Selling Products

Use factory design pattern when the seller adds a product for sale. Below is an example of a factory method you could write. Using enums will make it more robust and readable than using integers as choices in the productFactory:

enum class ProductCategory {Vehicle, Furniture, Books,....}; // used to

define all the category of products sellers can sell on this app. You may

add as many categories you like. There should be at least enough of them to

cover all the classes you created as part of the products inheritance

Hierarchy.

Product\* productFactory(ProductCategory pc) {

switch(pc){

case ProductCategory::Vehicle:

return new Vehicle(...);

case ProductCategory::Furniture:

return new Furniture(...);

...

default:

return new Product(....);

}

}

## Bidding on Products

* Each product that is placed for sale will be open to accepting bids until the seller closes the bid.
* Buyers may place a bid only products are active for bidding.
* A buyer should not be able to place a bid that exceeds their account balance. You may assume they have a fictional balance amount in their account at the start of the run.
* A buyer may place a bid on more than one product.
* Once a bid has been closed, the buyer with the highest bid will own the product.
  + The buyers should be notified through messages if they won or lost the bid.
  + Once the selling process is complete, the account balances of the buyer and the seller are updated.
  + At the time of sale, if the buyer refuses to buy the product or if the account balance of the buyer is less than the bid, the sale to that buyer is cancelled and the bid is won by the buyer with next best bid.
  + The product is then taken off the market.

## Messaging between Buyer and Seller

* A buyer may message a seller if they win the bid for the seller's product. Similarly, a seller may also message only a buyer who has won a bid for their product.
* The receivers should be able to respond to the messages.
* You may use any underlying data structure to do this. Using one that is time and storage efficient, will grant you extra credit points.

You can display the messages one of 2 ways:

* Iteratively display all the messages and have the user the option to respond to each message, or
* display all messages and allow the user to choose which messages to reply to.

## Updating user information

The users can update their name, phone number, address.

## Products

* You are required to implement at least 5 concrete product category classes each having at least one level of inheritance.
* All products should have a base price at which the bid will start.
* The seller must indicate the quality of the product when posting one for sale. The quality can be one of these: New, Used-VeryGood, Used-good, Used-okay.
* You have the freedom to decide additional behavior and data points for the classes. Plan to have detailed classes. Well-designed and detailed inheritance design will contribute to the extra credit.

### Store and contribute through Github

* Ensure that your repository is kept private and add Julie @jjarz as a collaborator to your repository.

# Deliverable 1: Design Document & user data

### Part 1: Create a design document for your program’s design

Your design document should include:

1. A diagram or table including:
   1. the classes your project will use,
   2. what methods and attributes they contain, and
   3. how they will interact with one another
2. Pseudocode for the main (driver) file.

You do not need to follow any formal diagramming language, but your diagram needs to communicate all of the points above in detail. You can whiteboard boxes and arrow or create a table.

### Part 2: Create program data

Create two CSV files containing data that your program will process, including:

1. Information about sellers and buyers.
2. Product/Bid information:
   1. Consider this to be historical data.
   2. Plan to have product information, buyer information and seller information at the least.
   3. This file will help with features where the seller can look at historical data to decide a price for their product. You do not have to normalize this data.

### Part 3: Divison of Labor

Mark areas of the project that each team member will work on. You might consider using <https://github.com/features/issues> but planning within a Google doc or other format is acceptable.

# Deliverable 2

1. Use factory design pattern when seller is adding a product for sale.
2. Write at least an outline for the user and product classes.
3. Write a main.cpp that allows you to instantiate the driver class and build a map of dummy bids.

# Deliverable 3

1. Implement the rest of your program.
2. Fill out the peer evaluation form linked on the Canvas item. Peer score contributes 10 points to your grade for this homework. Failure to fill the form by the deadline will lead to a 0 for your partner.
3. Write a reflection (2-3 bullet points or sentences per topic below) about your original design from deliverable 1 and the implementation process:
   1. What worked well? Why?
   2. What could be better in the design? How would you change that?
   3. How did you use AI for writing code for your project? Was your use of AI effective? What went well and what could have been better with respect to your use of AI for writing code for this project?
   4. If you went above and beyond requirements in any areas, note these in your reflection for extra credit consideration.
4. Schedule an appointment for interview grading.
   1. **Each individual** will complete an interview with Julie the week of March 18th.
   2. Slots will be available on Canvas by March 7th.

# Grading Rubric (150 points total)

### Deliverable 1: design (30 points)

* Diagram of program (20 points)
  + Classes that manage all project requirements
  + Include class methods
  + Include class fields
  + Include connections between classes – what methods classes will call, how they interact
* Data files (5 points)
  + Show detailed structure of data for users, products and bids
* Division of labor (5 points)

### Deliverable 2: initial coding sprint (40 points)

* Factory design pattern
* Outline for user and product classes
* Main.cpp builds a map of dummy bids

### Deliverable 3: final coding sprint (40 points)

* Remainder of program requirements are satisfied
* Written reflection on design and AI (10 points)

### Peer Evaluation (10 points)

* Average of rating 1-10 given by your project partners

### Comments and style (15 points):

* Your files and functions should have comments. We should know how to run your program and how to use your functions from your comments.
* Your variables should have meaningful names. Your code should be easily readable. If you have complex sections of code, you should use inline comments to clarify.
* You should follow the conventions set out in our Concise Style Guide, posted on the course github.

### Interview Grading (15 points)

* Each individual will schedule an interview with Julie the week of March 18th.
* Slots will be available on Canvas by March 7th.

### Extra Credit (15 points):

* Variable based on quality of design, efficiency of implementation using appropriate object design and suitable data structures for reduced time of execution.
* If you go beyond project requirements in certain areas, note these in your written reflection AND in the code comments so they will be considered in grading.