# Lab 05

## Resources

- Discussion slides (./lab05slides.pdf) - Copy of the presentation given by the TA

## Idea

In this lab, you will become familiar with going from an english description of a class messy distributed method of coding it up, to a more elegant class-based representati You should come out of the lab with a better understanding of how to design in an ol oriented manner. This should help in transforming your Project 1 into a state more s to Project 2.

## In-lab assignment

### Description

You will be implementing a sorted group of people. This group is going to be able to one member at a time as well as shrink one member at a time. Somebody using your group is also going to want a nifty way of seeing the members of the group printed to console.

We are going to assume that a person is only made up of a name and an age. We are further going to assume that somebody's name always begins with a capital letter fol by only lowercase letters and that it is only one word. So, john is not a valid name. N is JoHn, JOHN, and certainly not J0hn. John, however, is.

Your mission is to create a good object oriented implementation of this description.

## Requirements/Deliverables

You will represent your group using a class that will need the following member func

1. insert - Should take in a name and an age representing a person and add them t group. This function should return a value representing whether or not the inser successful.
2. delete - Should delete a given person from the group based on a name paramete Again, the function returns whether or not the deletion was successful.
3. print - Should print out the contents of the group in the following manner: `name age`, for each group member. Each member should be placed on a new line.

You will also need a main that displays a menu to the user and allows them to do the things.

I have already written the logic you will need for the group. It can be found here: group.cpp (./group.cpp). This file contains the functions and data you will need for a group as well as a main that implements the menu. It is not done in a proper object oriented manner though. You will need to fix this.

The following things must be done to turn this program into an acceptable object ba program:

1. You will need to make a group class. This class should contain any data member necessary to a group as well as all the functions of the group. Private and public identifiers should be used correctly.
2. You will need to break up the group and main logic into separate files like we di the last lab. File names should be: group.cpp, group.h, and main.cpp. A makefil suggested but not required for this lab.
3. Finally, instead of using paired arrays (names and ages), you should make a clas Person containing just a name and an age (you will also need to write getters an setters as these members should be private). Then, your Group class should only contain an array of type Person. You will need to edit the logic in the functions t reflect this.

## Submission

When finished, please compress your group.cpp, group.h, and main.cpp into a zip fil
called lab05.zip. Submit this archive to Sakai in the Lab 05 assignment area.

## Hints

- Get this program working incrementally. Start by wrapping up all the group logi
  is written in a class. Then, compile and make sure everything works. Then break
  into separate files. Make sure everything still works. Finally, add in the Person cl
  and fix the logic.
- You don't have to use my implementation. If you can make it work by starting fr
  scratch, do so, but it must follow the exact implementation specified above.

## Grading Distribution

- 5 points for putting the correct things in the correct file. So, no function defintic
  the .h file. All main logic in main. Only Group and Person definitions in group.cp
  etc.
- 5 points for getting the group class to work correctly (without the Person part). t
  includes correctly using the class declaration, use of private/public identifiers, e
- 5 points for correct implementation of the Person including making the Group h
  only an array of persons and correct implementations of the functions.
- 5 points for good format and style

## Optional Enhancements

- Make it so members are sorted secondarily based on age. So, if `Mike, 25` and `M
  26` were placed in the group, `Mike, 25` should be listed before `Mike, 26`
- Add in logic to catch any errors you can think of. Some of these include: incorre
  input, Duplicate people being added into the group, incorrect name format, etc.
- Modify the Person so that each person has a first and last name in addition to th
  age
- Change the logic to allow any names to be accepted as valid (think user names).
  means you will have to ignore case and decide what to do with numbers and sym
  (if you think modular here, you could write your own compare strings method th
  will compare any two strings how you want)
- Write methods that will re-sort the members based on their different parameter

other words, make a sort by age and a sort by name function. To do so, you will
to implement a sorting algorithm. Bubble sort is very simple to understand and
and will work fine for now. Quicksort or merge sort will get things done quicker,
are more difficult to understand and code.