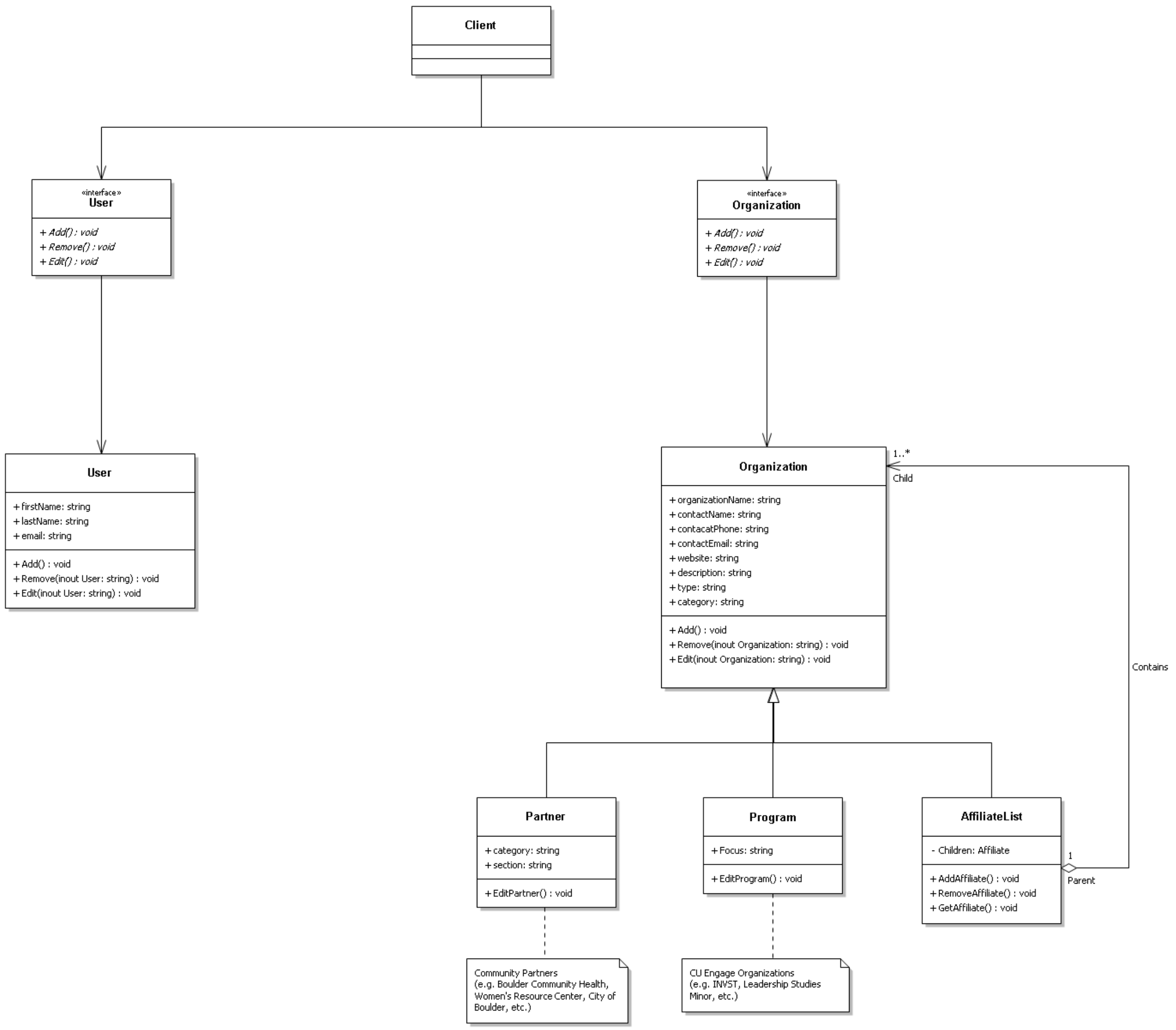
Engage
UNIVERSITY OF COLORADO **BOULDER**

Team: Erin Boeger, Edward Crawford, Kyle Knight, Josh Rinaldi

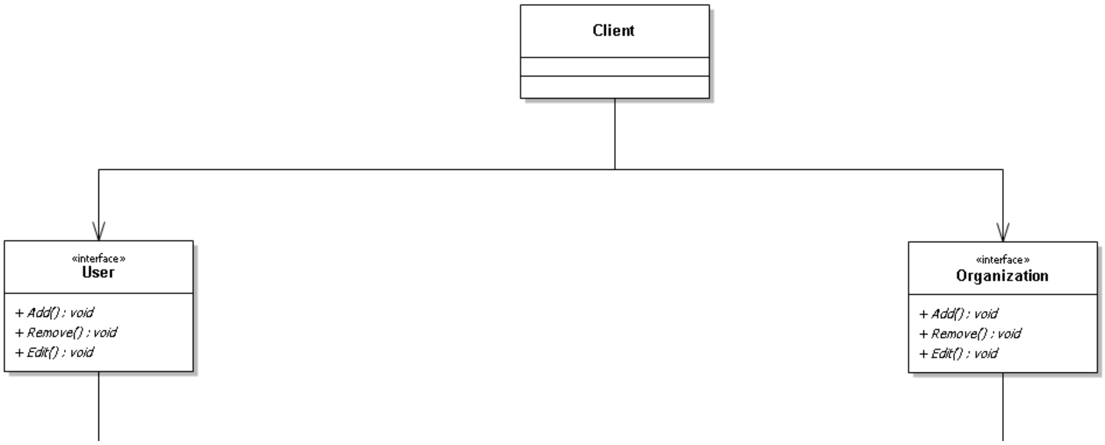
Title: CU Engage

Sponsor: Letica Sanchez (leticia.sanchez@colorado.edu)

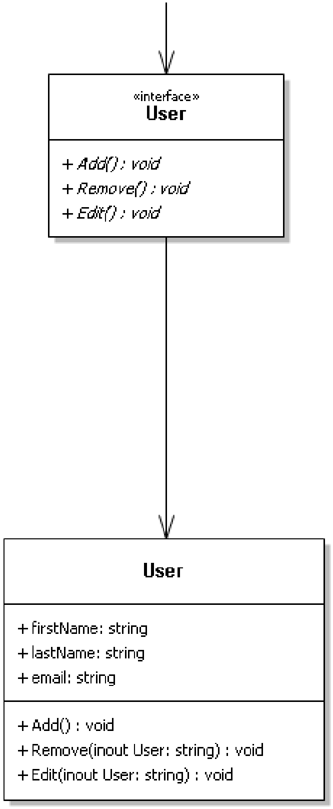
Question 1: What features were implemented? Include a class diagram showing the final set of classes and relationships of the system. Discuss what changed in your class diagram and why it changed, or how it helped doing the diagrams first before coding if you did not need to change much.



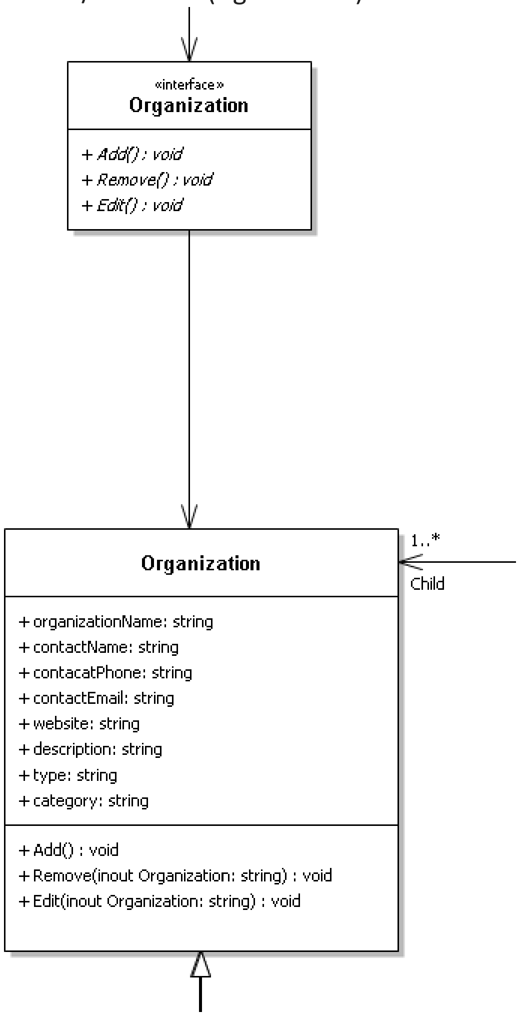
Top of diagram:



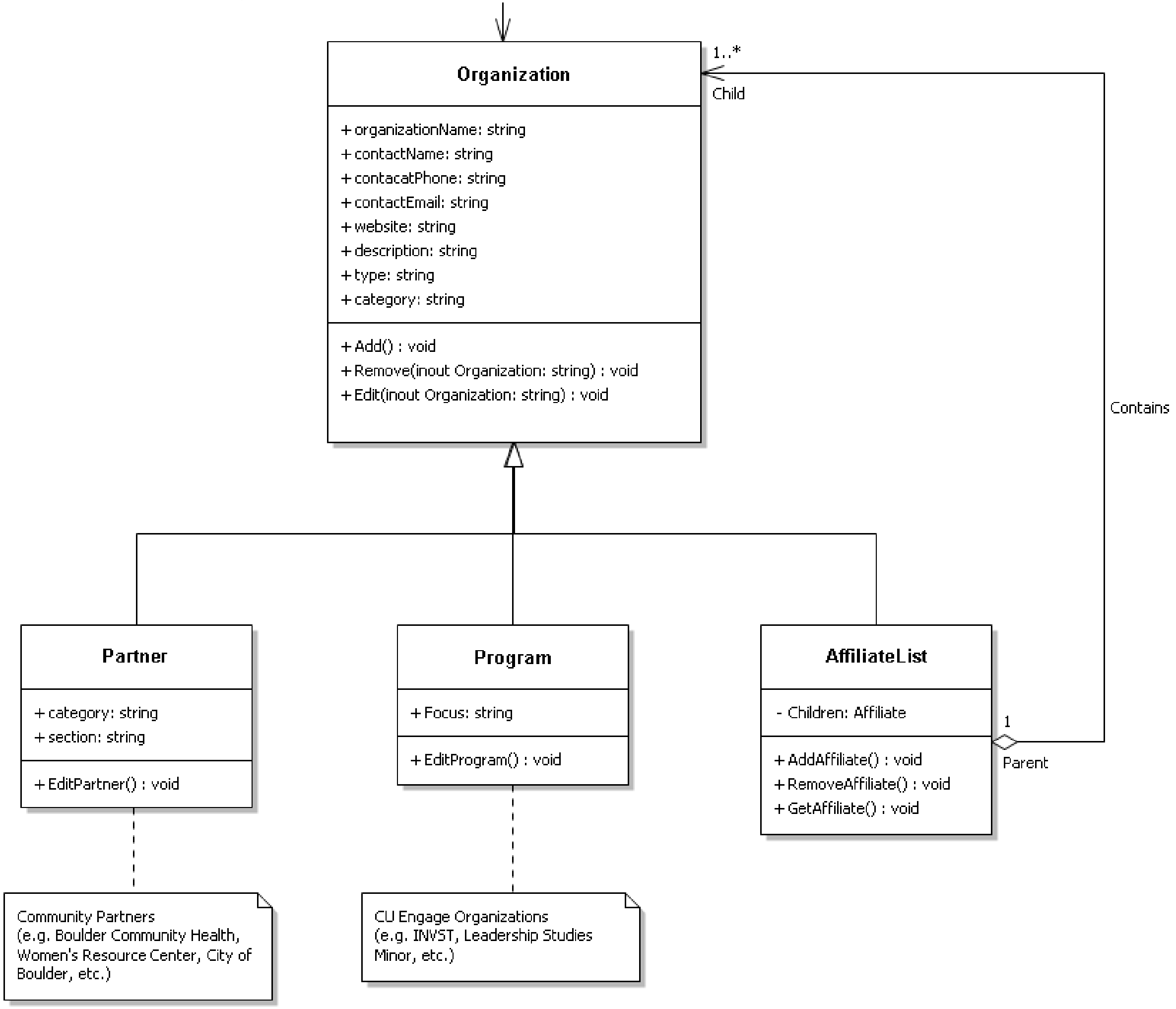
“User” section (bottom left) of diagram:



Organization/Interface (right center) section of diagram:



Organization (bottom right) section:



The features that we implemented in our program were:

1. Adding Users
2. Removing Users
3. Editing Users
4. Adding Organizations
5. Removing Organizations
6. Editing Organizations

These features are crucial to the system. Without the ability to add organizations, the entire purpose of this program would be lost. We also wanted to be able to control who has access to the program, hence why we have the ability to add users. Now, as circumstances change over time (e.g. organizations change, or are disbanded, and Users get new contact info or leave the program) it is necessary to have the ability to remove and edit both organizations and users.

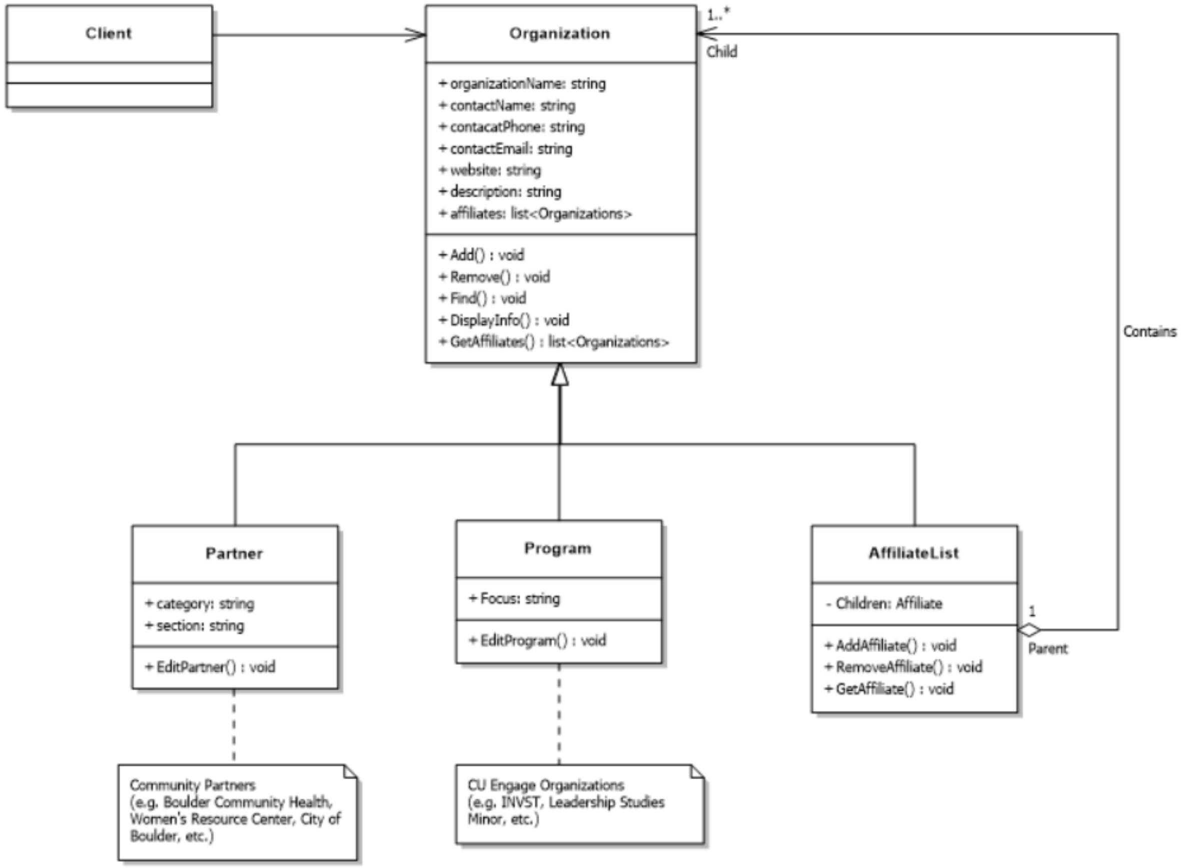
The biggest benefit is that you have a road map of what needs to be coded before you even start coding. Designing before-hand provides you with a skeleton that simply needs to be fleshed out, and it gives you the opportunity to spot potential pitfalls before they are even an issue.

Question 2: Did you make use of any design patters in the implementation of your final prototype? How?

One of the design patterns that we used in our project was the Composite Pattern. In our project, we have various organizations, each of which works with a number of other organizations. This list of affiliated organizations is something that we wanted to keep track of, and we found that the Composite Design pattern very nicely took care of this problem for us.

Question 3: Discuss how the final system changed from the design presented in Project part 2. Include both the original, and the updated class diagrams. Compare and contrast the two.

Project part 2 diagram



The biggest change from our original class diagram is the addition of Users. As discussed in the first questions, we felt it was necessary to limit who has access to the system, so we added in users for that purpose. In addition to this, we also cut down on the number of methods that are in the Organization class, as we felt that there were just more than were actually needed. This is always a good thing to know, because it is good to be cognizant of just how complicated (or uncomplicated) a program needs to be; you should never make a program more complicated than it needs to be, because it can lead to redundancies, useless code, unnecessary functionality and code that is near impossible to make sense of.

Question 4: What have you learned about the process of analysis and design?

The big lesson that we learned was not to jump right into coding. This is the path that we originally took, and we ended up going the complete wrong direction with this project because of that. It took a long time to come up with not a lot product, and the road to getting to that product was rocky at best. We hit a wall, and decided to start the project again from square 0. We made ourselves step through the design process and did not concern ourselves with the specifics, and in the end, we found that the design process was actually quite a bit smoother, and quite a bit easier the second time through. It was nice to actually have a full design for our project before we even wrote a single lick of code because it gave us a fantastic guideline to follow when we actually started coding. Essentially, what we learned was that it is essential to design the program to the fullest extent possible (at that juncture) before you start coding.