# 1 480 April 29, 2013 Notes,

## 1.1 QA assignment, it will be posted on April 30, and the same group will work together again, and using the same bitbucket account

**Techniques we will have to use**

- Coverage, unit testing, have to cover almost 100 %, have to find bugs too. Can't just cover 100 % without finding any bugs. Put bugs explanation in your reflection.
- Code Review.
- Wirte Blackbox test, based on specification completely.

**What we should not do in this assignment**

- We can't have each person do one technique because we all need to know this shit.
- More communication among team members(bullshit!!!).

**Suggestion**

- you splits number of methods.
- each apply 3 QA techniques.

## 1.2 UML diagram

- We need abstraction to understand big program.**Definition**: And the abstraction that we use is UML( stands for Unified Modeling Language), this is a graphical language, there are 16 UML diagram types
- We will learn 4 types: - UML are blueprint of our program, an abstraction of the program, it tells us important infomation of the program. For example: blueprint of a building, like about wireless, electricity, water system diagram
- We can apply the same concept, so we can easily track thing in our program instead of directly looking at the code

- Strutural vs Behavioral()
- Static vs Dynamic( same as static and dynamic analysis)

## 1.3 Class diagram

- **Example**: Class diagram is a static and structural. When people said draw them a uml diagram, they refer to class diagrams. The look of class diagram is like the one that we saw when we learnt about class in our 141 class.
- Diagram:
Student——— name
String name —— field
int id ——field
string name()—— method
- We usually don't even look at the field because they are private, we pay more attetion to our methods because they are the ones that we will use.

- There are 3 ways that we can connects class together. - **Connection types in class diagram**

- 1st is **inheritance**.
- 2nd is **composition**: means own it.
- third one we haven't learn is **agregation**: kinda like composition but it is now owning. For example: student can have course but student does not own course, but student owns names, ID(composition)

**Note:** The differences between agregation and composition becomes really important when it comes to C++ because you have to delete your object after you are done with it, and who deletes whom really matters because you don't want to forget to delete, or double deletion. **Other than connections between classes.** - You should sometimes use dash arrows for calls, or detects.

**Figure right here**

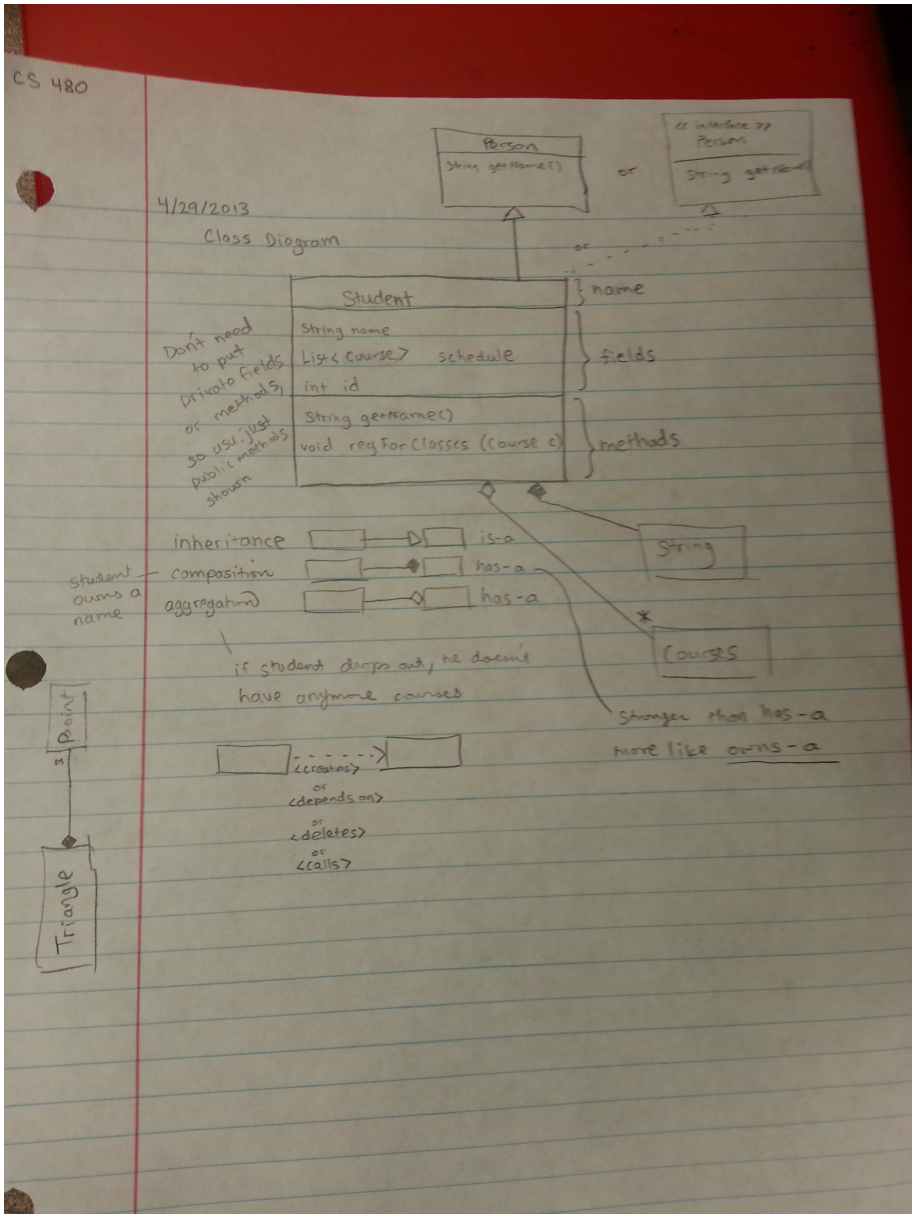


Figure 1: Structure Diagram

## 1.4 Object Diagram

**Properties**

- Dynamic
- Structural

**Figure right here**

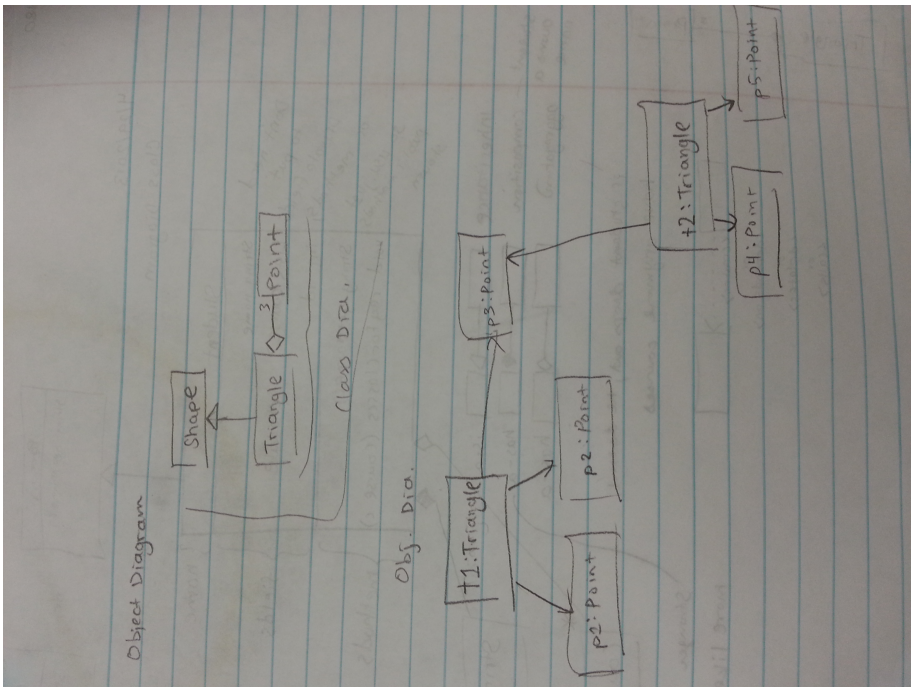


Figure 2: **Notes:** Object diagram

## 1.5 Sequence Diagram

**Properties**

- Dynamic
- Behavioral

**Figure right here**

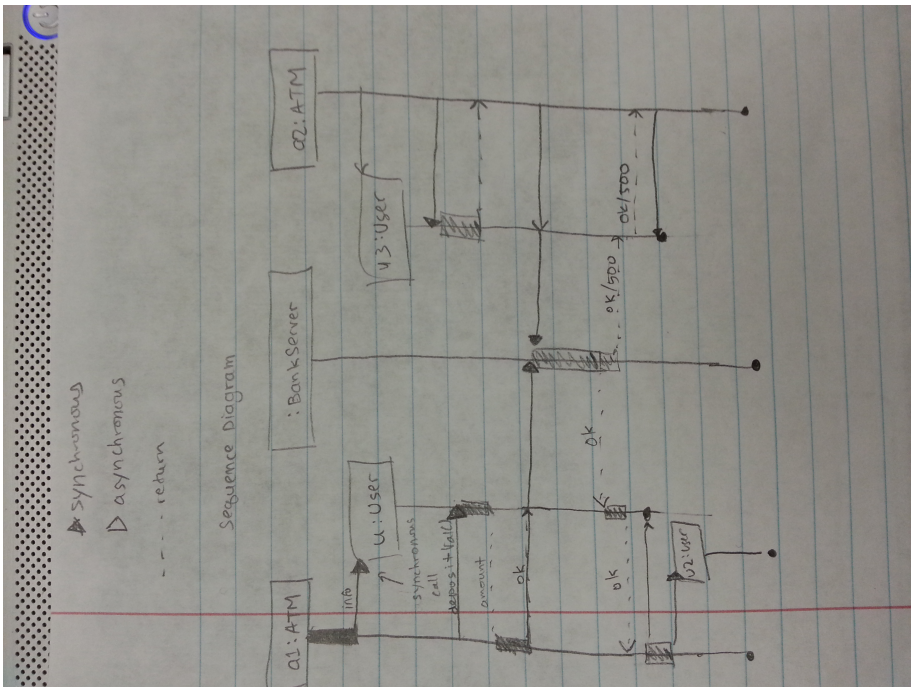


Figure 3: **Notes:** Dash arrow for returning, solid for giving infomation

## 1.6 State machine Diagram

**Properties**

- static
- behavioral

**Note:** The diagram of this type looks exactly like automata diagram