Course Learning Goal 4
Designing Solutions with **Object Oriented** Software Engineering

Goals

1. See how to approach software design in an **object oriented** manner.
2. **Define classes** - properties and behaviors
3. **Create objects** from classes.
4. **Constructor**

Imagine you are asked to design a software solution for some problem (e.g. education, engineering, business, etc...)

What should you do?

1. **Understand what you need to do**
2. **Determine what software objects need to be created from 1.**

Software Engineering Activities

- Define the requirements
- Analyze the requirements
- Design the System
- Implement the Design
- Test the Implementation
- Deploy the Implementation
- Maintain the Deployed Software

Activities don’t always proceed sequentially

Problem: Build an **Information Database** for Whitworth University

Define the requirements

The requirements say nothing about how the software will work internally!

- What will the software do?
- Who will use the software?
- Talk to the customer. What do they want?
- How will they use it?

Define the Requirements

Each student has a name and an ID. A student can be male or female. A student has an advisor. The advisor is a faculty member. A student enjoys studying and doing homework. A student can also register for classes, or change advisor.

Analyze the Requirements

What are the nouns? (Potential classes or class properties)

What are the verbs? (Potential object behaviors)

Each student has a name and an ID. A student can be male or female. A student has an advisor. The advisor is a faculty member. A student enjoys studying and doing homework. A student can also register for classes, or change advisor.

Analyze the Requirements

What are the nouns?
(Potential class objects or class properties)
What are the verbs?
(Potential object behaviors)

Each student has a name and an ID. A student can be male or female. A student has an advisor. The advisor is a faculty member. A student enjoys studying and doing homework. A student can also register for classes, or change advisor.

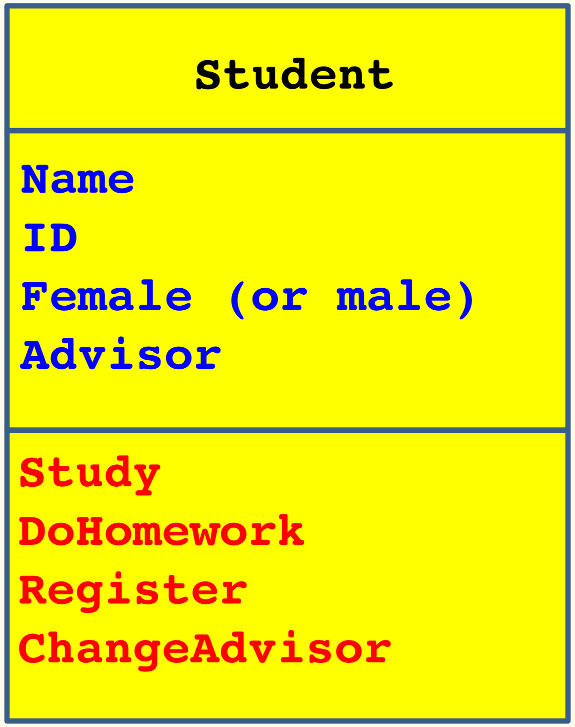
Design the System

We model the important classes of objects using the **Unified Modeling Language** (UML)

Focus on the important classes to define their **properties** and **behaviors**

Student

UML DIAGRAM



Student

Name
ID
Female (or male)
Advisor

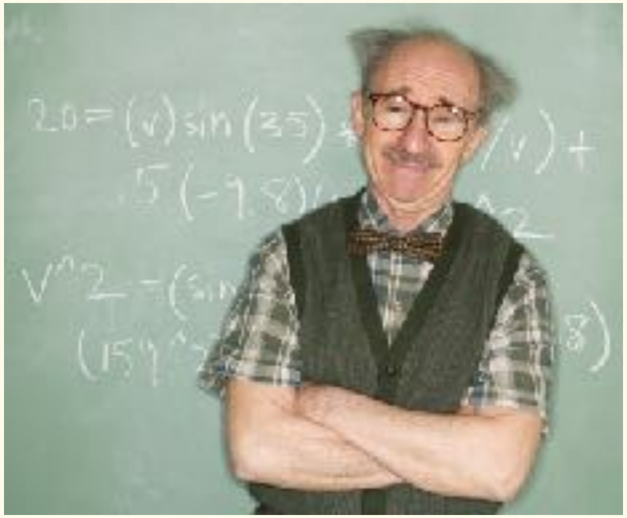
Study
DoHomework
Register
ChangeAdvisor

Design the System

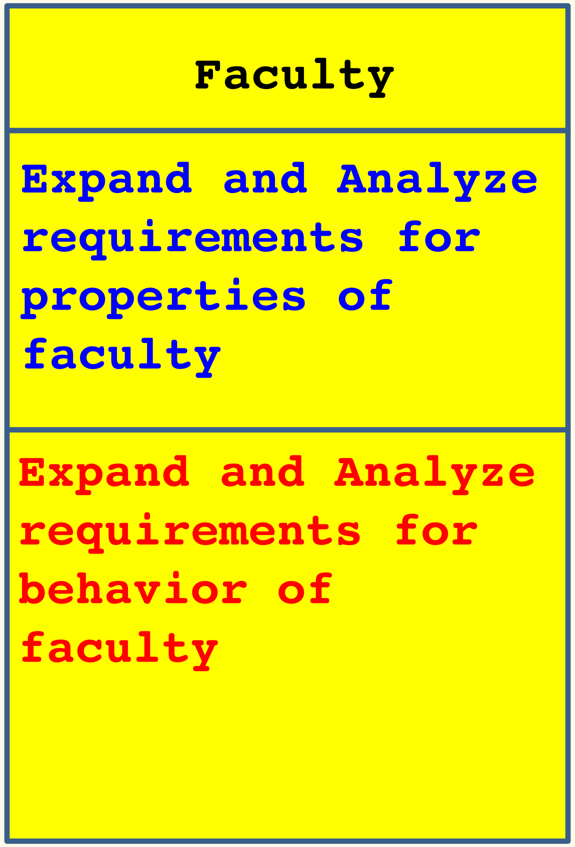
Your turn

Come up with requirements for **faculty** members and analyze it for faculty properties and behaviors

Faculty



UML DIAGRAM



Faculty

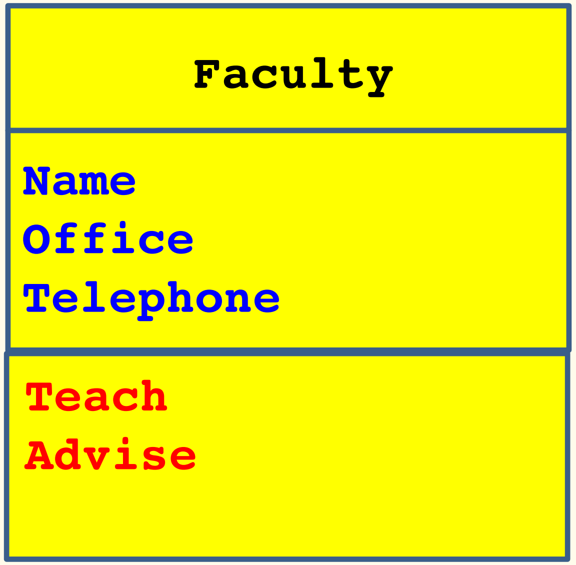
Expand and Analyze requirements for properties of faculty

Expand and Analyze requirements for behavior of faculty

Define and analyze the Requirements

Each faculty member has a name. The faculty member has an office and a telephone number. Faculty members love to teach and advise students.

UML DIAGRAM



Faculty

Name
Office
Telephone

Teach
Advise

Refining the Design

1. Add data types
2. Convert behaviors to functions

UML DIAGRAM

Student
Name ID Female Advisor
Study DoHomework Register ChangeAdvisor

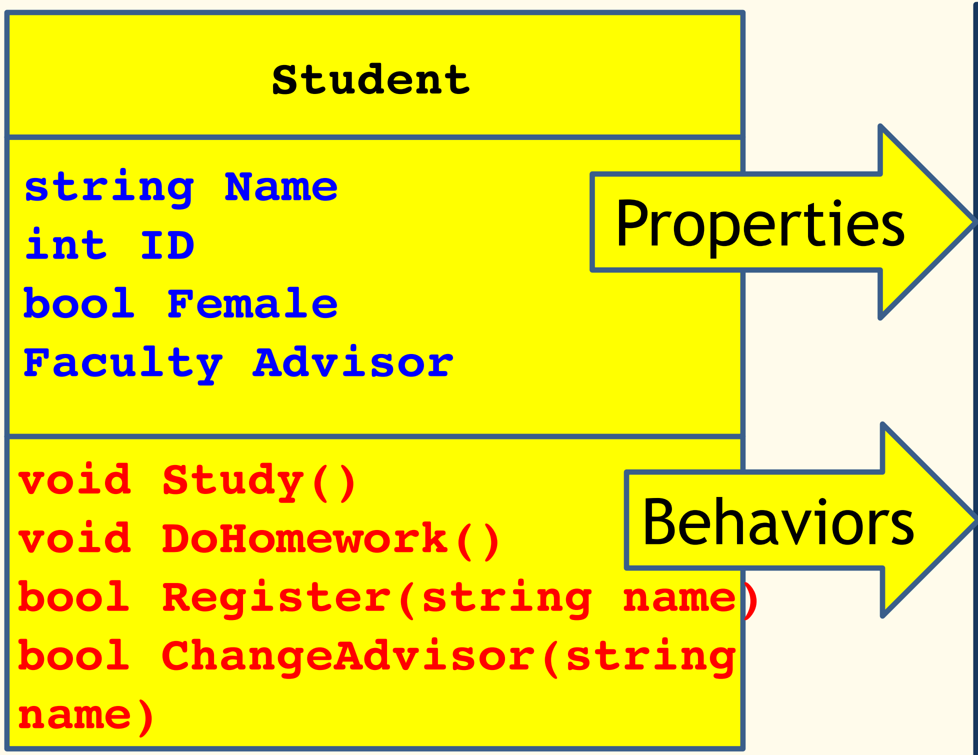
Expanded UML DIAGRAM

Student
string Name int ID bool Female Faculty Advisor
void Study() void DoHomework() bool Register(string name) bool ChangeAdvisor(string name)

Your turn … refine the design for Faculty

Implementation of Design

UML CLASS



Student

string Name
int ID
bool Female
Faculty Advisor

void Study()
void DoHomework()
bool Register(string name)
bool ChangeAdvisor(string name)

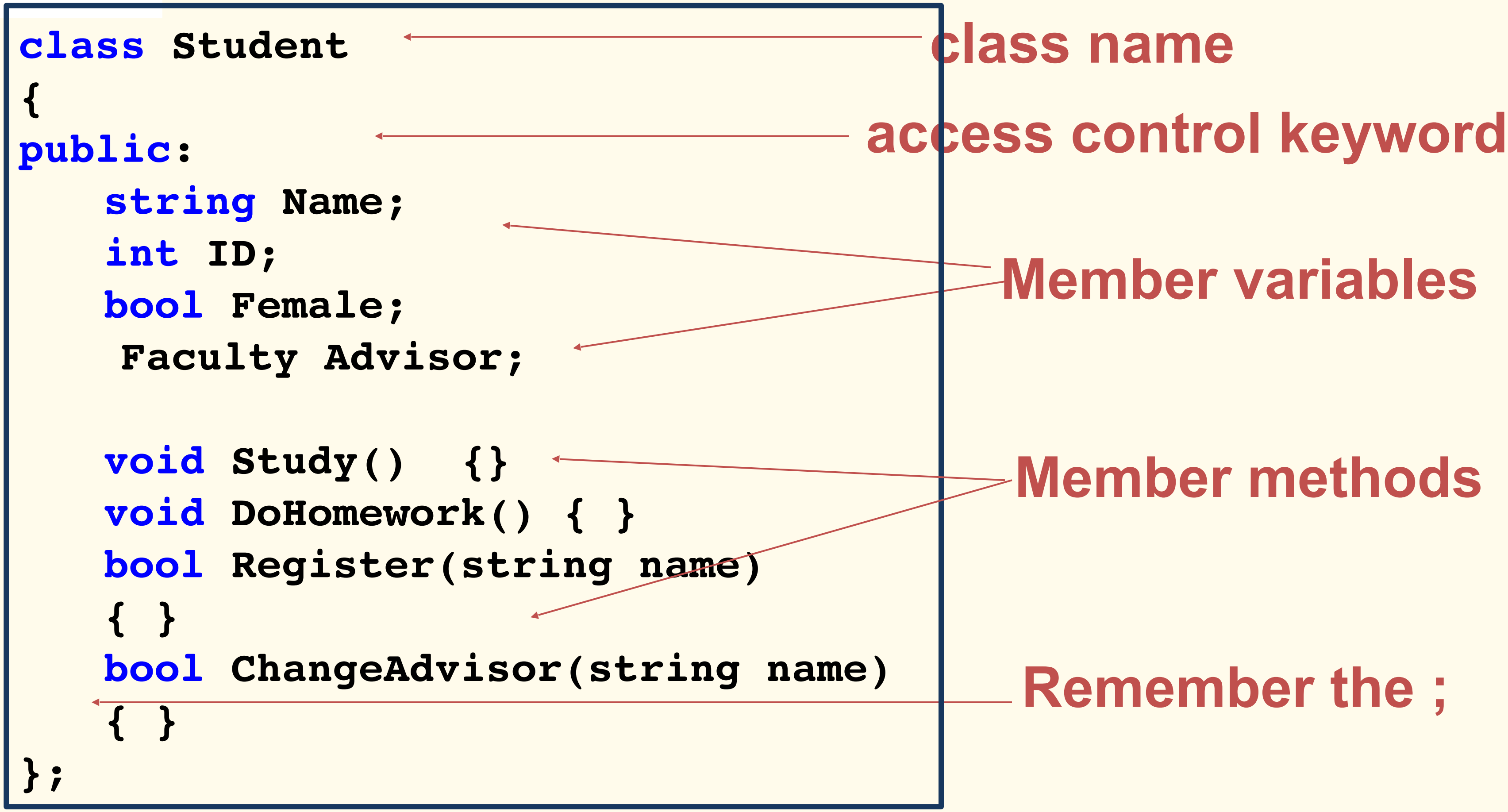
Properties

Behaviors

C++ CLASS

```
class Student
{
public:
    string Name;
    int ID;
    bool Female;
     Faculty Advisor;

    void Study()  {}
    void DoHomework() { }
    bool Register(string
    name)
    { }
    bool ChangeAdvisor(string
    name) { }
};
```

A **class** declaration defines a new data type using existing types



```
class Student
{
public:
    string Name;
    int ID;
    bool Female;
     Faculty Advisor;

    void Study()  {}
    void DoHomework() { }
    bool Register(string name)
    { }
    bool ChangeAdvisor(string name)
    { }
};
```

Your turn … implement the design for Faculty

You can make objects from your new classes

- The **class** definition defines a blueprint for making **objects** of your new type.
- To actually define an object, (i.e. variable) use the class name as the type

```
Student S1;
```

S1 is now an object of class type **Student**.

Use the dot (.) operator to access to the public members of objects

```
Student S1, S2;
Student S3;

S1.Name = "Mike";
S2.Name = "Jill";
S3.Name = "Bob";

S1.Study();  // Make Mike study!
S2.DoHomework(); // Make Jill do her homework!
```

Your turn. Write code to make faculty objects, and make them do something

We can also define a *Constructor* to Initialize Class Data Members

- A *constructor* is a special function that is used to initialize the member variables of the class when an object is created.
  - The constructor name must be the same as the class name.
  - The constructor must have no return type.
- A constructor is called automatically when an object is created!!

Since constructors are simply functions, we can define overloaded constructors

```
class Student
{
public:
  // Two overloaded constructors
  Student(string stu_name,
          bool female_flag = true){
        name = stu_name;
        female = female_flag;
  }
  // No-Arg constructor
  Student(){ }

  ...
};
```

Using different class constructors

```
// Create a anonymous student
// with no-arg constructor
Student s1;

// Create a female student called
// Jane - use default argument
Student s2("Jane");

// Create a male student called
// Justin
Student s3("Justin", false);
```

Access Control keywords

Controls who can access the properties and behavior of an object.

- public
  - **Any one** can access externally.
- private
  - Only the **object itself** can access internally.
- protected
  - Like private, but **subclasses has access** (we will cover this later this semester).

Class Exercise

- Let’s define some **private** properties/behaviors for the **Student** class
  - Student ID
  - Registered classes
  - GPA

Section 1&2 are here

What if you need to **get** (or **set**) the value of a private data member?

```
class Student {
private:
        int ID;
public:
        int getID() { return ID; }

        void setID(int new_id)
         {     // check if new_id is in a valid range
              if (new_id > 0 && new_id < 100000)
                 ID = new_id;
        }
};
```

Get function

Set function

Class Definition File

- Justin is assigned to work on the **Student** class
- Sara is assigned to work on the **Faculty** class

We will put the class definitions in their own *.h file (class definition file)

Faculty

Faculty.h

```
#include <string>
using namespace std;

class Faculty {
   string name;
   string office;
   string telephone;

   void Teach();
   void Advise();
};
```

Faculty.cpp

```
#include "Faculty.h"

void Faculty::Teach() {

}

void Faculty::Advise() {

}
```

Student

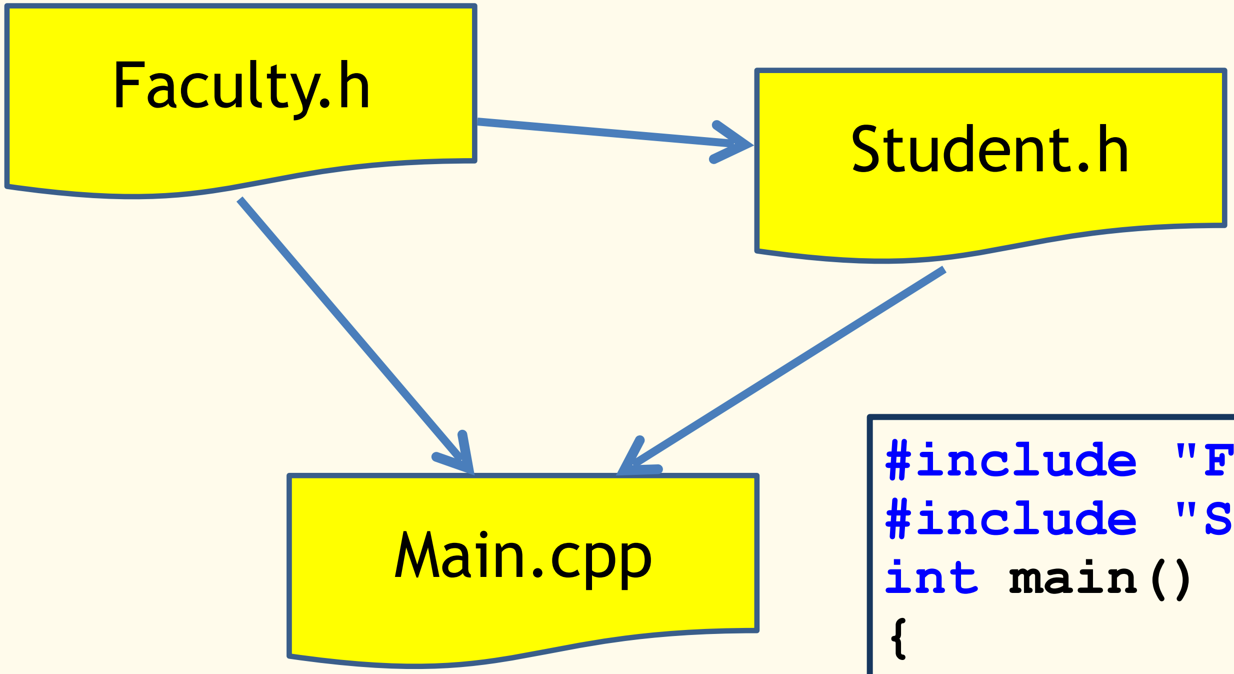
Student.h

```cpp
#include <string>
#include "Faculty.h"
using namespace std;

class Student {
   string name;
   int ID;
   bool female;
   Faculty advisor;

   void Study();
   void DoHomework();
   bool Register(string name);
   bool ChangeAdvisor(string
name);
   void Speak();
};
```

Student.cpp

```cpp
#include "Student.h"

void Student::Study() {

}

void Student::DoHomework() {

}

bool Student::Register(string name){

}

bool Student::ChangeAdvisor(string name)
{

}

void Speak() {

}
```

Prevent multiple definitions of a class


Faculty.h

Student.h

Main.cpp

```
#include "Faculty.h"
#include "Student.h"
int main()
{
    Faculty president;
    Student joe;
}
```

We are including Faculty.h twice!

Inclusion Guard in header files

```
#ifndef FACULTY_H_
#define FACULTY_H_

  class Faculty
  {
  public:
    void Teach();
    void Advise();
    …
  };
#endif
```

Preprocessor directive checks if macro **FACULTY_H_** has been defined!

Separate implementation from definition

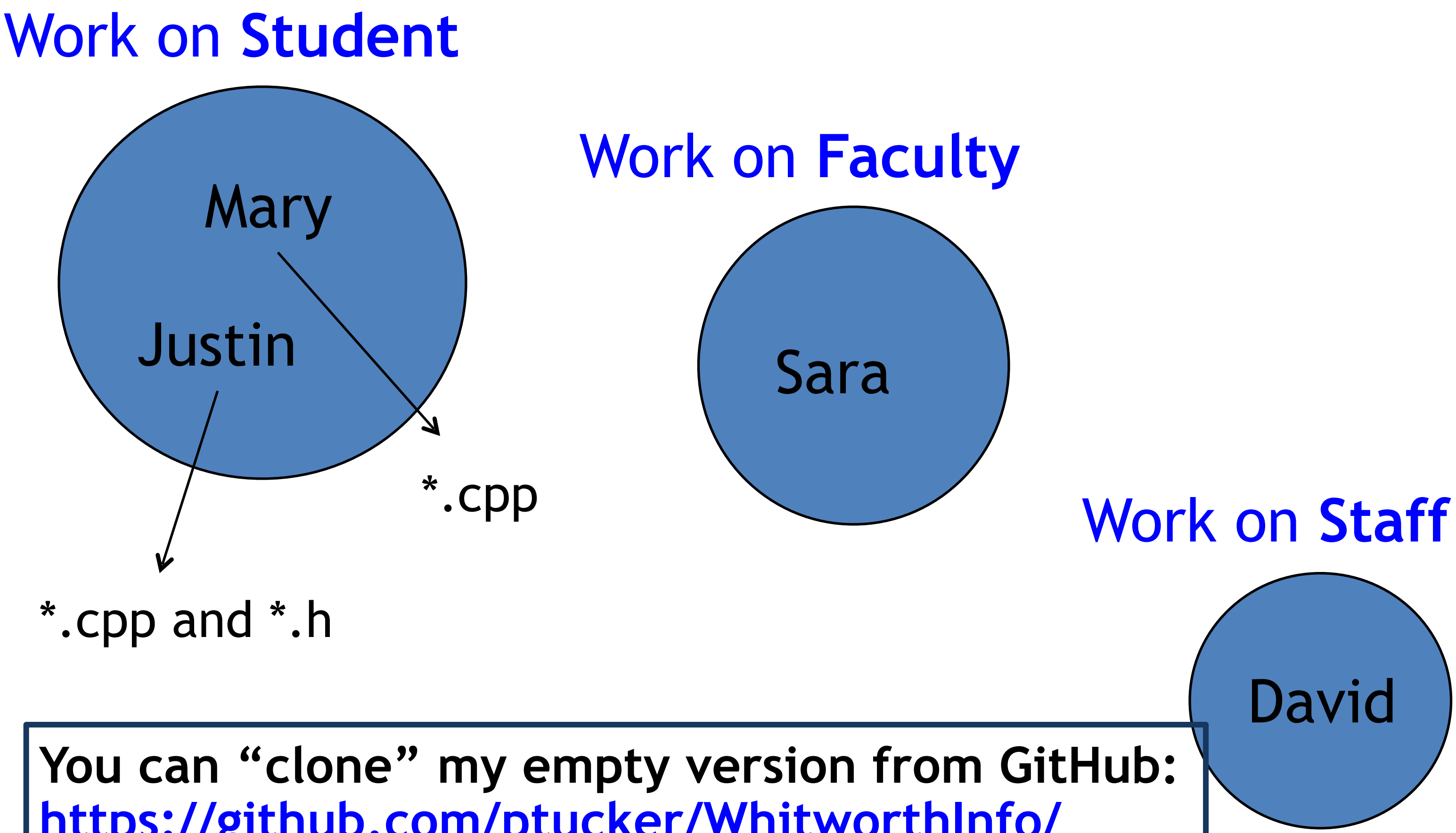
```
class Faculty
{
private:
     …
public:
     void Teach()
     {
     }
};
```

inline function

However, Sara needs help, and asks Mary to her help implement some functions.

We can separate the implementation from its definition

Software Engineering



Work on **Student**

Mary

Justin

*.cpp

*.cpp and *.h

Work on **Faculty**

Sara

Work on **Staff**

David

You can “clone” my empty version from GitHub:
https://github.com/ptucker/WhitworthInfo/

Homework Assignment 3

- Part 1 (chapter 9)
  - Due Feb 23
  - Please start early, i.e. today ☺

Summary

- We saw how to approach software design in an **object oriented** manner.
- We saw how to **define classes**.
- We saw how to **create objects** from classes.
- We studied what is a **constructor**, and created **overload constructors**
- Learned about **private, public, and protected access**
- Learned how to **separate implementation from definition** for classes

Object Oriented Design Helps: From Analysis to Implementation

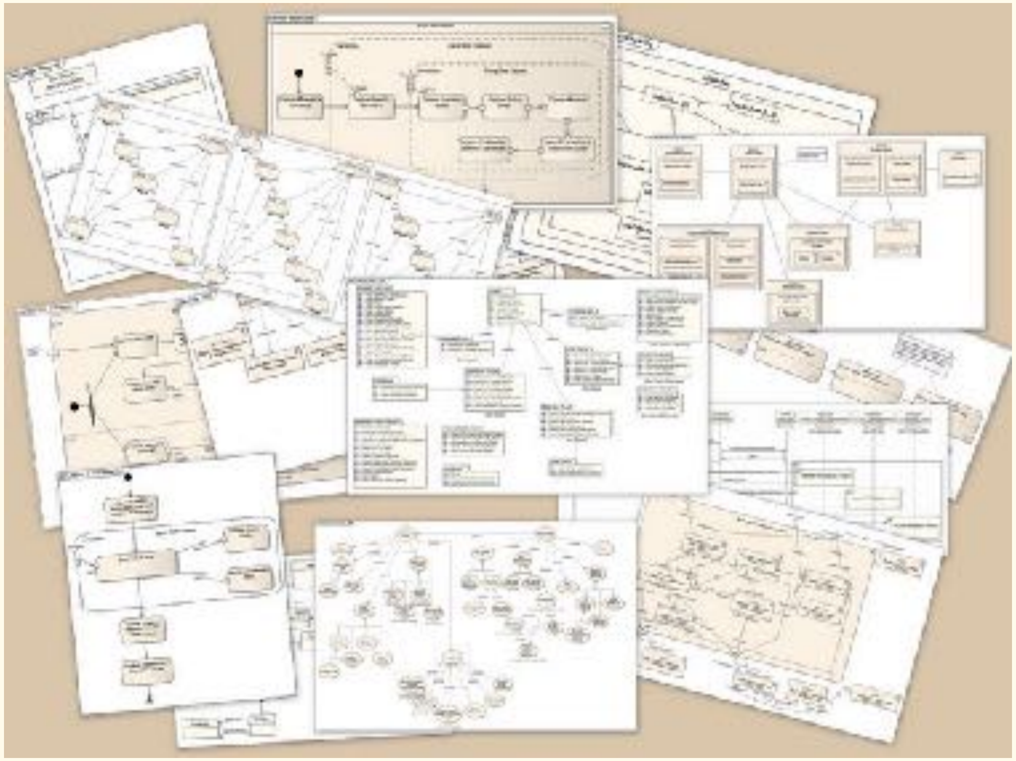


Image Source: http://en.wikipedia.org/wiki/Unified_Modeling_Language