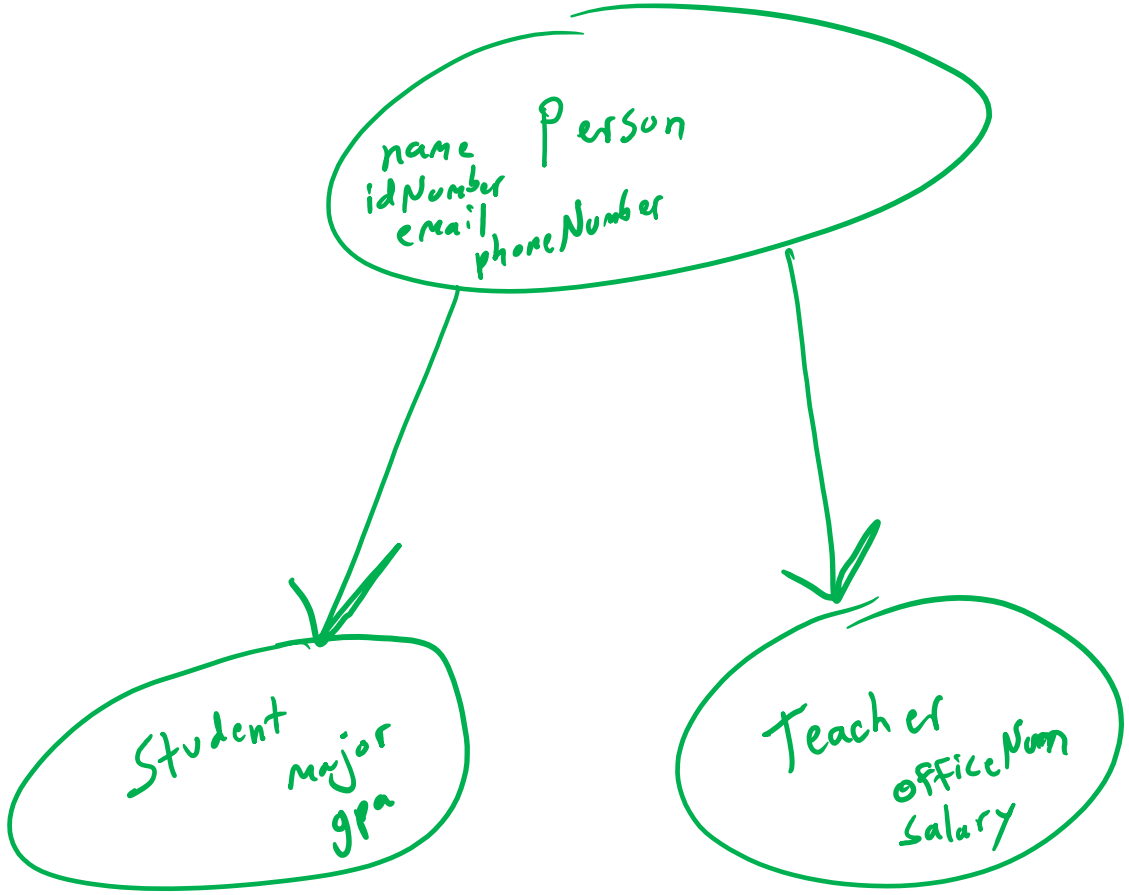
Introduction to Inheritance

- Suppose that we implement two C++ classes with the following member variables:
  - Student
    - ID number
    - Email address
    - Phone number
    - Major of Study
    - GPA
  - Teacher
    - ID number
    - Email address
    - Phone number
    - Office number
    - Office hours
    - Salary

Is there a better way to handle this?



Inheritance

- “**is-a**” relationship

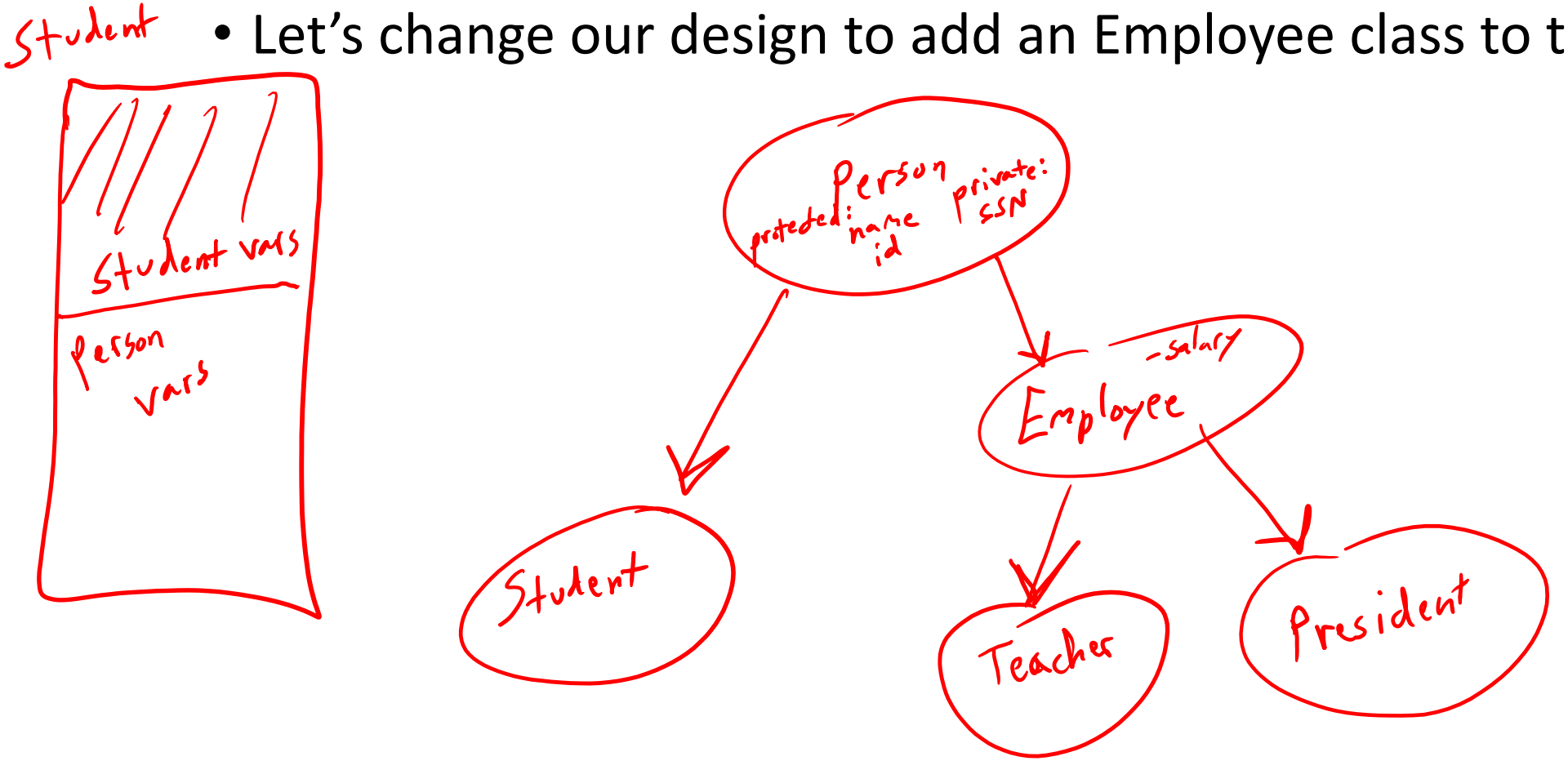
- A student **is a** person
- A teacher **is a** person
- Student and Teacher class can both inherit from the Person class

Basics of Inheritance

- Classes that inherit properties are “derived classes”
  - Also known as “child” class
- The “parent” class is referred to as a “base class”
- Helps us avoid re-inventing the wheel
  - If a Student and an Instructor are both derived classes, we don’t need to write the same code twice
  - **Person** class could hold any redundant information

Inheritance cont…

- Inheritance is not limited to a single level
  - Let's change our design to add an Employee class to the hierarchy



Member access specifiers:

- public
- private
- protected – Allow inherited classes to access members

Note: Make sure you do the assigned reading! (Canvas)

Member access specifiers:

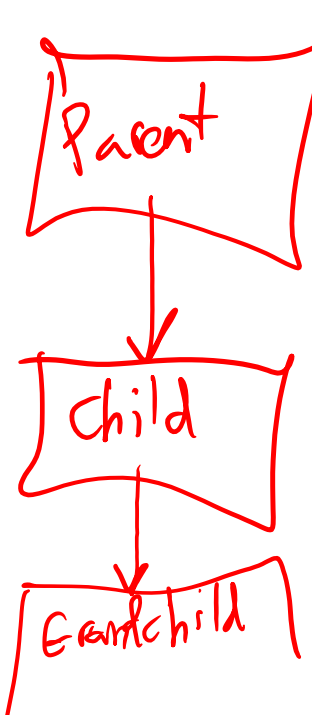
public
private
protected – Allow inherited classes to access members

Note: Make sure you do the assigned reading! (Canvas)

Base access specifiers: ← Used during inheritance

public
private
protected

Most of the time this is a good fit.



Public Inheritance

(99% of the time)

Animal	Monkey
public →	public
protected →	protected
private →	No access

Protected Inh.

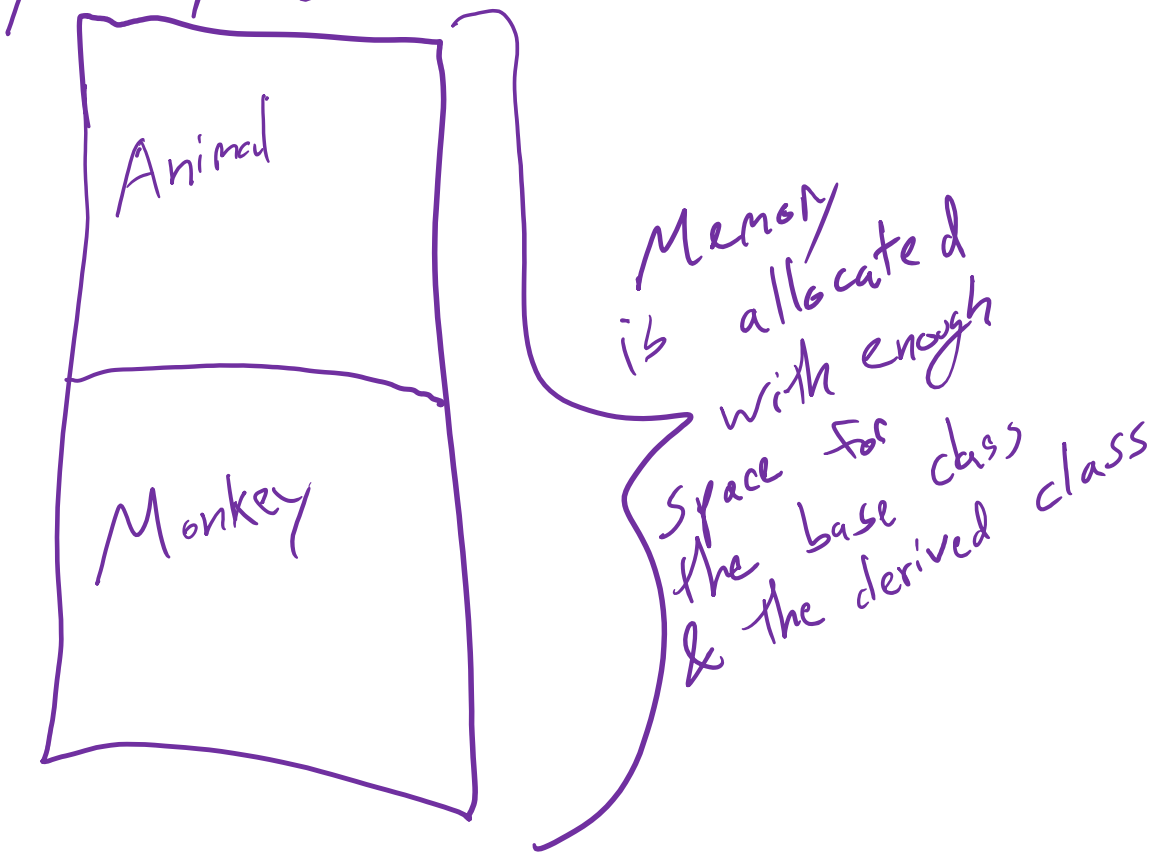
Animal	Monkey
public →	protected
protected →	protected
private →	No access

Private Inh.

Animal	Monkey
public →	private
protected →	private
private →	No access

Construction of Inherited classes

Monkey Object



Memory is allocated with enough space for the base class & the derived class

First the Animal constructor is invoked, then the Monkey constructor.