CS 162
Intro to CS II

“Has a” vs. “Is a” Relationship
```cpp
#ifndef PARA_H
#define PARA_H
#include "./mystring.h"

class paragraph {
    public:
        paragraph();
        const string * get_sentence() const;
        void set_sentence(const string &);
    private:
        string sentence[5]; // have a relationship
};
#endif
```
```cpp
#include "./paragraph.h"
#include <iostream>

paragraph::paragraph() : sentence("hello"){
    for(int i=0; i<5; i++)
        sentence[i]=string("hello");
}

const string* paragraph::get_sentence() const {
    return sentence;
}

// We'll write set_sentence later
```

"paragraph.cpp" 16L, 295C 11,1 All
#include <iostream>
#include "./mystring.h"
#include <fstream>
#include <stdio.h>
#include "./paragraph.h"

using std::cout;
using std::endl;
using std::fstream;
using std::ios;

int main() {
    paragraph p;
    cout << p.get_sentence()[2].at(1) << endl;
    string str2("hello");
    //string str=str2; //supposed to call copy constructor to make new
"main.cpp" 40L, 823C 14,3
What is inheritance?

• Webster Definition?
  – the reception of genetic qualities by transmission from parent to offspring
  – the acquisition of a possession, condition, or trait from past generations

• CS Definition?
  – Base class (Parent) and Derived class (Child)
  – Ancestor class and Descendant class (generations)
Inheritance Interface

class parent {
    public:
        parent(); //Have a constructor
        void print_mssg();
        int get_shared_var();
    private:
        int shared_var;
};

class child : public parent {
    public:
        child(); //This constructor needs to call parent() constructor
        void print_mssg(); //Redefine or Override inherited function
    private:
        int unique_var;
};
Inheritance Implementation

parent::parent() {
    shared_var = 0;
}
int parent::get_shared_var() {
    return shared_var;
}
void parent::print_mssg() {
    cout << "I'm parent!" << endl;
}

//child class implementation
child::child() : parent() { //Need to call inherited constructor first
    unique_var = 0;
}
void child::print_mssg() {
    cout << "I’m child!" << endl; //This will take precedence over parent
}
What is not inherited?

• Constructors
• Destructors
• Friends
• Assignment Op Overload

• Inherited, but not accessible: Private Members
Demo: Vehicle Toll ...

• Get into groups of 4-5 people
• Design the classes for a vehicle and bike to provide the toll amount based on the seats for all vehicles, except bikes that are free.
  – Non-default constructors to set the seats
  – Accessor function for the seats
  – Provide toll amount for vehicles and bikes
• How will you make sure it is working?
#ifndef V_H
#define V_H

class vehicle {
  private:
    int seats;
  public:
    vehicle(int);
    int get_seats();
    virtual int get_toll();
};
#endif

"Vehicle.h" 11L, 163C
#include "./Vehicle.h"

vehicle::vehicle(int x) {
    seats=x;
}

int vehicle::get_seats(){
    return seats;
}

int vehicle::get_toll(){
    return 20*seats;
}
#ifndef B_H
#define B_H
#include "./Vehicle.h"

class bike : public vehicle {
    public:
    bike(int);
    int get_toll();
};

#endif
```cpp
#include "./bike.h"

bike::bike(int x): vehicle(x){}

int bike::get_toll(){
    return 0;
}
```
```cpp
#include <iostream>
#include "../Vehicle.h"
#include "../bike.h"
using std::cout;
using std::endl;
int main()
{
    vehicle v(4);
bike b(1);
    vehicle *vpotr = &b;
    cout << v.get_seats() << endl;
    cout << b.get_seats() << endl;
    cout << v.get_toll() << endl;
    cout << b.get_toll() << endl;
    cout << vptr->get_seats() << endl;
    cout << vptr->get_toll() << endl;
    return 0;
}
```