CS 162
Intro to CS II

“Has a” vs. “Is a” Relationship
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?
Protected vs. Private Inheritance

• Public, protected, and private members
  – Who has access to these?

• Public, protected, and private inheritance
  – What does this mean?
    class child : public parent { ... }
    class child : protected parent { ... }
    class child : private parent { ... }
Protected vs. Private Inheritance

• Protected
  – Public members are protected in child

• Private
  – All members are private to child

• Either case:
  – Breaks “is a” relationship
Operator Overload Non-Member...

bool operator == (const Point &, const Point &);
int main () {
    Point p1, p2(2, 2);

    // How do we test if(p1 == p2)?

    return 0;
}
bool operator == (const Point &p1, const Point &p2) {
    if(p1.get_x() == p2.get_x() && p1.get_y() == p2.get_y())
        return true;
    else
        return false;
}

What if you don’t want to go through the accessor/mutator functions?
Operator Overload/Friends...

bool operator == (const Point &p1, const Point &p2);
int main () {
    Point p1, p2(2, 2);

    // How do we test if(p1 == p2)? return 0;
}

bool operator == (const Point &p1, const Point &p2) {
    if(p1.x == p2.x && p1.y == p2.y) // How do we have direct access to data members!
        return true;
    else
        return false;
}


Operator Overload Non-Member

class Point {
    public:
        Point();  //Default Constructor
        Point(const int x_val, const int y_val);  //Non-Default Constructor
        void set_xy(const int theX, const int theY);  //Mutator Function
        int get_y() const;  //Accessor Function
        int get_x() const;  //Accessor Function
        friend bool operator == (const Point &, const Point &);
    private:
        int x;
        int y;
};
Point::Point(const int x_val, const int y_val) { x=x_val; y=y_val; }
Point::Point() { x=0; y=0; }
Operator Overload Member

class Point {
    public:
        Point(); //Default Constructor
        Point(const int x_val, const int y_val); //Non-Default Constructor
        void set_xy(const int theX, const int theY); //Mutator Function
        int get_y() const; //Accessor Function
        int get_x() const; //Accessor Function
        bool operator == (const Point &p2) const;
    private:
        int x;
        int y;
};
Point::Point(const int x_val, const int y_val) { x=x_val; y=y_val; } 
Point::Point() { x=0; y=0; } 
bool Point::operator == (const Point &p2) const {
    if(x == p2.x && y == p2.y)
        return true;
    else
        return false;
}