Comp 11 Lectures

Mike Shah

Tufts University

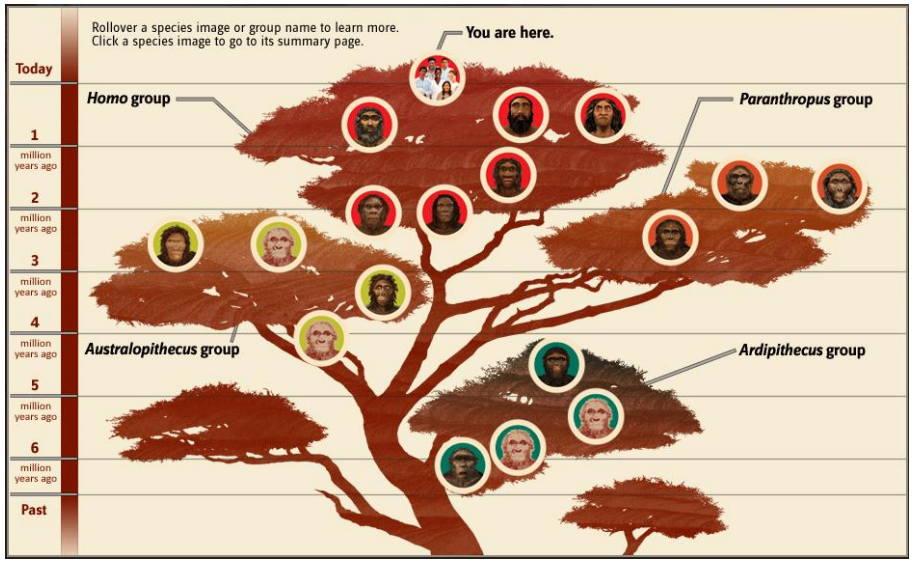
July 19, 2017

Please do not distribute or host these slides without prior permission.

Object Oriented Programming 3

Comp 11 - Pre-Class warm up

Your family tree holds information about your past. Specifically, it has information about who your ancestors are. From your ancestors, you inherited a variety of their genetics, which express certain properties that make you well–you (hair color, eye color, etc.).



Lecture

Bill Gates

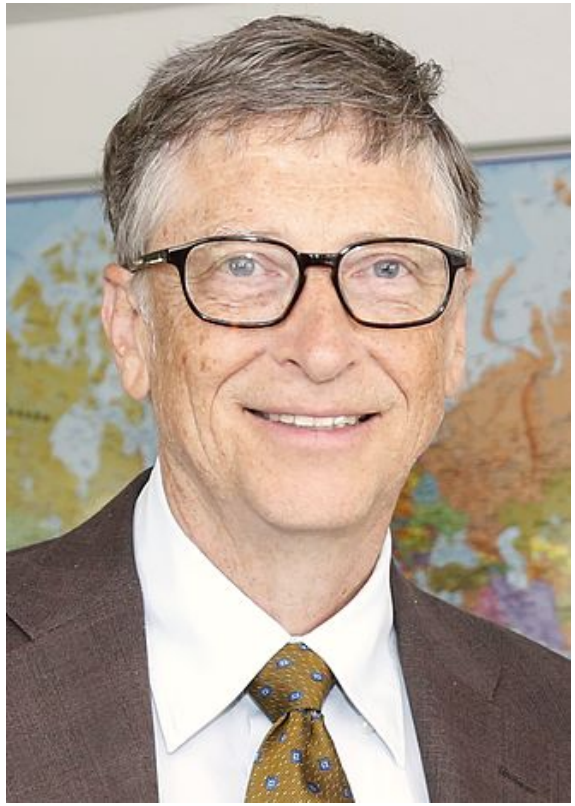


Figure 1: Sir William Gates was one of the co-founders of Microsoft along with Paul Allen. Gates is now most actively known for his work in the Bill and Melinda Gates Foundations Philanthropic work.

Inheritance

Reviewing our abstractions for Code Reuse

Our toolbox in this class

- Loops
- Functions
- Templates
- classes and structs
- And a new one–**inheritance**

Inheritance Definition

What does it mean to inherit something?

- Biology and Genetics
- A gift or donation
- C++ Object Oriented Programming technique with classes and structs – bingo!

Inheritance Definition in C++

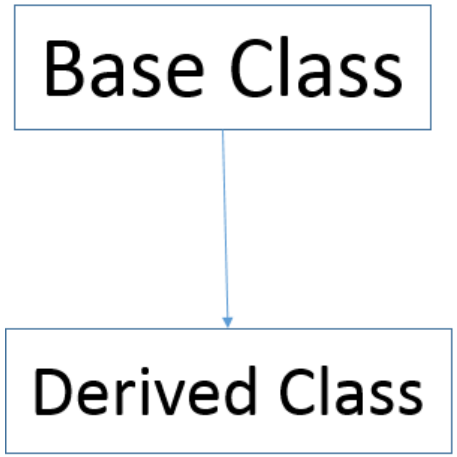
Inheritance

The ability to define a class in terms of another class such that the data and behaviors may be passed from one class to another.

- So to do this, you have a base class (parent) and a derived class (child) that inherits the member variables and member functions from the base(parent) class.

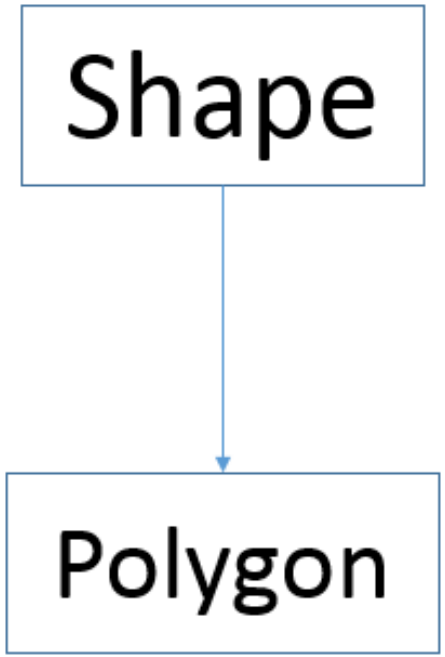
Inheritance - Base and Derived Class

- Some base class here defines some functionality.
- The derived class is a specialization of the base class.
- That is, it can do many of the things the base class can, or perhaps in a different manner, and perhaps even more!



Inheritance - shape

- As an example, we may have a shape class
- A polygon is a n-sided shape.



Shape Code Example

```
1  struct Color{
2    int r,g,b;
3  };
4  class Shape{
5  public:
6    Color c;
7
8  };
9
10 class Polygon : public Shape{
11 public:
12   int numberOfSides;
13     // Also has access to "Color c" because we inherit
14
15 };
```

Listing 1: Polygon inherits from Shape

Shape Code Example - Using inherited members

```
1  class Polygon : public Shape{
2  public:
3    int numberOfSides;
4      // Also has access to "Color c" because we inherit
5      // And we can set all of the public members as such.
6    Polygon(){
7          c.r = 255;
8          c.g = 255;
9          c.b = 255;
10     }
11 };
```

Listing 2: Polygon has access to Shapes attributes

What is new

class Polygon : public Shape

- So we have our regular class declaration
- Follow it with a colon
- Specify the access level (e.g. public)
- Then the class we want to inherit from.

Access Levels for a class

- We have mentioned this topic before with classes.
- **public:** - Members are available for use inside and outside of class.
- **private:** - Members are unavailable for use outside of class.

(Ordered bottom to top with how restrictive each access level is)

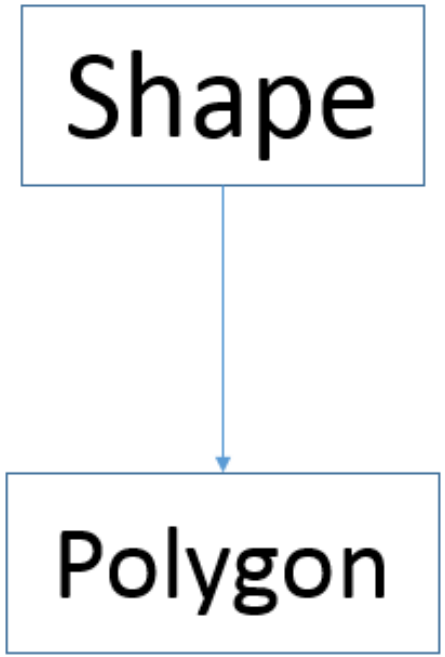
Access Levels for inheritance

- We have mentioned this topic before with classes.
- **public:** - Members are available for use inside and outside derived classes.
- **protected:** (new one) Members are available for use in derived classes, but not outside.
- **private:** - Members are not inherited.

(Ordered bottom to top with how restrictive each access level is)

Key phrase in previous example

- As an example, we may have a shape class
- A polygon **is a** n-sided shape.



is-a relationship

- The isa relationship means that a Polygon **is a** shape.
- This is not to be confused with composition.
- Composition defined as a **has-a** relationship.

Inheritance is not composition

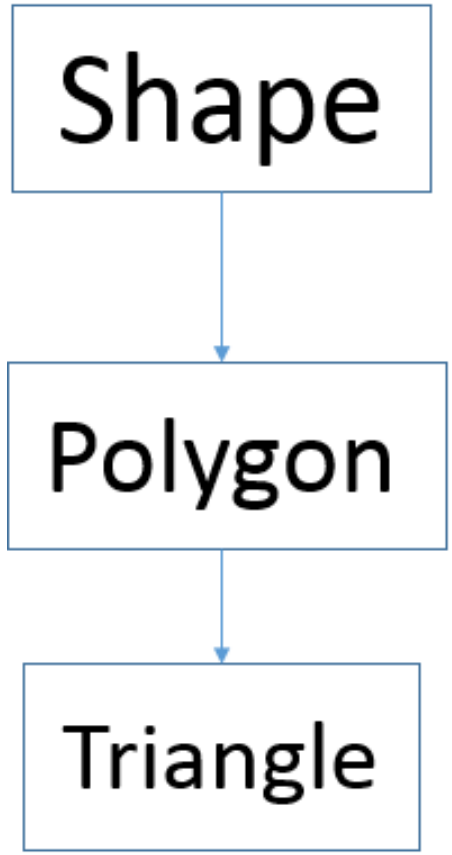
- Inheritance is not composition.
- That is, a class that contains another class as a member object is not inheritance.
- You will see this in literature referred to a "has a" relationship.
- Code is less strongly coupled together
- e.g. class StudentDB "has a" member variable Student.

```
1 struct Student{
2   // member variables and functions
3 };
4
5 // A StudentDB has a Student as a member varible
6 class StudentDB{
7 public:
8   Student* students;
9 };
```

Listing 3: Has a relationship

Inheritance Levels

- With inheritance, we are not limited to how many times we inherit one class from another.
- In fact we can continuously derive from as many classes as our compiler will let us.
- Although, it may be recommended to only inherit a few times so we can comprehend what is going on.



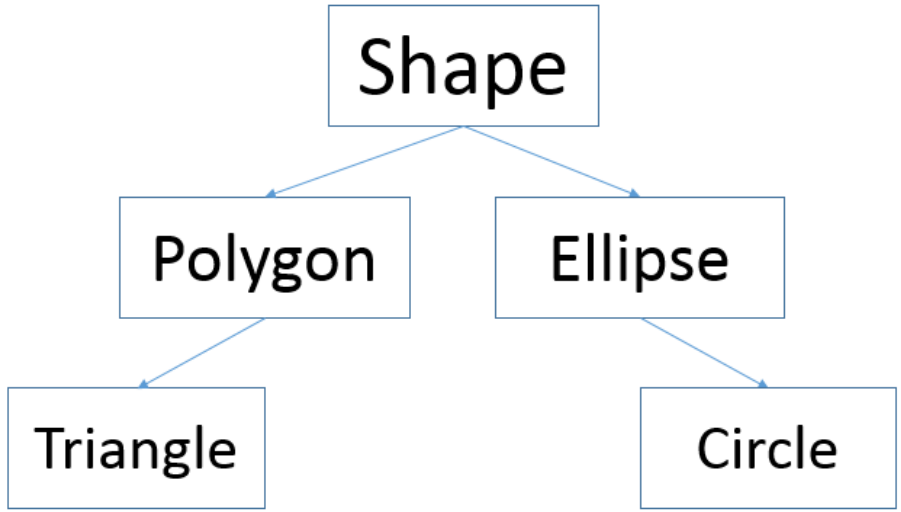
Shape-Polygon-Triangle

```
1  class Shape{
2  public:
3    Color c;
4  };
5  class Polygon : public Shape{
6  public:
7    int numberOfSides;
8      // Also has access to "Color c" because we inherit
9  };
10
11 class Triangle : public Polygon{
12   float base,height;
13 public:
14     // Has access to "Color c" because we inherit
15     // Has access to "int numberOfSides" because we inherit
16   int area(){
17       return base * height * 0.5;
18     }
19 };
```

Listing 4: Three levels of inheritance

Diverging Inheritance

- Inheritance can also span multiple paths.
- You do not have to derive from the same linear path.



Functions - Round 3?

virtual functions

- We can qualify member functions in a class with **virtual**
- What this means, is that a derived class *may* override or redefine the behavior of the member function it is inheriting from.
- We can push a step further, with **pure virtual functions**.
- With a pure virtual function, the derived class *must* override or redefine the behavior of the member function it is inheriting from.

Concrete and Abstract Classes

- With pure virtual functions, why must we override functionality?
- The idea is we can create an abstract class. This serves as an interface that you want to enforce.
- Thus, you can derive from this abstract class (which is your base class), and ensure programmers implement member functions.

Example virtual functions

```
1  class Polygon{
2  public:
3    virtual void area(){
4        std::cout << "Calculates area\n";
5      }
6  };
7
8  class Triangle : public Polygon{
9    float base,height;
10 public:
11     // -- Overrides area -- Now Triangle has redefined
       behavior
12   int area(){
13       return base * height * 0.5;
14     }
15 };
```

Listing 5: Overriding functionality

Example pure virtual function

```
1  class Polygon{
2  public:
3    // Pure virtual function -- must be overridden!
4    virtual void area() = 0;
5  };
6
7  class Triangle : public Polygon{
8    float base,height;
9  public:
10     // Overrides area
11   int area(){
12       return base * height * 0.5;
13     }
14 };
```

Listing 6: Overriding functionality

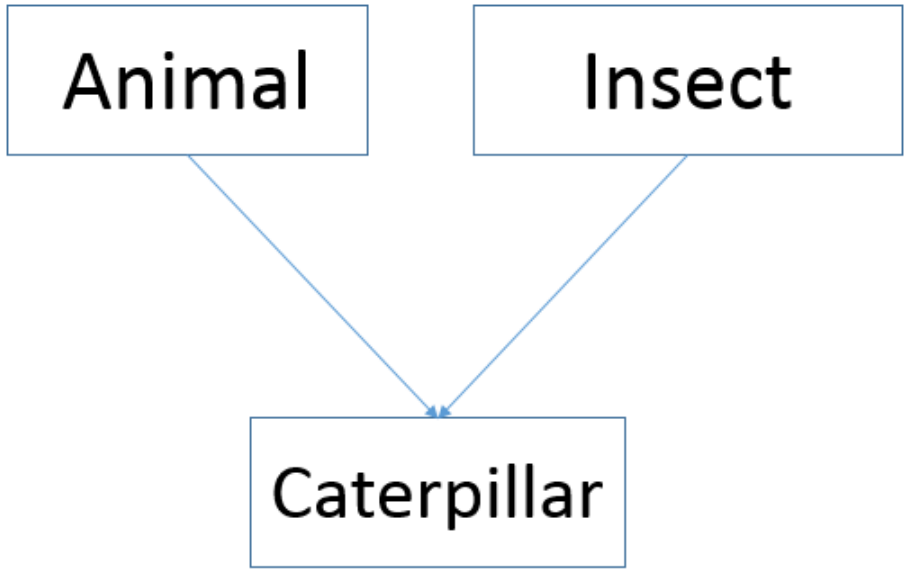
More with Functions

- Friend Functions are external to a class.
- That is, a friend function can access a class's data, but it exists external to the class.
- Be aware they exist, and that they may be useful.

Advanced Inheritance

Advanced Inheritance

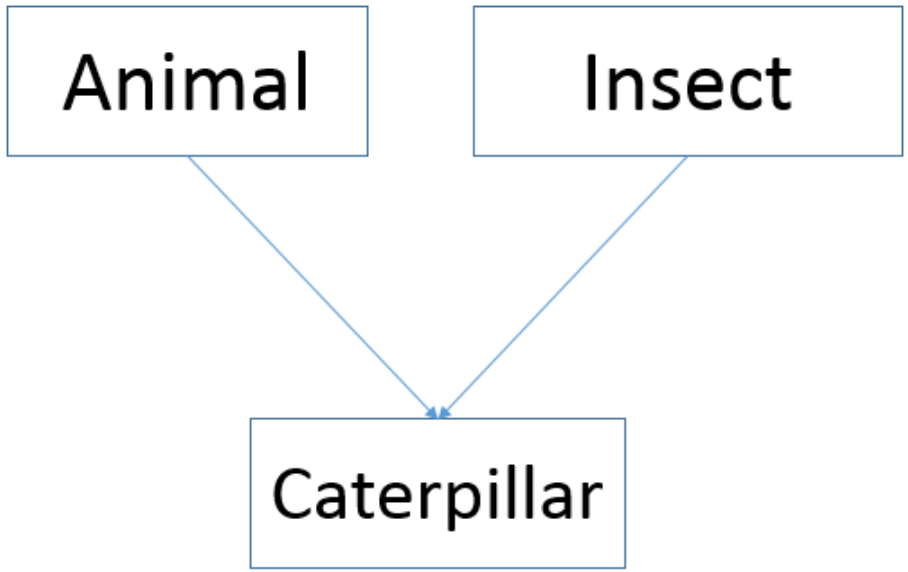
- Multiple Inheritance is the idea that you can have a derived class that has multiple base classes.
- This means the derived class gets attributes from both base classes.



Syntax: class Caterpillar : public Animal, public Insect { ... };

Advanced Inheritance - Be Careful

- So what if Animal and Insect have the same method? Which one does Caterpillar get?
- Who knows is the answer? The compiler may guess or give an error or do something else. This is generally not good!



Inheritence

Pros

- Gives power to the designer (abstract classes, concrete)
- Changes can propagate, and only need to be maintained in one spot (Code is strongly coupled).
- Natural to think about, and gives objects logical relationships

Cons

- Code can be difficult to reason about
- Updating code means understanding the designers original intent, or otherwise reading comments.

In-Class Activity

http://www.mshah.io/comp/11/activities/activity12/activity.pdf

Activity Discussion

Review of what we learned

- (At least) Two students
- Tell me each 1 thing you learned or found interesting in lecture.

5-10 minute break

To the lab!

Lab: `http://www.mshah.io/comp/11/labs/lab12/lab.pdf`
[1]

[1] You should have gotten an e-mail and hopefully setup an account at `https://www.eecs.tufts.edu/~accounts` prior to today. If not–no worries, we'll take care of it during lab!

Glossary

Inheritance The ability to define a class in terms of another class such that the data and behaviors may be passed from one class to another. 10