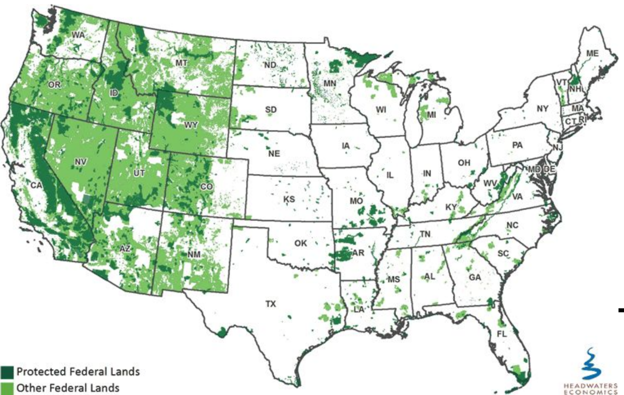
Inheritance

Private vs Protected

- All variables have been listed as *private* so far
  - Only available to that class ONLY
  - Great for individual classes...

Private vs Protected

- Protected
  - Allows for situations where variables can be used by other classes
  - Is much more common
    - Write “protected: “ instead of “private: ” in a .h file



Inheritance

- Classes can *inherit* methods and variables from other classes
  - Allows for relationships between various classes without useless copy & pasting

Inheritance

- Base class
  - Describes the class where methods & variables originate from

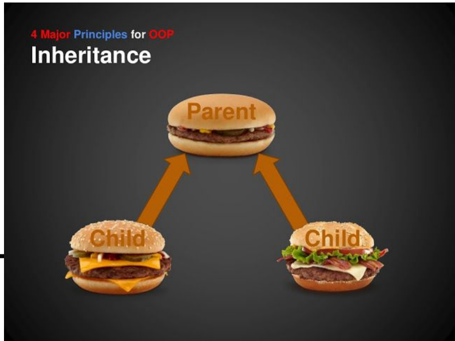
Inheritance

- Derived class(es)
  - Describes the class(es) that inherit methods & variables from a base class
  - Can have unique methods and variables as well

Inheritance Notation

- Written in C++ as:

```
class DerivedClass :: public BaseClass {
   //New methods and variables
}
```

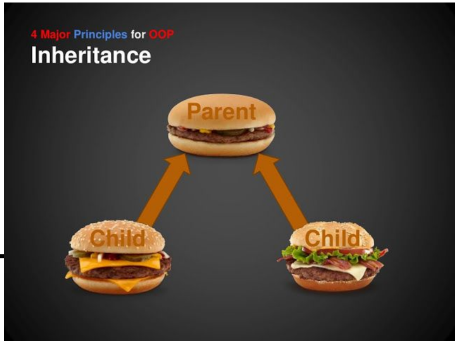


Inheritance Notation

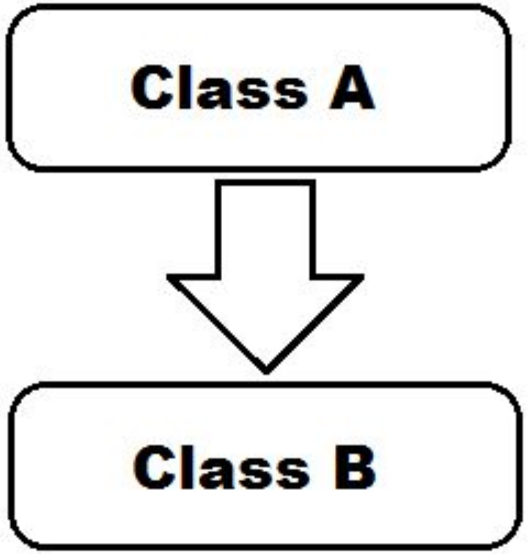
- Written in C++ as:

```
class DerivedClass :: public BaseClass {
   //New methods and variables
}
```

- Can also be *protected or private*
  - *Public* is the most common

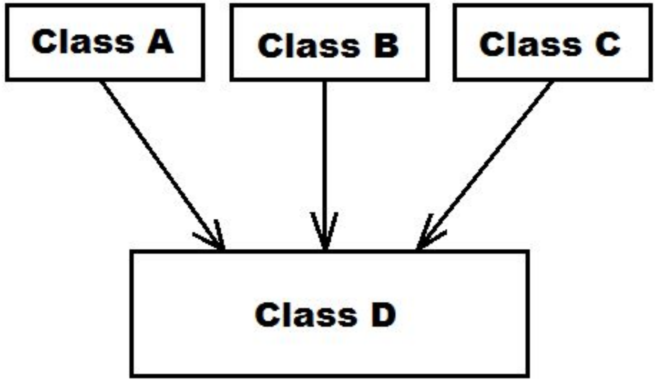


Single Inheritance



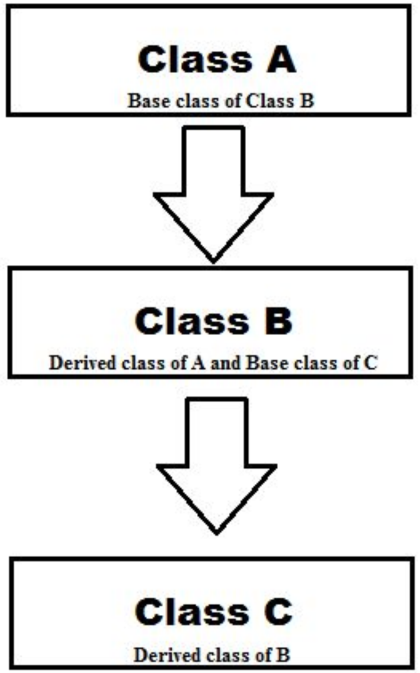
- One base class & one inherited class
  - Example: Person = base class; student = inherited class

Multiple Inheritance



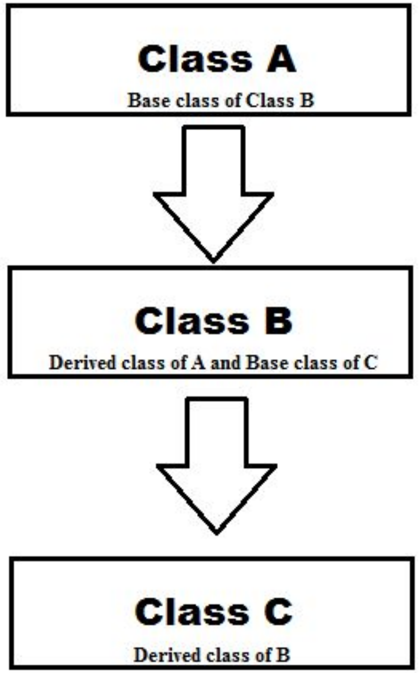
- One inherited class, multiple base classes
  - Example: Animal, Mammal, FlyingAnimal = base classes; bat = inherited class

Multilevel Inheritance



- Class can be both base and inherited classes
  - Example: Person → Student → Curley_Student

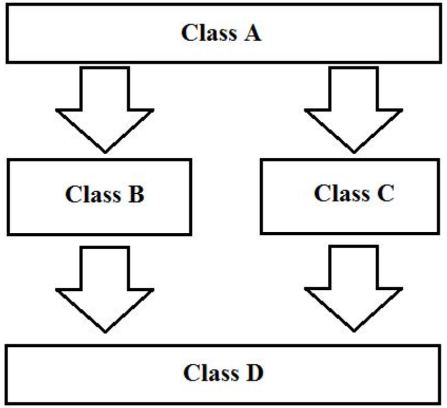
Multilevel Inheritance



- Class can be both base and inherited classes
  - Example: Person → Student → Curley_Student
  - This can be combined with other types of inheritance
  - Can extend an arbitray number of levels

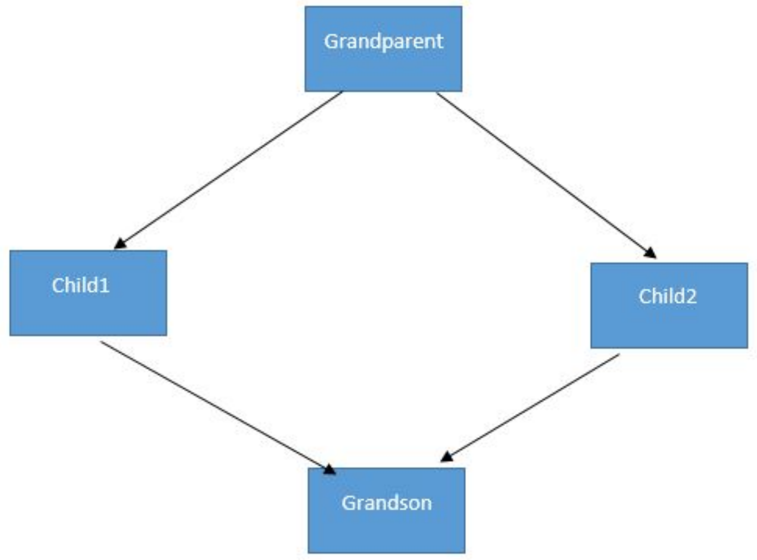
Multilevel Inheritance

- The base and inherited classes resolve into a single inherited class
  - Splits into a diamond
  - What kind of problems can this cause?



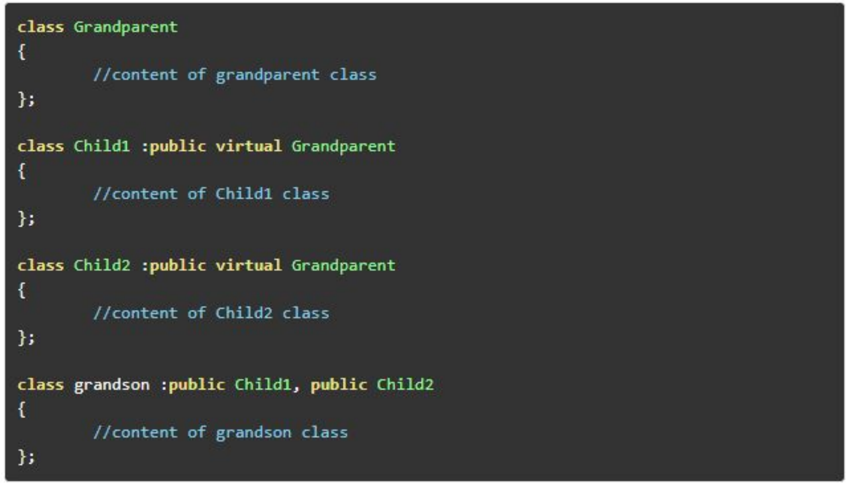
Multilevel Inheritance - Diamond Problem

- The Grandson class would have duplicate methods from the Grandparent



Multilevel Inheritance - Diamond Problem

- Solution = *virtual inheritance*
  - Each child would have “virtual inheritance” from the Grandparent



```
class Grandparent
{
        //content of grandparent class
};

class Child1 :public virtual Grandparent
{
        //content of Child1 class
};

class Child2 :public virtual Grandparent
{
        //content of Child2 class
};

class grandson :public Child1, public Child2
{
        //content of grandson class
};
```

Constructors

- Each class has a unique constructor
  - Each constructor is called in a specific order
  - https://www.tutorialcup.com/cplusplus/inheritance.htm

Constructor Coding Challenge

- https://www.hackerrank.com/challenges/java-inheritance-1/problem
  - Model this example in C++
  - *Summary*: Create an *animal* class & a *bird* class. The *animal* class should print “I am walking”, and the *bird* class should extend *animal* and print “I am flying” and “I am singing”