Class inheritance, polymorphism

Inheritance

- a dog is an animal
- a rose is a plant
- a rectangle is a shape
- a bus is a vehicle
  - a car is also a vehicle
- a <special> is a <general>, plus additional properties

Inheritance

```
class vehicle{
  int maxspeed;
  char* maker;
  void setMaker(char*)
};

class car : public vehicle{
  int rating;
  int mpg;
  int ndoors;
  void setRating(double)
};

class bus : public vehicle{
  int tank_capacity;
  int nseats;
};
```

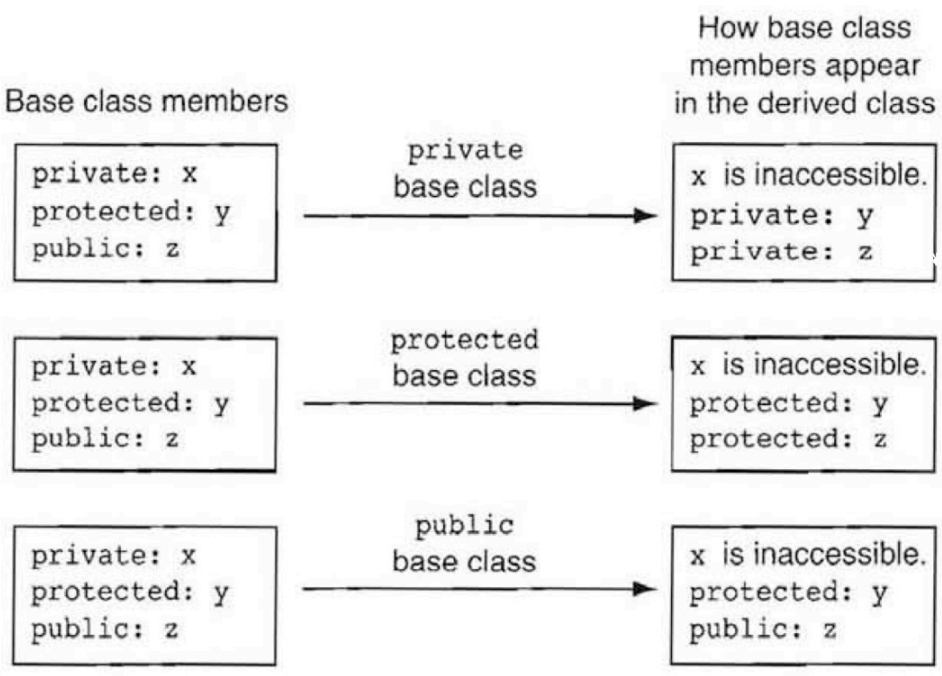
- class car has now:
  - maxspeed
  - maker
  - rating
  - mpg
  - ndoors
  - setmaker()
  - setrating()

- a class X that inherits class Z has all its members, methods, etc;
  - and additional members, methods
- many classes X, Y can inherit the same class Z
- Z=base class
- X,Y = derived classes
- derived classes can be inherited Z->X->T

protected members

- private members <base> = inaccessible to <derived> code
  - they can still be accessed through the <base> functions
- protected = same as private, but <derived> methods can use them
- the inheritance also has an "access mode"
  - `class car : public vehicle...`

access



constructor call

- the <derived> constructor can call the <base> constructor
- default <base> constructor always called first
- car(char* maker, double rating, int mpg, int ndoors) : vehicle(maker)

Multiple Inheritance

- a class can inherit 2 other classes
- class goods{
  - double price;
  - int salary;
  - char* seller;
- }
- class car: public vehicle, public goods{...
- now class car has all the members of vehicle, and all members of class goods

Polymorphism

- declare an outside function with parameter <base> variable
  - `parameter is passed by reference`
  - `int myfunction (vehicle &a){ ... }`
- call the function on a <derived> variable
  - `car x;`
  - do something with x…
  - `myfunction(x);`
- it works because <derived> variables are also <base> variables