Overview of Inheritance

- Means of specifying **hierarchical relationships** between types (i.e., classes)
- **Subclass** (or **derived class**) inherits stuff from **superclass** (or **base class**)
- Subclass can inherit member variables and functions from superclass; <u>not</u> friend functions
- **"Is a" relationship**: object of subclass also "is a" object of superclass
- <u>Examples</u>: manager is an employee, truck is a vehicle, dog is a pet
- Can have **multiple inheritance** in C++ (later…)

Example

```
// PET is superclass (also called base class)
class PET {
protected:
  CString name;

public:
  PET() { cout << "\nPET constructor #1 called.\n"; name = ""; }

  PET(const CString n) : name (n)  {
      cout << "\nPET constructor #2 called for " << name << endl;}

  ~PET() { cout << "\nPET destructor called for " << name << endl; }

  void setName(const CString s) { name = s; }
  CString getName() const { return(name); }

  void print() const { cout << "\nName: " << name << endl; }

  friend istream& operator >> (istream& ins, PET& p) {
      char charArray[21];
      cout << "\nEnter pet name: ";
      ins.getline(charArray, 21);
       p.name = charArray;
      return(ins);
  }

  friend ostream& operator << (ostream& outs, const PET& p) {
      outs << "\nName: " << p.name << endl;
      return(outs);
  }

  bool operator == (const PET& p) const {
     return(name == p.name);
  }
};
```

```
// DOG is subclass of PET (also called derived class)
class DOG : public PET {
private:
  CString breed;

public:
  DOG( ) { cout << "\nDOG constructor #1 called.\n"; breed = "Mixed"; }

  DOG(const CString b, const CString n) : PET(n), breed(b) {
    cout << "\nDOG constructor #2 called for " << breed << endl; }

  ~DOG() { cout << "\nDOG destructor called for " << breed << endl; }

  CString getBreed() const { return(breed); }
   void setBreed(const CString b) { breed = b; }

   void print() const {
       PET::print();
       cout << "Breed: " << breed << endl;
   }

  friend istream& operator >> (istream& ins, DOG& d) {
      char charArray[21];
      ins >> ((PET) d);
       cout << "Enter breed: ";
       ins.getline(charArray, 20);
       d.breed = charArray;
       return(ins);
  }

  friend ostream& operator << (ostream& outs, const DOG& d) {
       outs << ((PET) d);
       outs << "Breed: " << d.breed << endl;
      return(outs);
  }

  bool operator == (const DOG& d) const {
     bool samePet = PET::operator ==((PET) d);
     return(samePet && (breed == d.breed));
  }
};
```

Ouput that is produced is shown in **blue**

```
int main() {
DOG d;                                         Calls DOG( )
                                                   Calls PET( )  PET constructor #1 called.
                                                   DOG constructor #1 called.

cout << "\nJust did declaration of DOG d in main( ).\n";   Just did declaration of DOG d in main( ).

d.setBreed("Great Dane");
d.setName("Scooby-Doo");

d.print( );                                    Calls DOG::print
                                                   Calls PET::print    Name: Scooby-Doo
                                                   Breed: Great Dane

((PET) d).print( );                            Constructs temporary PET
                                               Calls PET::print    Name: Scooby-Doo
                                               Destroys temp PET    PET destructor called for Scooby-Doo

d.PET::print( );                               Calls PET::print    Name: Scooby-Doo

cin >> d;                                      Calls DOG::>>
                                                   Constructs temporary PET
                                                   Calls PET::>>    Enter pet name: Snoopy
                                                   Destroys temp PET   PET destructor called for Snoopy
                                                   Enter breed: Beagle

cout << "\nd is now: "                         d is now:
     << d << endl;                             Calls DOG::<<
                                                   Constructs temporary PET
                                                   Calls PET::<<    Name: Scooby-Doo
                                                   Destroys temp PET  PET destructor called for Scooby-Doo
                                                   Breed: Beagle

if (d == DOG("Collie", "Lassie"))              Calls DOG("Collie", "Lassie")
                                                   Calls PET("Lassie")  PET constructor #2 called for Lassie
                                                   DOG constructor #2 called for Collie

                                               Calls DOG::==
                                                   Constructs temporary PET
                                                   Calls PET::==
                                                   Destroys temp PET    PET destructor called for Lassie

                                               Calls ~DOG( )
                                                   DOG destructor called for Collie
                                                   Calls ~PET    PET destructor called for Lassie

  cout << "\nThat is Lassie\n";
else cout << "\nThat's not Lassie\n";          That's not Lassie

return(0);                                     Calls ~DOG( )
                                                   DOG destructor called for Beagle
                                                   Calls ~PET    PET destructor called for Scooby-Doo
```