11/23/2014

CS 142
Inheritance/Polymorphism

Announcements

• Program 8 has been assigned - due Tuesday, Dec. 9th by 11:55pm

Object Composition

• A class can use another class as a member variable (a field).
• This called object composition.
• Use this when you would say "An object of class A has an object of class B.”
  – A dog has an owner.
  – A car has an engine.
  – A student has an advisor.
  – A line segment has a starting point and an ending point.
• Known as a "has-a" relationship.

```cpp
class person {
  // things here
};

class dog {
  public:
    ...
    private:
      person owner;
};
```
class point {
    // things here
};

class line {
    public:
        ...
    private:
        point start, end;
};

class line {
    public:
        double getLength() { 
            return sqrt(
                pow(start.getX() - end.getX(), 2) + 
                pow(start.getY() - end.getY(), 2));
        } 
    private:
        point start, end;
};

Inheritance

• A different kind of relationship is an "is-a" relationship.
• Use this relationship to express when a class is a specific kind of another class.
  – A poodle is a specific kind of dog.
  – A racecar is a specific kind of car.
• This concept is called inheritance

Inheritance (is-a) versus composition (has-a)

• Inheritance expresses that one class can do everything another class can do, plus more:
  – A racecar is just a car that can also drive extra fast around a race track.
• Composition expresses that one class is a component of another class:
  – An engine is a piece of a car.
Derived Class Inherits from Base Class

- Inside the class, the derived class has access to all the public and protected members of the base class.
- Inside the class, the derived class cannot access private members.
- Outside the class, the derived class has all the same public members as the base class has.
  — except constructors

Constructors with inheritance

- Constructors (even if public) are not automatically inherited by derived classes.
- Derived classes must create their own constructors if you want them.

```cpp
class dog {
    public:
        dog(string s);
    private:
        string name;
};
class showdog : public dog {
};

main:
    dog mydog("Fido");
    showdog otherdog("Herbert");
}
```

```cpp
class dog {
    public:
        dog(string s);
    private:
        string name;
};
class showdog : public dog {
    public:
        showdog(string s);
};

main:
    dog mydog("Fido");
    showdog otherdog("Herbert");
```
Constructors with Inheritance

- All classes must have at least one constructor.
  - If you don’t write at least one, a default one (with no args) is generated behind the scenes for you.
- Every time an object of a class is constructed, a constructor must be called.
  - Default (no arg) constructor is used unless otherwise specified.

Constructors with Inheritance

- When you construct an object of a derived class:
  - The derived class constructor is called
    - default constructor if not otherwise specified
  - Before running its own code, the derived class constructor must call a base class constructor.
    - default constructor if not otherwise specified
  - Once the base class constructor code runs, the derived class constructor code runs.

```
class Derived : class Base {
    
    Derived::Derived(...) : Base(...) 
    {
        // normal things here
    }
```
Overriding Methods

- A derived class is allowed to "rewrite" methods in a base class.
  - Very common; done to alter the way a derived class behaves.
- This is called **overriding**.
- Overriding a method in a derived class "hides" the base class method code and replaces it with your new code.

Example Code

- See public directory