Inheritance II

Is-a versus has-a

- When an object of class A ***has a***n object of class B, use ***object composition***.
  - Class A will have a field (variable) of class B in its implementation.
- When class A ***is a*** specific kind of another class B, use ***inheritance***.
  - Class A will inherit from class B.

Is-a or has-a?

- Class = ***Animal***
  - Heart
  - Porcupine
  - Duck
- Class = ***Phone***
  - Cell Phone
  - Ringtone
  - Text Message
  - Landline

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.

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

class showdog : public dog {

};
```

```
main:

dog mydog("Fido");
showdog otherdog("Herbert");
```

```
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.

Constructors with inheritance

- Derived class constructors are allowed to ***explicitly*** call base class constructors.
- Commonly used to initialize private variables that derived classes do not have access to.

```
class Derived : class Base {
};

Derived::Derived(…)
: Base(…)
{
  // normal things here
}
```

Put a colon after the derived class constructor line, and explicitly call the Base constructor that you want.

Only time in C++ when you are allowed to explicitly call a constructor.

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.

- Add two new car types to the race by defining two new classes that inherit from car:
- A `racecar`:
  - can accelerate at 10 mph every second, rather than 5 mph every second
  - all race cars have a top speed of 200 mph.
- A `clunker`:
  - still accelerates at 5 mph per second.
  - top speed of 50 mph.
  - But after calling drive() 3 times, the car dies, immediately stops, can't be fixed, and you have to call your parents to pick you up.