Object-Oriented Programming

C++ Inheritance Specifiers

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

Classes

```
class Base {
public:
    virtual void operator1() const;
    virtual void operator2(int n);
};

class Derived : public Base {
public:
    virtual void operator1();
    virtual void operator2(char n);
};
```

Client

```
int main() {

    Base b;
    b.operator1();
    Derived d;
    d.operator1();

    Base* pb = &b;
    pb->operator1();
    pb = &d;
    pb->operator1();

    char c;
    b.operator2(c);
    d.operator2(c);

    pb = &b;
    pb->operator2(c);
    pb = &d;
    pb->operator2(c);

    int n;
    b.operator2(n);
    d.operator2(n);

    pb = &b;
    pb->operator2(n);
    pb = &d;
    pb->operator2(n);

    return 0;
}
```

```cpp
class Base {
public:
    virtual void operator1() const;
    virtual void operator2(int n);
};

class Derived : public Base {
public:
    virtual void operator1();
    virtual void operator2(char n);
};
```

Output

```cpp
int main() {

    Base b;
    b.operator1();
    Derived d;
    d.operator1();

    Base* pb = &b;
    pb->operator1();
    pb = &d;
    pb->operator1();

    char c;
    b.operator2(c);
    d.operator2(c);

    pb = &b;
    pb->operator2(c);
    pb = &d;
    pb->operator2(c);

    int n;
    b.operator2(n);
    d.operator2(n);

    pb = &b;
    pb->operator2(n);
    pb = &d;
    pb->operator2(n);

    return 0;
}
```

```
Base::operator1();

Derived::operator1();

Base::operator1();

Base::operator1();

Base::operator2(int n);
Derived::operator2(char n);

Base::operator2(int n);

Base::operator2(int n);

Base::operator2(int n);
Derived::operator2(char n);

Base::operator2(int n);

Base::operator2(int n);
```

Problems

```
class Base {
public:
    virtual void operator1() const;
    virtual void operator2(int n);
};

class Derived : public Base {
public:
    virtual void operator1();
    virtual void operator2(char n);
};
```

- A non-const does not override a const
- Name of a method ***hides*** (not *overrides*)

(Partial) Solution: `override`

```cpp
class Base {
public:
    virtual void operator1() const;
    virtual void operator2(int n);
};

class Derived : public Base {
public:
    void operator1() override;
    void operator2(char n) override;
};
```

```
override.cpp:13:18: error: 'operator1' marked 'override' but does not override any member functions
    virtual void operator1() override { std::cerr << "Derived::operator1()\n"; }
                 ^
override.cpp:14:18: error: 'operator2' marked 'override' but does not override any member functions
    virtual void operator2(char n) override { std::cerr << "Derived::operator2(char n)\n"; }
                 ^
```

`override` Notes

```
class Base {
public:
    virtual void operator1() const;
    virtual void operator2(int n);
};

class Derived : public Base {
public:
    void operator1() override;
    void operator2(char n) override;
};
```

- `override` is a *specifier*
- Acts as a *contextual keyword*, and you are still allowed to use `override` as an identifier, e.g., `int override = 0;`

  C# Contextual Keywords

  C++ Keywords
- Style Note: Can replace the declaration of `virtual` in derived classes
- General Note: Instead of *guessing*, *experiment*

Classes

```cpp
class Base {
public:
    virtual void operation1();
};

class DerivedA : public Base {
public:
    void operation1() final;
    void operation2() final;
};

class DerivedB final : public DerivedA {
public:
    void operation1() override;
};

class DerivedC final : public DerivedB {
public:
    void operation1() override;
};
```

```
final.cpp:9:23: error: only virtual member functions can be marked 'final'
    void operation2() final;
                      ^~~~~
final.cpp:14:10: error: declaration of 'operation1' overrides a 'final' function
    void operation1() override;
         ^
final.cpp:8:10: note: overridden virtual function is here
    void operation1() final;
         ^
final.cpp:17:31: error: base 'DerivedB' is marked 'final'
class DerivedC final : public DerivedB {
                              ^
final.cpp:12:7: note: 'DerivedB' declared here
class DerivedB final : public DerivedA {
      ^
final.cpp:19:23: error: only virtual member functions can be marked 'override'
    void operation1() override;
                      ^~~~~~~~
4 errors generated.
```

`final` Notes

```
 1 class Base {
 2 public:
 3     virtual void operation1();
 4 };
 5
 6 class DerivedA : public Base {
 7 public:
 8     void operation1() final;
 9     void operation2() final;
10 };
11
12 class DerivedB final : public DerivedA {
13 public:
14     void operation1() override;
15 };
16
17 class DerivedC final : public DerivedB {
18 public:
19     void operation1() override;
20 };
```

- On a class, enforces that we cannot derive from the class
- On a method, enforces that we cannot override the method