C++ Inheritance

grab handout!

Quick advertisement!

- **We're looking to hire additional CS 126 moderators for next semester!**
  - We pay!
  - Looks good on a resume!
  - It is a great way to get to know faculty!
    - Good for future letters of recommendation / references
  - Get to know cool members of course staff!

Destructors, cont.

- **Syntax: ~Type();**

- **Called under two circumstances:** implicit
  - When you delete a heap-allocated object
  - When a stack allocated object goes out of scope

- **A default is provided by the compiler (that does nothing)**

Automatic functions

- **Created for you by the compiler**
  - Default (0-argument) constructor
  - Copy constructor
  - Assignment operator
  - Destructor

rule of three

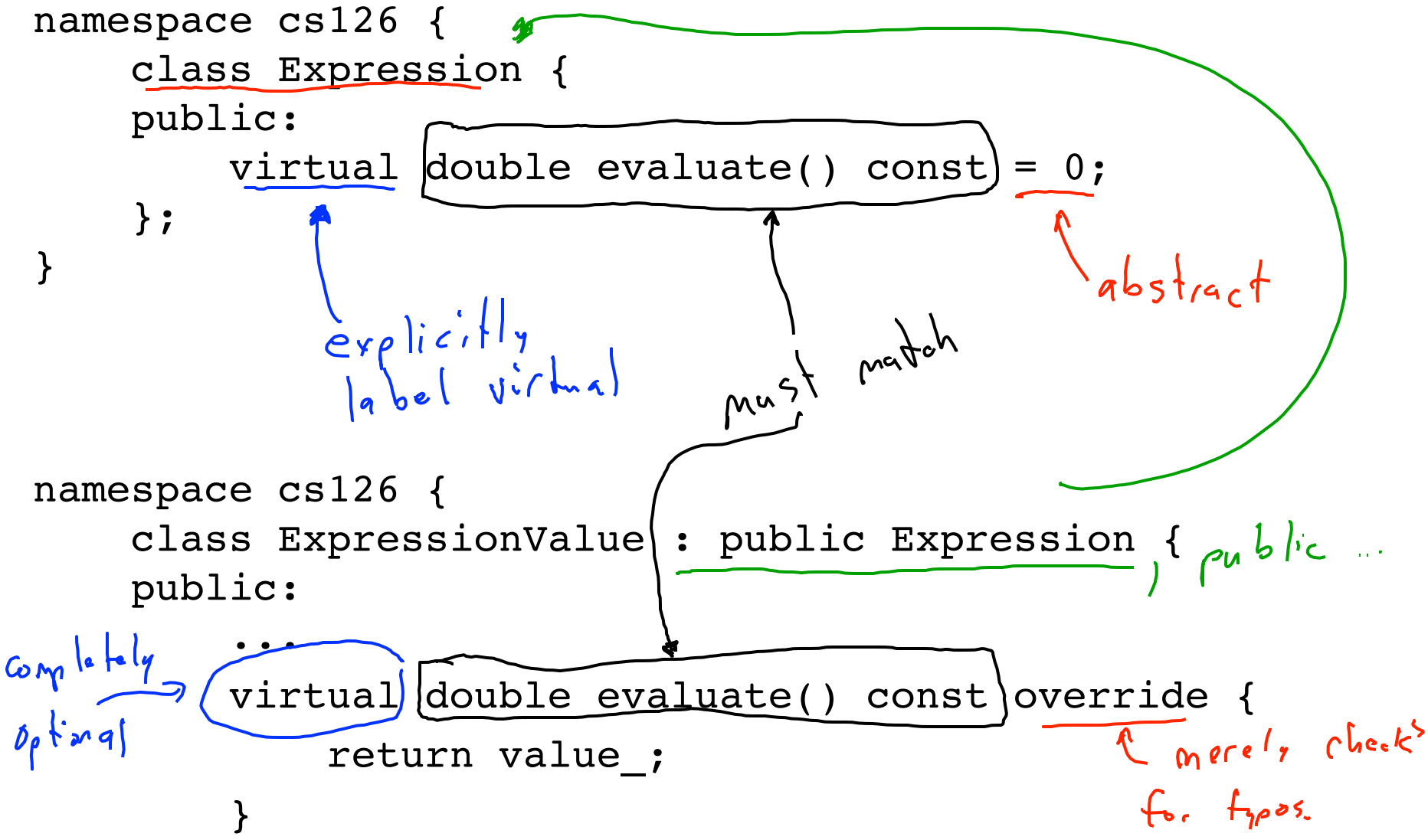
- **You can over-ride the definition of any of these functions**

The problem with Expression code, so far

Adding an abstract base class

```
namespace cs126 {
    class Expression {
    public:
        virtual double evaluate() const = 0;
    };
}
```

```
namespace cs126 {
    class ExpressionValue : public Expression {
    public:
        ...
        virtual double evaluate() const override {
            return value_;
        }
```



Different possibilities for base class code

```
virtual double evaluate() const = 0;
```

```
double evaluate() const = 0;
```

```
virtual double evaluate() const { return 4; }
```

```
double evaluate() const { return 4; }
```

```
Expression *expr = new Expression();
cout << expr->evaluate() << endl;
```

A) doesn't compile
B) 4
C) 11

Different possibilities for base class code

1
```
virtual double evaluate() const = 0;
```

```
double evaluate() const = 0;
```

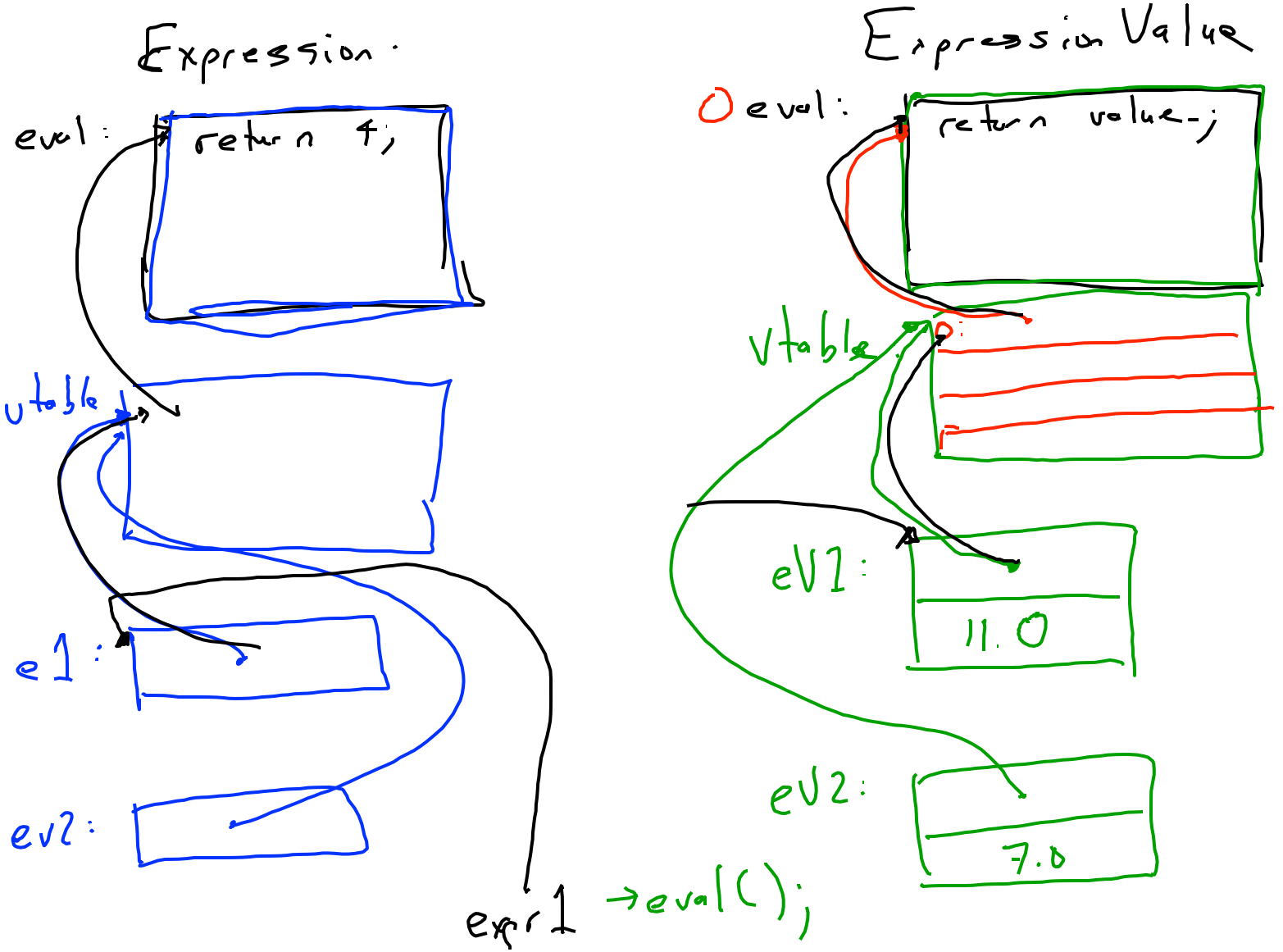
3
```
virtual double evaluate() const { return 4; }
```

4
```
double evaluate() const { return 4; }
```

```
Expression *expr = new ExpressionValue(11.0);
cout << expr->evaluate() << endl;
```

A) doesn't compile
B) 4
C) 11

How does it know which version to run?



Handling more complex objects

- **With pointers/references to other objects**
  - Copy constructor
  - Assignment operator
  - destructor

clone() design pattern

- **A function we can call on an object to reproduce itself**
  - Using virtual functions we don't need to know what it is

```
Expression *clone() const;
```