OPERATORS AS `friend` FUNCTIONS

Definition and Calling Syntaxes

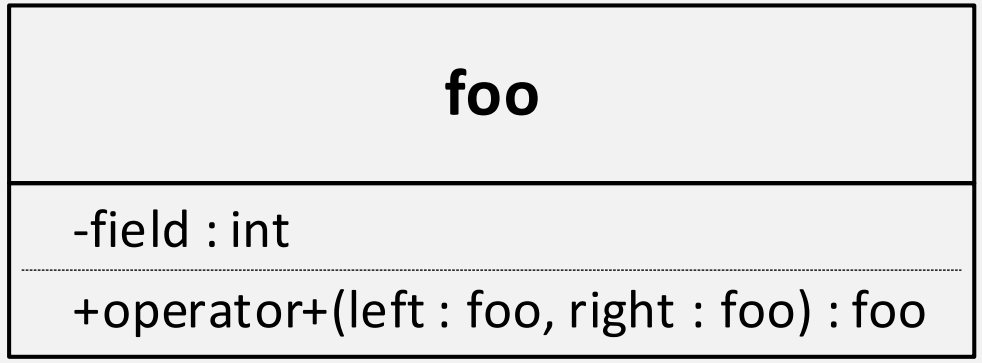
Delroy A. Brinkerhoff

`friend` FUNCTION REVIEW

- `friend` functions are not members of the befriending class
- Nevertheless, `friend` functions have access to the private features of a class
- `friend` functions must be declared as friends in the class specification with the "friend" keyword
- Overloaded operators are often implemented as `friend` functions
- There is no UML notation to indicate a `friend` function

CLASS WITH AN OVERLOADED `friend` OPERATOR

UML



C++

```
class foo
{
    private:
        int field;
    public:
        friend foo operator+(foo left, foo right);
};
```

FUNCTION DEFINITION

- Function is not a member, so
  - The class name and the scope resolution operator are not used
  - All operands are passed as explicit arguments inside the parentheses
  - All fields must be accessed using argument names

```
foo operator+(foo left, foo right)
{
    return foo(left.field + right.field);
}
```

A + B

CALLING SYNTAXES

USED IN PRACTICE

- `foo a;`
- `foo b;`
- `foo c;`

- `c = a + b;`

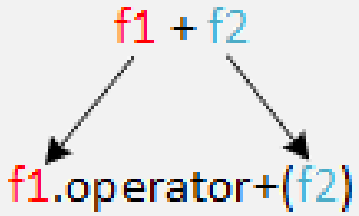
USED FOR ILLUSTRATION

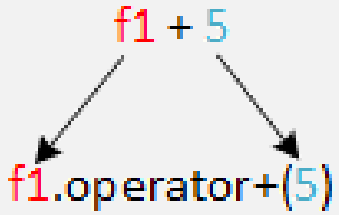
- `foo a;`
- `foo b;`
- `foo c;`

- `c = operator+(a, b);`

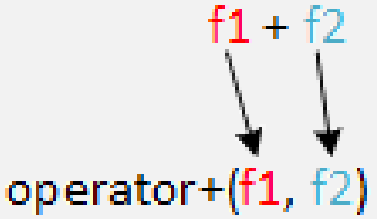
THE RELATIONSHIP BETWEEN OPERANDS AND ARGUMENTS

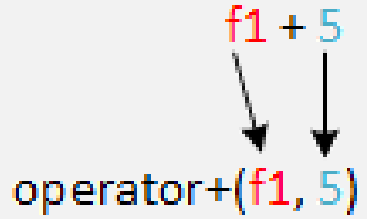
MEMBER FUNCTION





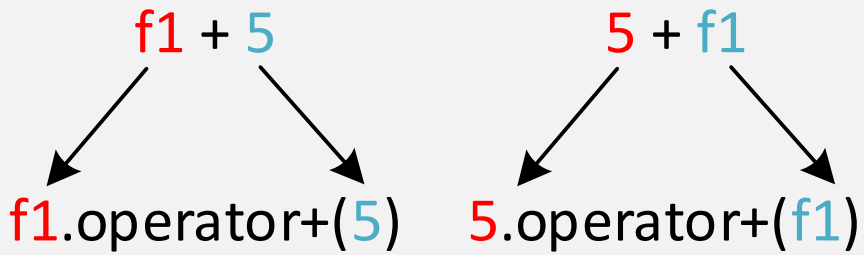
friend FUNCTION





WHY `friend` FUNCTIONS? THE PROBLEM WITH MEMBER OPERANDS

- Addition is commutative:
  - `f1 + f2 ≡ f2 + f1`
  - `f1 + 5 ≡ 5 + f1`
  - We can implement `f1 + 5` with another overloaded member function:
    - `foo operator+(int right);`
- But the dot operator's left-hand operand must be an object - not a fundamental type like "int"



COMPLETE SOLUTION

FUNCTIONS

- `foo operator+(foo right);`
- `foo operator+(int right);`
- `friend operator+(int left, foo right)`

FUNCTION CALLS

- `foo f1, f2, f3;`

- `f3 = f1 + f2;  // f1.operator+(f2)`
- `f3 = f1 + f2;  // f1.operator+(5)`
- `f3 = f1 + f2;  // operator+(5, f2)`

ONE FUNCTION TO RULE THEM ALL

- Conversion constructor
  - `foo(int f) : field(f) {}`
- Overloaded operator implemented as a `friend`
  - `friend foo operator+(foo left, foo right);`