FRACTION 1 EXAMPLE

One Class

Multiple Objects

Delroy A. Brinkerhoff

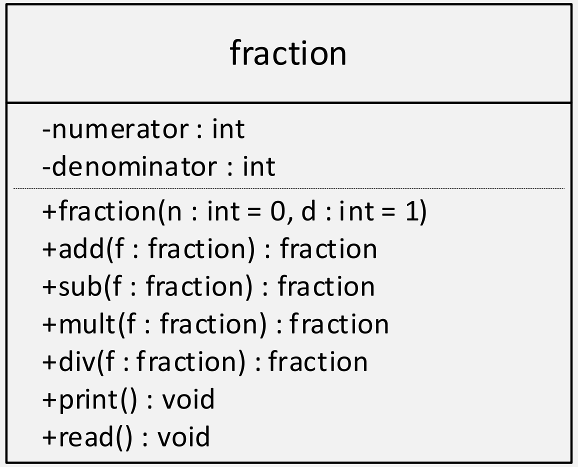
FRACTIONS:
CLASSES AND OBJECTS

- Fractions are a good example of a class
  - Two member variables – simple, but enough to be interesting
  - Multiple ways of building them – interesting constructor functions
  - Algorithmic operations – non-trivial member functions that use the variables
  - I/O – simple but necessary
- Fractions are a good example of a multi-object program
  - Even simple operations involve multiple objects: f1, `f2`, and `f3` are fraction objects
  - `f3 = f1 + f2` translates to `f3 = f1.add(f2);`

REQUIREMENTS

- Default constructor to make an empty fraction: `0/1`
- Conversion constructor to convert an integer to a fraction: 5 to 5/1
- A general constructor to make fraction from two integers: 2 & 3 to 2/3
- Improper fractions are okay, but constructors must reduce new fractions to lowest terms
- Operations do not alter the original fractions
- Each operation creates a new fraction to denote its result
- The output displays the fraction as numerator / denominator: 2/3, 5/3, or 5/1
- The input reads the numerator and denominator one at a time

FRACTION CLASS



fraction

-numerator : int
-denominator : int

+fraction(n : int = 0, d : int = 1)
+add(f : fraction) : fraction
+sub(f : fraction) : fraction
+mult(f : fraction) : fraction
+div(f : fraction) : fraction
+print() : void
+read() : void

```
class fraction
{
    private:
                int             numerator;
                int             denominator;

    public:
                fraction(int n = 0, int d = 1);
                fraction      add(fraction f2) const;
                fraction      sub(fraction f2) const;
                fraction      mult(fraction f2) const;
                fraction      div(fraction f2) const;
                void          print() const;
                void          read();
};
```

FRACTION CONSTRUCTOR

- Fraction f1(2, 3);   2/3
- Fraction f2(5);   5/1
- Fraction f3;   0/1

```
fraction(int n = 0, int d = 1)
    : numerator(n), denominator(d)
{
 int common = gcd(numerator, denominator);
    numerator /= common;
    denominator /= common;
}
```

FRACTION FORMULAS

- Addition: $\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd}$
- Subtraction: $\frac{a}{b}-\frac{c}{d}=\frac{ad-bc}{bd}$
- Multiplication: $\frac{a}{b}\times\frac{c}{d}=\frac{ac}{bd}$
- Division: $\frac{a}{b}\div\frac{c}{d}=\frac{a}{b}\times\frac{d}{c}=\frac{ad}{bc}$

FORMULAS TO OBJECTS

$$\underset{f1}{\boxed{\frac{a}{b}}} + \underset{f2}{\boxed{\frac{c}{d}}} = \underset{f3}{\boxed{\frac{a*d + b*c}{b*d}}}$$

numerator

denominator

- `a = f1.numerator`
- `b = f1.denominator`
- `c = f2.numerator`
- `d = f2.denominator`
- `a*d + b*c = f3.numerator`
- `b*d = f3.denominator`

ADD:
VERSION 1

```
fraction fraction::add(fraction f2) const
{
    fraction f3;
    f3.numerator = numerator * f2.denominator + f2.numerator * denominator;
    f3.denominator = denominator * f2.denominator;
    return f3;
}
```

ADD:
VERSION 2

```
fraction fraction::add(fraction f2) const
{
    int       n = numerator * f2.denominator + f2.numerator * denominator;
    int       d = denominator * f2.denominator;
    return  fraction(n, d);
}
```

ADD:
VERSION 3

```
fraction fraction::add(fraction f2) const
{
    return fraction(numerator * f2.denominator + f2.numerator * denominator,   denominator * f2.denominator);
}
```

FRACTION I/O

```
void fraction::print( ) const
{
    cout << endl << numerator << "/"
        << denominator << endl;
}
```

```
void fraction::read( )
{
    cout << "Please enter the numerator: ";
    cin >> numerator;
    cout << "Please enter the denominator: ";
    cin >> denominator;
}
```