Operator Overloading

Review of Operator Overloading

- Two weeks ago we started talking about overloading operators in classes. What does that mean?
- Which operators did we overload in our examples?
- What are the two ways in which we overload operators?

Rational

```
class Rational
{
  friend ostream &operator<<( ostream &, const Rational & );
  friend istream &operator>>( istream &, Rational & );
  public:
    Rational( int = 0, int = 0 );
    const Rational operator+( const Rational & );
    const Rational operator-( const Rational & );
    const Rational operator*( const Rational & );
    const Rational operator/( const Rational & );
    bool operator>( const Rational & ) const;
    bool operator<( const Rational & ) const;
    bool operator>=( const Rational & ) const;
    bool operator<=( const Rational & ) const;
    bool operator==( const Rational & ) const;
    bool operator!=( const Rational & ) const;
  private:
    int numerator;
    int denominator;
    void reduction();
};
```


Definitions

- Write the definition for:
  - `const Rational operator*( const Rational & );`
- Why are >> and << defined as friend functions and not member functions?
- Write the definition for the >> function so it will allow the user to input a fraction in the form 4/6
  - The client should be able to just type
    - `Rational a;`
    - `cin >> a;`

Definitions

- Write the definition for the function
  - `bool operator>( const Rational & ) const;`

Summary

- We have completed Chapter 8 in the book
- Next time we will look at inheritance