class Rational
{
    public:
        Rational();
        Rational(int n);
        Rational(int n, int d);

        Rational add(Rational r) const;
    ...
};
RationalClient.cpp

Rational f1, f2(3), f3(1,2);

f1 = f2.add(f3);
RationalClient.cpp

Rational f1, f2(3), f3(1,2);

f1 = f2.add(f3);

f1 = f2 + f3;
class Rational
{
  public:
    Rational();
    Rational(int n);
    Rational(int n, int d);

    Rational add(Rational otherRational) const;

    +

  ...;
};
class Rational
{
    public:
        Rational();
        Rational(int n);
        Rational(int n, int d);

        Rational add(Rational otherRational) const;

        Rational operator + (Rational otherRational) ;

    ...}

class Rational
{
    public:
        Rational();
        Rational(int n);
        Rational(int n, int d);

        Rational add(Rational otherRational) const;

        Rational operator + (Rational otherRational) const;

    ...
};
class Rational
{
    public:
        Rational();
        Rational(int n);
        Rational(int n, int d);

        Rational add(Rational otherRational) const;

        Rational operator + (Rational otherRational) const;
        Rational operator - (Rational otherRational) const;
        Rational operator * (Rational otherRational) const;
        Rational operator / (Rational otherRational) const;

    ...
};
Rational operator + (Rational otherRational) const
{
    Rational sum;
    sum.num = num*otherRational.denom + denom*otherRational.num;
    sum.denom = denom*otherRational.denom;
    return sum;
}

}
Rational Rational::operator + (Rational otherRational) const
{
    Rational sum;
    sum.num = num*otherRational.denom + denom*otherRational.num;
    sum.denom = denom*otherRational.denom;
    return sum;
}
Rational Rational::operator + (Rational otherRational) const
{
    Rational sum;
    sum.num = num*otherRational.denom + denom*otherRational.num;
    sum.denom = denom*otherRational.denom;
    return sum;
}

\[
\begin{array}{c c c}
\frac{a}{b} & + & \frac{c}{d} \\
\hline
\end{array} = \frac{ad + bc}{bd}
\]
Rational `Rational::operator + (Rational otherRational) const`
{
    Rational sum;

    sum.num = num*otherRational.denom + denom*otherRational.num;
    sum.denom = denom*otherRational.denom;

    return sum;
}
Rational Rational::operator + (Rational otherRational) const {
    Rational sum;
    sum.num = num*otherRational.denom + denom*otherRational.num;
    sum.denom = denom*otherRational.denom;
    return sum;
}

a        c        ad + bc
+       =
b        d            bd
Rational f1, f2(3), f3(1,2);

f1 = f2 + f3;
Rational f1, f2(3), f3(1,2);

f1 = f2 + f3;
f1 = f2 - f3;
f1 = f2 * f3;
f1 = f2 / f3;
class Rational
{
  public:
    void read (istream& inputStream);
    void write (ostream& outputStream) const;

    ...
};
RationalClient.cpp

Rational f;

...

f.read (cin);
f.write(cout);
RationalClient.cpp

Rational f;

...

f.read (cin);
f.write(cout);

```cpp
cin >> f;
cout << f;
```
class Rational
{
    public:
        void read (istream& inputStream);
        void write (ostream& outputStream) const;

        Rational operator + (Rational otherRational) const;
        Rational operator - (Rational otherRational) const;
        Rational operator * (Rational otherRational) const;
        Rational operator / (Rational otherRational) const;

        ...
};
class Rational
{
    public:
        void read (istream& inputStream);
        void write (ostream& outputStream) const;

        Rational operator + (Rational otherRational) const;
        Rational operator - (Rational otherRational) const;
        Rational operator * (Rational otherRational) const;
        Rational operator / (Rational otherRational) const;

     ... 
}

        operator >>
class Rational
{
    public:
        void read (istream& inputStream);
        void write (ostream& outputStream) const;

        Rational operator + (Rational otherRational) const;
        Rational operator - (Rational otherRational) const;
        Rational operator * (Rational otherRational) const;
        Rational operator / (Rational otherRational) const;

        ...
class Rational
{
  public:
    void read (istream& inputStream);
    void write (ostream& outputStream) const;

    Rational operator + (Rational otherRational) const;
    Rational operator - (Rational otherRational) const;
    Rational operator * (Rational otherRational) const;
    Rational operator / (Rational otherRational) const;

    ...
};

    operator >> (istream& ins, Rational& f);
class Rational {
    public:
        void read (istream& inputStream);
        void write (ostream& outputStream) const;

        Rational operator + (Rational otherRational) const;
        Rational operator - (Rational otherRational) const;
        Rational operator * (Rational otherRational) const;
        Rational operator / (Rational otherRational) const;

    ... 
};

    istream& operator >> (istream& ins, Rational& f);
class Rational
{
    public:
    void read (istream& inputStream);
    void write (ostream& outputStream) const;
    
    Rational operator + (Rational otherRational) const;
    Rational operator - (Rational otherRational) const;
    Rational operator * (Rational otherRational) const;
    Rational operator / (Rational otherRational) const;
    
    ... 
};

    istream& operator >> (istream& ins, Rational& f);
    ostream& operator << (ostream& outs, Rational f);
```cpp
void Rational::read (istream& ins) 
{
    char slash;
    ins >> num;
    ins >> slash;
    ins >> denom;
}
```
void Rational::read (istream& ins)
{
    char slash;
    ins >> num;
    ins >> slash;
    ins >> denom;
}

istream& operator >> (istream& ins, Rational& f)
{
    // Implementation
}

void Rational::read (istream& ins)
{
    char slash;
    ins >> num;
    ins >> slash;
    ins >> denom;
}

istream& operator >> (istream& ins, Rational& f)
{
    f.read(ins);
    return ins;
}
```cpp
void Rational::write (ostream& outs) const
{
    outs << num << "/" << denom;
}
```
void Rational::write (ostream& outs) const
{
    outs << num << "/" << denom;
}

ostream& operator << (ostream& outs, Rational f)
{
}

void Rational::write (ostream& outs) const
{
    outs << num << "/" << denom;
}

ostream& operator << (ostream& outs, Rational f)
{
    f.write(outs);
    return outs;
}
void Rational::write (ostream& outs) const
{
    outs << num << "/" << denom;
}

/*
ostream& operator << (ostream& outs, Rational f)
{
    f.write(outs);
    return outs;
}
*/

ostream& operator << (ostream& outs, Rational f)
{
    outs << f.num << "/" << f.denom; // Will not compile because num & denom are private
    return outs;
}
class Rational {
    public:
    
    void read (istream& ins );
    void write (ostream& outs) const;
    ...

};

istream& operator >> (istream& ins, Rational& f);
ostream& operator << (ostream& outs, Rational f);
class Rational
{
    public:
        void read  (istream& ins );
        void write (ostream& outs) const;
        ...

friend istream& operator >> (istream& ins, Rational& f);
friend ostream& operator << (ostream& outs, Rational  f);
};
void Rational::write (ostream& outs) const
{
    outs << num << "/" << denom;
}

/*
ostream& operator << (ostream& outs, Rational f)
{
    f.write(outs);
    return outs;
}
*/

ostream& operator << (ostream& outs, Rational f)
{
    outs << f.num << "/" << f.denom; // Will compile now because
    return outs; // a friend function can
    // access private variables