```
/*
 * Complex.h
 *
 *  Created on: Oct 23, 2013
 *      Author: Nathaniel Gallinger
 */

#ifndef COMPLEX_H_
#define COMPLEX_H_

#include <cstdlib>
#include <iostream>
using std::ostream;
using std::istream;

namespace NathanielGallinger
{
  class Complex
  {
    // friend function stream operator overloads
    friend ostream &operator<<(ostream &out, const Complex &value);
    friend istream &operator>>(istream &in, Complex &value);

  public:
    // Constructor
    Complex(double 0.0, double 0.0);

    // member function operator overloads
    Complex operator+(const Complex &op1, const Complex &op2);
    Complex operator-(const Complex &op1, const Complex &op2);
    bool operator==(const Complex &op1, const Complex &op2);
    bool operator!=(const Complex &op1, const Complex &op2);

  private:
    // Private data members
    double real;
    double imaginary;
  };
}

#endif /* COMPLEX_H_ */


/*
 * Complex.cpp
 *
 *  Created on: Oct 23, 2013
 *      Author: Nathaniel Gallinger
 */
#include <cstdlib>
```

```cpp
#include <iostream>
#include "Complex.h"
using std::cerr;
using std::cout;
using std::ostream;
using std::istream;
using namespace NathanielGallinger;
using NathanielGallinger::Complex;

// << operator overload
ostream &operator<<(ostream &out, const const Complex &value)
{
  out << value.real;

  // if positive, print positive sign
  if (value.imaginary >= 0)
    out << "+";
  out << value.imaginary;
  out << "i";

  return out;
}

// >> operator overload
istream &operator>>(istream &in, Complex &value)
{
  in >> value.real;
  in >> value.imaginary;

  return in;
}

// Constructor
Complex(double real = 0, double imaginary = 0)
{
  this->real = real;
  this->imaginary = imaginary;
}

// + operator overload
Complex operator+(const Complex &op1, const Complex &op2)
{
  double resultReal, resultImag;

  resultReal = op1.real + op2.real;
  resultImag = op1.imaginary + op2.imaginary;

  return Complex(resultReal, resultImag);
}
```

```
// - operator overload
Complex operator-(const Complex &op1, const Complex &op2)
{
  double resultReal, resultImag;

  resultReal = op1.real - op2.real;
  resultImag = op1.imaginary - op2.imaginary;

  return Complex(resultReal, resultImag);
}

// == operator overload
bool operator==(const Complex &op1, const Complex &op2)
{
  return op1.real == op2.real && op1.imaginary == op2.imaginary;
}

// != operator overload
bool operator!=(const Complex &op1, const Complex &op2)
{
  return !(op1 == op2);
}


/*
 * hw4.cpp
 *
 *  Created on: Oct 23, 2013
 *      Author: Nathaniel Gallinger
 */

#include "Complex.h"
#include <iostream>
using std::cout;
using NathanielGallinger::Complex;

int main()
{
  // Create 2 complex objects
  Complex c1(3, 5);
  Complex c2(8, 7);

  // Test overloaded operators
  Complex result1 = operator+(c1, c2);
  cout << result1 << "\n";
  Complex result2 = operator-(c1, c2);
  cout << result2 << "\n";
  cin >> c1 >> c2;
  cout << c1 << " == " << c2 << ": " << (c1 == c2) << "\n";
  cout << c1 << " != " << c2 << ": " << (c1 != c2) << "\n";
}
```

```
Output:
11+12i
-5-2i
3+5i
8+7i
3+5i == 8+7i: 0
3+5i != 8+7i: 1
```