Friend
class Money
{
public:
    Money(int D, int C);
    int getDollars();
    int getCents();
private:
    int dollars;
    int cents;
};

• Overload ‘==‘:

Money m1(5, 25), m2(5, 25);
...
If (m1==m2)
    cout << "m1 and m2 are the same amount of money\n";
...
Overloading as non-Member Functions

bool operator==(const Money& m1, const Money& m2) {
    return (m1.getDollars() == m2.getDollars()) &&
           m1.getCents() == m2.getCents();
}

• Equality operator, ==
  – Not defined as member function
  – Cannot access member variables by name
  – Overhead: two invocations of getDollars() and two
    invocations of getCents()
  – Hard to read
Friend Functions

• Friends can directly access private class data
  – No overhead, more efficient
• Operator must have access anyway
  – So: best to make nonmember operator overloads friends!
• Friends can be any function
class Money
{
public:
    Money(int D, int C);
    int getDollars();
    int getCents();
    friend bool operator ==(const Money& m1, const Money& m2);  
    friend Money operator +(const Money& m1, const Money& m2);
private:
    int dollars;
    int cents;
};
bool operator == (const Money& m1,
                 const Money& m2)
{
    return (m1.dollars == m2.dollars) &&
            m1.cents == m2.cents);
}
Friend Class

- Entire classes can be friends
  - Similar to function being friend to class
  - class F is friend of class C
    - All class F member functions are friends of C
    - NOT reciprocated

```cpp
class F;
class C
{
 public:
   ...
   friend class F;
   ...
};
class F
{
   ...
};
```