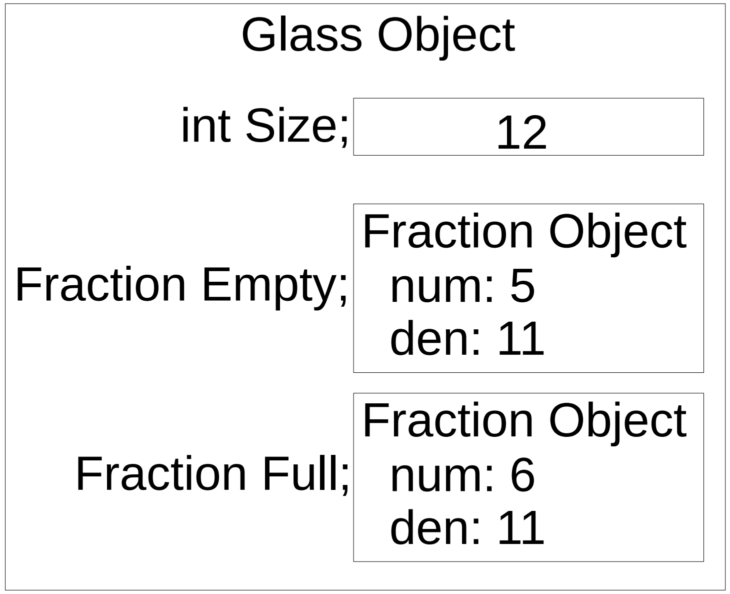
# Operator Composition

# Object's as Member Data

As we have previously learned, objects are a combination of member data, member functions and an interface.

But objects themselves are data (eg. We declare an object just as an int or float and use it)

Objects can also be member data (objects within objects).



Here we have an Object of class Glass.

It's member data includes:

The size of the glass in ounces (int).s

A Fraction object that represents the fraction of the glass that is empty.

A Fraction object that represents the fraction of the glass that is full.

# Composition

The relationship of “an object within an object” is called **composition.**

- Can be implemented by declaring an object or an object pointer/reference within the member data of a class.
- Often described as the “has-a” relationship
  - Glass has-a Fraction
  - Car “has a” Engine (object Engine is member data within Car class)
  - Deck object has 52 Card objects

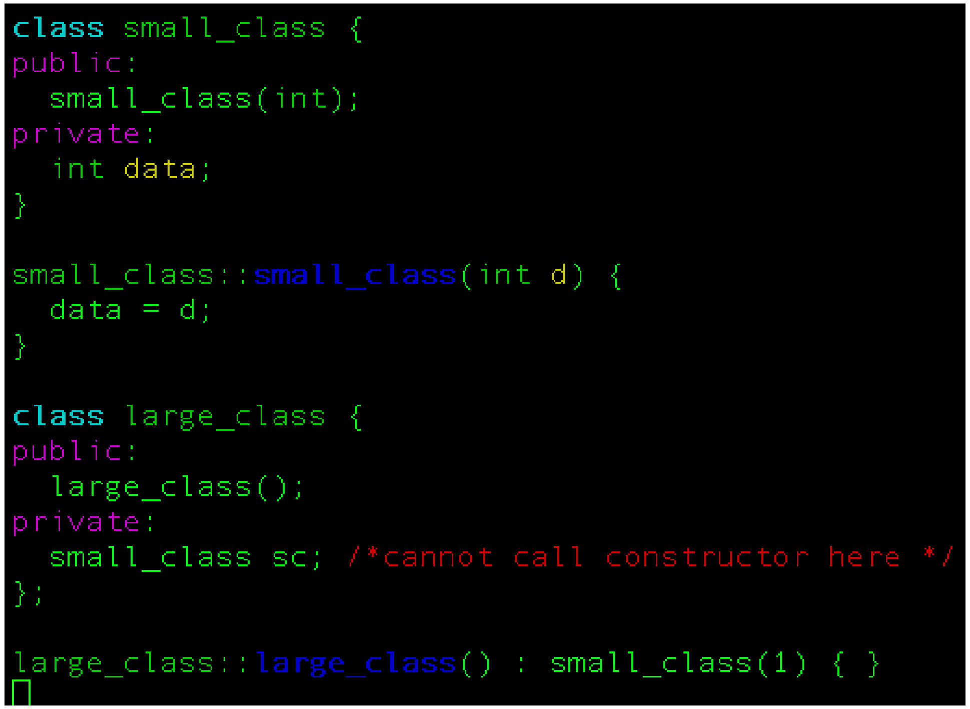
Composition allows code to be more modularized

- We can create smaller classes and combine them to realize larger functionality.
- See PokerHand Example.

# Member Data Object Constructor

If nothing special is done, an object that is member data of another object is initialized with it's default constructor.

If we wish to call another constructor, we can use an initialization list:



```
class small_class {
public:
  small_class(int);
private:
  int data;
}

small_class::small_class(int d) {
  data = d;
}

class large_class {
public:
  large_class();
private:
  small_class sc; /*cannot call constructor here */
};

large_class::large_class() : small_class(1) { }
```

## Extending the dot Operator

- If and object that is member data of another object has public members (data or functions), we can access it using the dot operator.
- See sample1.cpp