Review

- What is an object?
- What is a class?
- What are the steps to creating a new object type and using it?
- What is the scope resolution operator?
Class interface: Protection Levels and Constructors
Protection Levels

When we design a class we can control how member functions and data of that class can be accessed.

Each member of the class is assigned a protection level:

- **Public Members**: Object data and functions that can be accessed from both outside and inside the member functions of the object.
- **Private Members**: Object data and functions that can be accessed only from within member functions of the object.
  - Code that implements the class has accesses to private data, code that uses the class does not have access to private data.

The primary purpose of protection levels is to allow programmers to provide an unchanging **public interface** for a class while being able to modify the way the class is implemented without breaking code that uses the class.

This is a mechanism for information hiding.
Program output:
Decimal: 0.5

Now let's uncomment line 33,

Compiler output:

**sample1.cpp**: In function ‘int main()’:

**sample1.cpp:9**: error: ‘int Fraction::numer’ is private

**sample1.cpp:33**: error: within this context
Program output:

Hello!!!

Decimal: 0.5

With line 39 uncommented,

Compiler output:

sample2.cpp:32: error: `void Fraction::PrintHello()` is private
sample2.cpp:39: error: within this context
More on Protection Levels

Reasons for data hiding (private members):

- Makes interface simpler for user.
- Principle of least privileged (need-to-know)
- More secure. Less chance for misuse (accidental or malicious).
- Class implementation easy to change without affecting other modules that use it.

Other things to note:

- If protection levels not given, members default to private.
- When designing a class we should make private any members not part of the class interface.
Constructors

A constructor is a special member function in a class whose purpose is usually to initialize the members of an object.

A constructor is easy to recognize because:

- It has the same name as the class.
- It has no return type.

Circle Example:

```cpp
// declaration of a class of Circle objects

class Circle
{
public:
    Circle(); // this is a CONSTRUCTOR
    Circle(double r); // this is also a constructor

    void SetCenter(double x, double y);
    void SetRadius(double r);
    void Draw();
    double AreaOf();

private:
    double radius;
    double center_x;
    double center_y;
};
```
A constructor is a function so you can write any code inside it you want.

What is special about constructors?

- Called automatically whenever you instantiate an object (eg. Circle C1; )
- The constructor function is not called explicitly (eg. C1.Circle())

The term **default constructor** refers to a constructor with no parameters.

Every class **must** have a constructor, if you do not supply one an empty default constructor is created automatically.

How do we pass arguments to constructors with parameters?
Passing Arguments to a Constructor

It is simple, really. Just add (…) as part of the object instantiation:

```cpp
Circle C1(5); /* declares a new Circle object with the argument 5 passed to the second Circle constructor as parameter r */
```

Let's look at the Fraction example...