Review

- What is a class?
- What is an object?
- What are the steps to creating a new object type and using it?
- What is the scope resolution operator?

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.

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.

sample1.cpp

```
1  #include <iostream>
2
3  class Fraction {
4  public:
5    void SetNumerator(int n);
6    void SetDenominator(int d);
7    double ToDecimal();
8  private:
9    int numer;
10   int denom;
11 };
12
13 void Fraction::SetNumerator(int n) {
14   numer = n; /* notice we are accessing
15                 private member numer
16                 here. */
17 }
18
19 void Fraction::SetDenominator(int d) {
20   if(d != 0)
21     denom = d;
22   else
23     denom = 1;
24 }
25
26 double Fraction::ToDecimal() {
27   return (double) numer / denom; /* integer division */
28 }
29
30 int main() {
31   Fraction F;
32
33   /* F.numer = 1 */
34   F.SetNumerator(1);
35   F.SetDenominator(2);
36
37   std::cout<<"Decimal: "<<F.ToDecimal()<<std::endl;
38   return 0;
39 }
```

Program output:

Decimal:0.5

Now lets uncomment line 33, Compiler output:

protection.cpp: In function ‘int main()’:

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

protection.cpp:33: error: within this context

sample2.cpp

```
1  #include <iostream>
2
3  class Fraction {
4  public:
5    void SetNumerator(int n);
6    void SetDenominator(int d);
7    double ToDecimal();
8  private:
9    void PrintHello();
10   int numer;
11   int denom;
12 };
13
14 void Fraction::SetNumerator(int n) {
15   numer = n; /* notice we are accessing
16                 private member numer
17                 here. */
18 }
19
20 void Fraction::SetDenominator(int d) {
21   if(d != 0)
22     denom = d;
23   else
24     denom = 1;
25 }
26
27 double Fraction::ToDecimal() {
28   PrintHello();
29   return (double) numer / denom; /* integer division */
30 }
31
32 void Fraction::PrintHello() {
33   std::cout<<"Hello!!!"<<std::endl;
34 }
35
36 int main() {
37   Fraction F;
38
39   /* F.PrintHello(); */
40   F.SetNumerator(1);
41   F.SetDenominator(2);
42
43   std::cout<<"Decimal: "<<F.ToDecimal()<<std::endl;
44   return 0;
45 }
```

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:

```
// 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;
};
```

Constructors continued

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

```
// 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;
};
```

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

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...