LAB 4
Spring 2011, BESE- 16
C++ Classes and Objects

Objectives

- Create classes with public and private members
- Create member functions of class
- Constructor overloading
- Pass arguments to class members
- Use static and const class members

Submission Requirements

You are expected to complete the assigned tasks within the lab session.

Static Data Members:

Classes can contain static member data and member functions. When a data member is declared as **static**, only one copy of the data is maintained for all objects of the class.

Class members can be declared using the storage class specifier static in the class member list. Only one copy of the static member is shared by all objects of a class in a program. When you declare an object of a class having a static member, the static member is not part of the class object.

A typical use of static members is for recording data common to all objects of a class. For example, you can use a static data member as a counter to store the number of objects of a particular class type that are created. Each time a new object is created, this static data member can be incremented to keep track of the total number of objects.

Static Member Functions:

Like static member variables, static member functions are not attached to any particular object. Because static member functions are not attached to a particular object, they can be called directly by using the class name and the scope operator.

Static member functions can only access static member variables. They can not access non-static member variables. This is because non-static member variables must belong to a class object, and static member functions have no class object to work with!

Static member Functions are accesses using the class name and scope resolution operator.

Example:

```
#include <iostream>
#include <conio>

class IDGenerator
{
        private:
           static int s_nNextID;

        public:
            static int GetNextID() { return s_nNextID++; }
```

```
};

// We'll start generating IDs at 1
int IDGenerator::s_nNextID = 1;

int main()
{
   for (int i=0; i < 5; i++)
      cout << "The next ID is: " << IDGenerator::GetNextID() << endl;
getche();
   return 0;
}
```

Constant Class Members:
Making variables const ensures their values are not accidentally changed. This is particularly important when passing variables by reference, as callers generally will not expect the values they pass to a function to be changed.
Just like the built-in data types (int, double, char, etc…), class objects can be made const by using the const keyword. All const variables must be initialized at time of creation. In the case of built-in data types, initilization is done through explicit or implicit assignment:

```
1 const int nValue = 5; // initialize explicitly
2 const int nValue2(7); // initialize implictly
```

In the case of classes, this initialization is done via constructors:

```
1 const Date cDate; // initialize using default constructor
2 const Date cDate2(10, 16, 2020); // initialize using parameterized constructor
```

If a class is not initialized using a parameterized constructor, a public default constructor *must* be provided — if no public default constructor is provided in this case, a compiler error will occur.

Once a const class object has been initialized via constructor, any attempt to modify the member variables of the object is disallowed, as it would violate the constness of the object. This includes both changing member variables directly (if they are public), or calling member functions that sets the value of member variables.
Example:

```
#include <iostream>
#include <conio>

class Date
        {
        private:
```

```
        int m_nMonth;
        int m_nDay;
        int m_nYear;

        Date() { } // private default constructor
    public:
        Date(int nMonth, int nDay, int nYear)
{
            SetDate(nMonth, nDay, nYear);
        }

        void SetDate(int nMonth, int nDay, int nYear)
        {
            m_nMonth = nMonth;
            m_nDay = nDay;
            m_nYear = nYear;
        }

        int GetMonth() const { return m_nMonth; }
        int GetDay() const { return m_nDay; }
        int GetYear() const { return m_nYear; }
    };

void PrintDate(const Date &cDate)
    {
        // although cDate is const, we can call const member functions
        cout << cDate.GetMonth() << "/" <<
            cDate.GetDay() << "/" <<
            cDate.GetYear() << endl;
    }

    int main()
    {
        const Date cDate(10, 16, 2020);
        PrintDate(cDate);
              getche();
        return 0;
    }
```

Task 1:

1. Write a fraction class with two private data members: a double numerator and a double denominator.
2. Create the four arithmetic functions named add, sub, mult, and div public member functions, each accepting one fraction object passed by value.

The following equation can be used to add fractions: a/b + c/d = (a*d + b*c)/(b*d)

3. Create a public constructor
    a. taking two integer parameters, the numerator first, followed by the denominator
    b. use a default argument to set the denominator to 1 if a denominator is not specified
4. Create a read functions for numerator and one for denominator.