

Summary of the DateClass Example

Our original version of DateClass, which had the data members of the class public:

```
class DateClass
{
    public:
        int month;
        int day;
        void outdate();
};
```

With this declaration, the following could be done in the main program:

```
#include "dateclass.h"

void main()
{
    DateClass today;
    :
    :
    today.month = 2;
    today.day = 27;
    today.outdate();
} // END Main
```

The “information hiding” version of DateClass:

```
class DateClass
{
    public:
        void outdate();
        int setdate(int, int);
        DateClass(int, int);
    private:
        int month;
        int day;
};
```

Now, the data members `month` and `day` are private, so the main (or any other function) no longer has direct access. Instead, a new method has been declared, called `setdate`, that must be used to define the date. An advantage is that now we can verify that the month and day that the user enters is legal by checking it in the `setdate` method.

```
#include "dateclass.h"

void main()
{
    DateClass today;
    :
    :
    today.setdate(2, 27);
    today.outdate();
} // END main
```

Finally, we can provide a *constructor* for the class, which will allow the user to initialize the class in the class declaration, if desired. The constructor is a method that has the same name as the class itself; an example is shown later. With this constructor, the following is legal:

```
#include "dateclass.h"

void main()
{
    DateClass today(2, 27);
    .
    .
    .
} // END main
```

The following are the method definitions for the DateClass. Note that class methods have access to both public and private members of the class (or, put another way, "the scope of a class method includes all members of the class.")

```
#include "dateclass.h"

void DateClass::outdate()
{
    // This version is a little fancier than the one we wrote in class,
    // because it outputs the month as a string.
    switch(month)
    {
        default: cout << "Illegal date";
                 return;
        case 1:  cout << "January";
                 break;
        case 2:  cout << "February";
                 break;
        case 3:  cout << "March";
                 break;
        case 4:  cout << "April";
                 break;
        case 5:  cout << "May";
                 break;
        case 6:  cout << "June";
                 break;
        case 7:  cout << "July";
                 break;
        case 8:  cout << "August";
                 break;
        case 9:  cout << "September";
                 break;
        case 10: cout << "October";
                 break;
        case 11: cout << "November";
                 break;
        case 12: cout << "December";
                 break;
    }
    cout << day;
} // END of outdate
```

```

int DateClass::setdate(int m, int d)
{
    // This method checks the ranges for the month and day. It sets the month
    // and day to zero if there is a problem. The routine returns a 0 if
    // everything went OK, a -1 if there was an error.

    int max;

    if(m < 1 || m > 12)
    {
        month = 0;
        day = 0;
        return(-1);
    }
    switch(m)
    {
        case 1:      // January
        case 3:      // March
        case 5:      // May
        case 7:      // July
        case 8:      // August
        case 10:     // October
        case 12:     // December
            max = 31;
            break;
        case 2:      // February
            max = 28; // ignores leap years!
            break;
        case 4:      // April
        case 6:      // June
        case 9:      // September
        case 11:     // November
            max = 30;
    }
    if(d < 0 || d > max)
    {
        month = 0;
        day = 0;
        return(-1);
    }
    month = m;
    day = d;
    return (0)
} // END of setdate

```

```

DateClass::DateClass(int m, int d)
{
    // This is the constructor for the DateClass Class

    setdate(m, d);
} // END of constructor

```