Supplement IV.F: Immutable Objects and Classes

For Introduction to C++ Programming
By Y. Daniel Liang

If the contents of an object cannot be changed (*except through memberwise copy*) once the object is created, the object is called an *immutable object* and its class is called an *immutable class*. If you delete the set function in the **Circle** class in Listing 9.8, Circle2.h in the text, the class would be immutable because **radius** is private and cannot be changed without a set function.

A class with all private data fields and no mutators is not necessarily immutable. To demonstrate this, let us define two classes: **Person** and **Date**. Figures 1 and 2 give the UML class diagrams of these two classes.

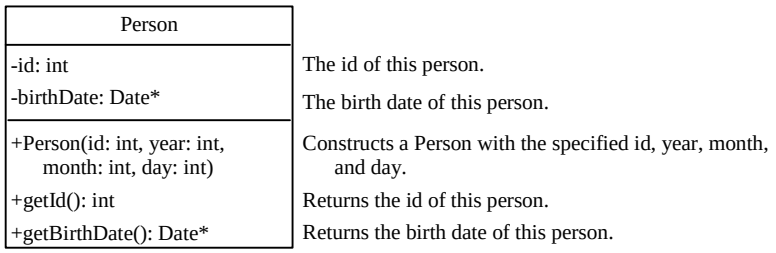

Figure 1
*The **Person** class encapsulates the id and birth date of a person.*

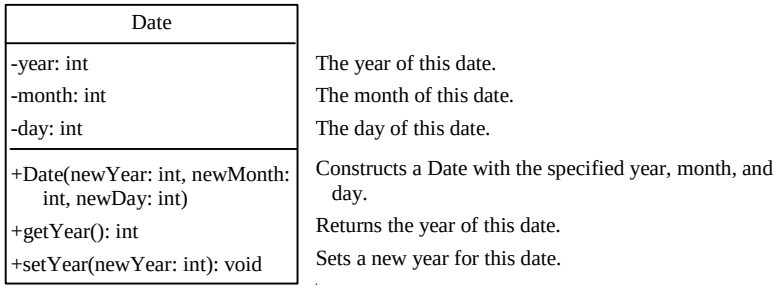

Figure 2
*The **Date** class encapsulates the year, month, and day.*

The **Person** class declaration and implementation are shown in Listings 1 and 2. The **Date** class declaration and implementation are shown in Listings 3 and 4.

Listing 1 Person.h

```
#include "Date.h"

class Person
{
public:
  Person(int id, int year, int month, int day);
  int getId();
  Date* getBirthDate(); // Return the pointer of the object

private:
  int id;
  Date* birthDate; // The pointer of the object
};
```

Listing 2 Person.cpp

```
#include "Person.h"

Person::Person(int id, int year, int month, int day)
{
  this->id = id;
  birthDate = new Date(year, month, day);
}

int Person::getId()
{
  return id;
}

Date* Person::getBirthDate()
{
  return birthDate; // Return the pointer of the object
}
```

Listing 3 Date.h

```
class Date
{
public:
  Date(int newYear, int newMonth, int newDay);
  int getYear();
  void setYear(int newYear);

private:
  int year;
  int month;
  int day;
};
```

Listing 4 Date.cpp

```
#include "Date.h"

Date::Date(int newYear, int newMonth, int newDay)
{
  year = newYear;
  month = newMonth;
  day = newDay;
}

int Date::getYear()
{
  return year;
}

void Date::setYear(int newYear)
{
  year = newYear;
}
```

The **Person** class has all private data fields and no mutators, but it is mutable. As shown in the client program in Listing 5, the data field **birthDate** is returned using the **getBirthDate()** function. This is a pointer to a **Date** object. Through this pointer, the year of the birth date is changed, which effectively changes the contents of the **Person** object.

Listing 5 TestPerson.cpp

```
#include <iostream>
#include "Person.h"
using namespace std;

int main()
{
  Person person(111223333, 1970, 5, 3);
  cout << "birth year before the change is " <<
    person.getBirthDate()->getYear() << endl;
  Date *pDate = person.getBirthDate();
  pDate->setYear(2010);
  cout << "birth year after the change is " <<
    person.getBirthDate()->getYear() << endl;
  return 0;
}
```

Sample output

```
birth year before the change is 1970
birth year after the change is 2010
```

For a class to be immutable, it must mark all data fields private and provide no mutator functions and no accessor functions that would return a reference or a pointer to a mutable data field object.