Ch 13: Introduction to Classes

Part 1

CS 2308
Fall 2011

Jill Seaman

Lecture 10

The Class

- A class in C++ is similar to a structure.
  - It allows you to define a new (composite) data type.
- A class contains:
  - variables AND
  - functions
- These are called members
- Members can be:
  - private: unaccessible outside the class
  - public: accessible outside the class.

Example class declaration

- Example: time.h

```
// file time.h
#include <string>
using namespace std;

  class Time         //new data type
  {
    // models a 12 hour clock
    private:
      int hour;
      int minute;
      void addHour();

    public:
      void setHour(int);
      void setMinute(int);

      string display();
      void addMinute();

  };
```


Access rules

- Private members can be accessed only from other member functions within the class
  - hour and minute cannot be accessed outside
  - must use setHour(int) and setMinute(int) to assign the minute values.
  - No way in this class to directly get the hours or the minutes.
- The public members provide the interface which defines how code outside the class can use instances (objects) of the Time data type.

Defining Member Functions

- the Time.cpp file:

```
// file time.cpp
#include <sstream>
#include <iomanip>
using namespace std;

#include "time.h"

void Time::setHour(int hr) {
  hour = hr;            // hour is a member var
}
void Time::setMinute(int min) {
  minute = min;         // minute is a member var
}

void Time::addHour() {  // a private member func
  if (hour == 12)
     hour = 1;
  else
     hour++;
}
```


Defining Member Functions

- the Time.cpp file cont.:

```
void Time::addMinute()
{
  if (minute == 59) {
     minute = 0;
     addHour();   // call to private member func
  } else
     minute++;
}

string Time::display()
// returns time in string formatted to hh:mm
{
   ostringstream sout;
   sout.fill('0');
   sout << hour << ":" << setw(2) << minute;
   return sout.str();
}
```

Defining member functions

- Member function definitions occur OUTSIDE of the class definition, usually in a separate file.
- The name of each function is preceded by the class name and :: operator
  - Time::setHour(int hr)

Defining an instance of the class

- ClassName objectname:

```
Time t1;
```

- This defines t1 to contain an object of type Time.
- Access public members of class with dot notation:

```
t1.setHour(3);
t1.setMinute(41);
t1.addMinute();
```

- Use dot notation OUTSIDE class only.

A driver progam that uses Time

- File driver.cpp:

```
//using Time class (driver.cpp)
#include<iostream>
#include "time.h"
using namespace std;

int main() {
  Time t;
  t.setHour(12);
  t.setMinute(58);
  cout << t.display() <<endl;
  t.addMinute();
  cout << t.display() << endl;
  t.addMinute();
  cout << t.display() << endl;
  return 0;
}
```


How to compile class + driver

- The quick and dirty way:

```
[...]$ g++ driver.cpp time.cpp
```

- And execute:

```
[...]$ ./a.out
12:58
12:59
1:00
```

Makefile

- makefile:

```
#makefile

timeTest: driver.o time.o
   g++ driver.o time.o -o timeTest

driver.o: driver.cpp time.h
   g++ -c driver.cpp

time.o: time.cpp time.h
   g++ -c time.cpp
```

- Note: “timeTest” is the name of the executable file in this example (not a.out).

Compile class + driver using make

- Make:

```
[...]$ make
g++ -c driver.cpp
g++ -c time.cpp
g++ driver.o time.o -o timeTest
```

- Execute:

```
[...]$ ./timeTest
12:58
12:59
1:00
```

Do not store stale data

- Why not store display string in a variable instead of composing it every time?
- Because it could become stale.
  - If the minute or hour changes, then the data in the object would be inconsistent:
  - stored display string would not match new hours and minutes.
- Don’t store any data that could become stale, compute it in a member function instead.