

Swinburne University of Technology

LABORATORY COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Lab number and title: 1, Part 2 Making a Class in C++
Author: Carmen Chai

1. Let's make a simple class!

First, create an empty project and add main.cpp to it. Remember to include these at the top!

```
#include <iostream>

using namespace std;
```

The class we want to make this lab session is the Time class, and it will store hours, minutes and seconds. All of these will be integer(int) variables.

Add an empty class container for the Time class.

```
class Time
{
};
```

Now let's add the private section of the class which will hold our 3 variables (int hours, int minutes and int seconds). You can declare it either way shown below:

```
class Time
{
private:
    int hours;
    int minutes;
    int seconds;
};
```

```
class Time
{
private:
    int hours, minutes, seconds;
};
```

Now let's add the public section of the class which will hold our methods.
First, add the constructors which will be used to initialize our class:

1. For all the variables (hours, minutes, seconds)
2. For only two of the variables (hours, minutes), seconds set to 0.
3. Default constructor. Where all variables are set to 0.

```
public:
    Time(int h, int m, int s)
    {
        hours = h;
        minutes = m;
        seconds = s;
    }

    Time(int h, int m) {
        hours = h;
        minutes = m;
        seconds = 0;
    }

    //default constructor
    Time() {
        hours = 0;
        minutes = 0;
        seconds = 0;
    }
```

Next, under the constructors, declare some Accessors (Also known as Getters and Setters). These are able to access the class variables and get or set the values.

Note that usually, the Getters are created as const functions to prevent any changes to the class variable values in that function. By making it a const function, the values will remain constant and unchanged.

Now in your main.cpp, add the following:

1. Setter for all 3 time variables
2. Setter for hours
3. Setter for minutes
4. Setter for seconds

5. Getter for hours (const function)
6. Getter for minutes (const function)
7. Getter for seconds (const function)

In this tutorial, we will also be adding some validation for the Setters.

1. Setter for hours should check if value entered is not a negative number (zero and above) and less than or equal to 23. Else, set it to 0.
2. Setter for minutes should check if value entered is not a negative number and less than or equal to 59. Else, set it to 0.
3. Setter for seconds should check if value entered is not a negative number and less than or equal to 59. Else, set it to 0.

Try this part on your own first. The solution is on the next page for your reference :)

More about const here: <https://isocpp.org/wiki/faq/const-correctness#overview-const>

```

void setTime(int h, int m, int s) {
    hours = h;
    minutes = m;
    seconds = s;
}

void setHours(int h) {
    if (h >= 0 && h <= 23)
        hours = h;
    else
        hours = 0;
}

void setMinutes(int m) {
    if (m >= 0 && m <= 59)
        minutes = m;
    else
        minutes = 0;
}

void setSeconds(int s) {
    if (s >= 0 && s <= 59)
        seconds = s;
    else
        seconds = 0;
}

int getHours() const {
    return hours;
}

int getMinutes() const {
    return minutes;
}

int getSeconds() const {
    return seconds;
}

```

Now let's write a simple function for the class that will display the time using cout.

```

void showTime() {
    cout << hours << ":" << minutes
        << ":" << seconds << endl;
}

```

Our Time Class is now complete! Good job! Now on to Part 2!

2. Testing our class in main()

We will run the functions we have created to test and see how our class works. Create a basic main() function as shown below:

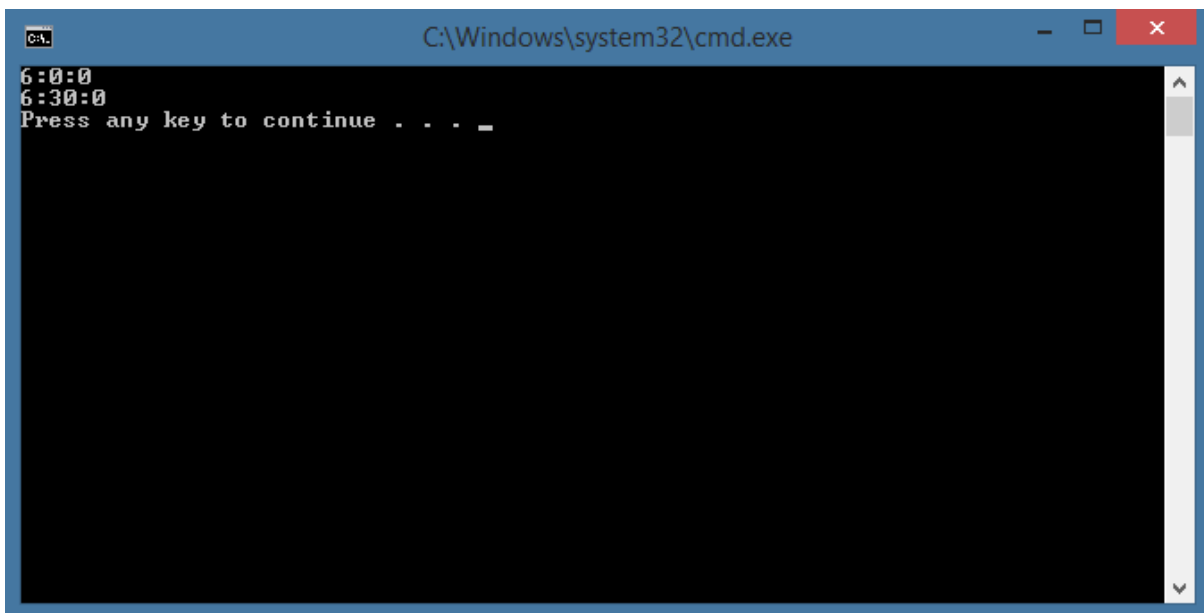
```
int main()
{
    //codes go here
    return 0;
}
```

Now, let's add the following inside the function.

1. Call a constructor for time in our main() function and set it to 6 hours, 0 minutes, 0 seconds.
2. Call the showTime() function.
3. Change the time by using our setMinutes() function. Change the minutes the 30.
4. Call the showTime() function again to see the changes.

```
int main()
{
    Time theTime(6, 0, 0);
    theTime.showTime();
    theTime.setMinutes(30);
    theTime.showTime();
    return 0;
}
```

Build your project and Start without Debugging. You should see the following output.



Congrats! The Time Class works!

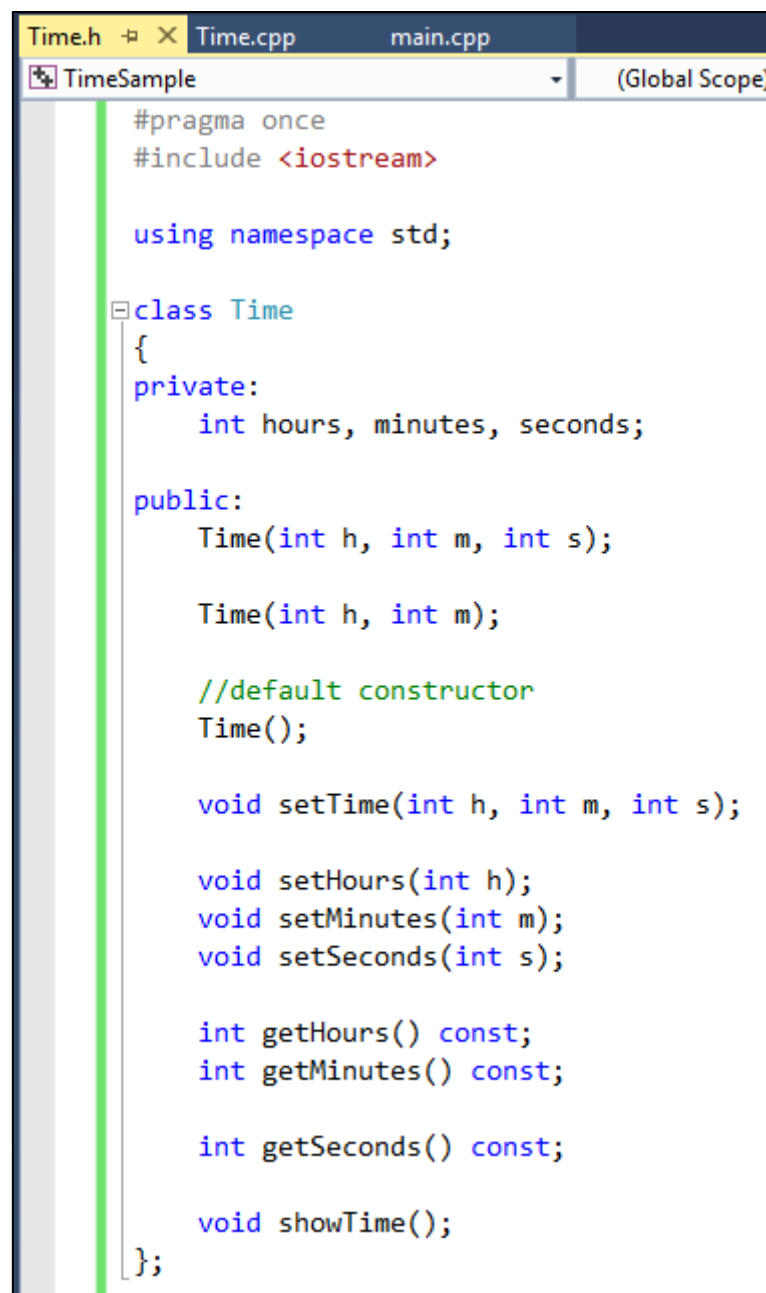
3. Using Class.h and Class.cpp

For the sake of learning how to use a class in a simple way, we've been coding all of them in main.cpp. However, to keep your codes clean and tidy, it is better to let the Class have their own files, a Class.h file and a Class.cpp file.

In our example, it will be the Time.h file and the Time.cpp file.

Create the Time.h file in the Header Files folder, and the Time.cpp in the Source Files folder.

In Time.h, only the parameters the function takes in are defined. The codes or implementation of the function will be defined in Time.cpp.



```
#pragma once
#include <iostream>

using namespace std;

class Time
{
private:
    int hours, minutes, seconds;

public:
    Time(int h, int m, int s);

    Time(int h, int m);

    //default constructor
    Time();

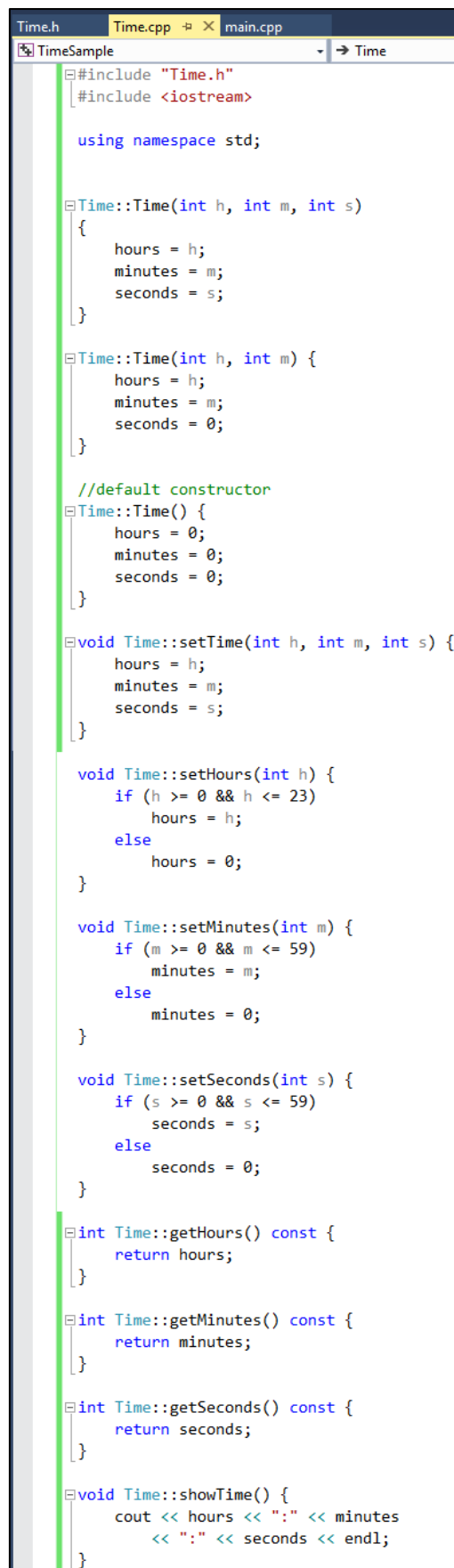
    void setTime(int h, int m, int s);

    void setHours(int h);
    void setMinutes(int m);
    void setSeconds(int s);

    int getHours() const;
    int getMinutes() const;
    int getSeconds() const;

    void showTime();
};
```

At the start of Time.cpp, add to include Time.h. Do this by typing `#include "Time.h"`.



```

Time.h  Time.cpp  main.cpp
TimeSample
Time
#include "Time.h"
#include <iostream>

using namespace std;

Time::Time(int h, int m, int s)
{
    hours = h;
    minutes = m;
    seconds = s;
}

Time::Time(int h, int m) {
    hours = h;
    minutes = m;
    seconds = 0;
}

//default constructor
Time::Time() {
    hours = 0;
    minutes = 0;
    seconds = 0;
}

void Time::setTime(int h, int m, int s) {
    hours = h;
    minutes = m;
    seconds = s;
}

void Time::setHours(int h) {
    if (h >= 0 && h <= 23)
        hours = h;
    else
        hours = 0;
}

void Time::setMinutes(int m) {
    if (m >= 0 && m <= 59)
        minutes = m;
    else
        minutes = 0;
}

void Time::setSeconds(int s) {
    if (s >= 0 && s <= 59)
        seconds = s;
    else
        seconds = 0;
}

int Time::getHours() const {
    return hours;
}

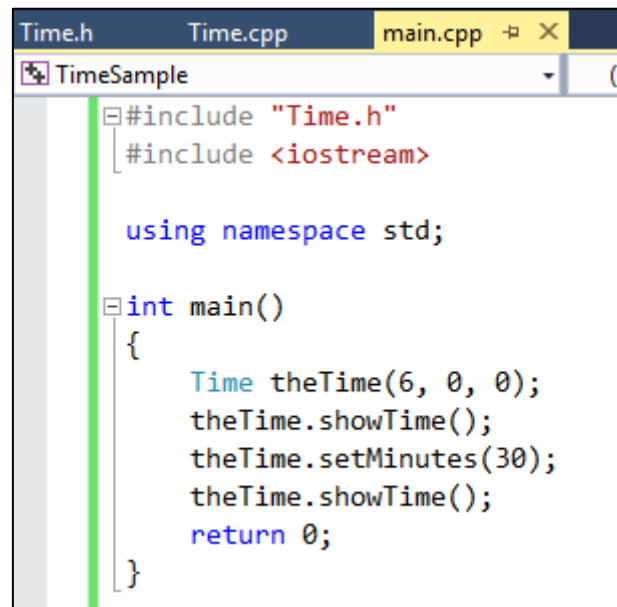
int Time::getMinutes() const {
    return minutes;
}

int Time::getSeconds() const {
    return seconds;
}

void Time::showTime() {
    cout << hours << ":" << minutes
         << ":" << seconds << endl;
}

```

Now that we have moved our Time class to their own files, we only need to include the Time class at the beginning of the main.cpp file to continue using it.



```

Time.h  Time.cpp  main.cpp
TimeSample
#include "Time.h"
#include <iostream>

using namespace std;

int main()
{
    Time theTime(6, 0, 0);
    theTime.showTime();
    theTime.setMinutes(30);
    theTime.showTime();
    return 0;
}

```

You can test it out by pressing Start without Debugging. It should still work exactly the same but now our files are much neater! :)

4. Defining friend operators for input and output

In our lecture, we have learned about “Friend” and see it being used for operator overloading. We will add this to our Time Class now.

Add the following lines in our Time.h.

```

//friend method that returns input stream
friend istream& operator>>(istream& aIstream, Time& aTime);
//friend method that returns output stream
friend ostream& operator<<(ostream& aOstream, Time& aTime);

```

You will see some green squiggly lines beneath operator>> and operator<< at this point. Not to worry, the compiler is simply letting us know that we have not yet defined the operators, which we will do in Time.cpp. Add the following to Time.cpp.

```

istream& operator>> (istream& istream, Time& time) {
    istream >> time.hours >> time.minutes >> time.seconds;
    return istream;
}

ostream& operator<< (ostream& ostream, Time& time) {
    ostream << time.hours << ":" << time.minutes
    << ":" << time.seconds << endl;
    return ostream;
}

```


Now we can use the friend operators to access our Time!

Head to main.cpp and add the following lines (bordered in **red**) before the return 0 statement.

```

Time.h  Time.cpp  main.cpp
TimeSample (Global Scope)
#include "Time.h"
#include <iostream>

using namespace std;

int main()
{
    Time theTime(6, 0, 0);
    theTime.showTime();
    theTime.setMinutes(30);
    theTime.showTime();

    Time newTime; //empty Time
    cout << "Enter a time:" << endl;
    cin >> newTime;
    cout << "This is the time that you have defined: ";
    cout << newTime;

    return 0;
}

```

Using this, we are able to use cin >> and cout << directly with our class. You should see the following output.

```

C:\Windows\system32\cmd.exe
6:0:0
6:30:0
Enter a time:
7 40 30
This is the time that you have defined: 7:40:30
Press any key to continue . . .

```

And that's the end of this tutorial! :) Check out the tutorial this Lab is based on here:

<https://www.youtube.com/watch?v=yQYit5QvcGY>