**Using the this Pointer to Enable Cascaded Function Calls**

**3rd class – Time.h:**

#include <string>

#ifndef TIME\_H

#define TIME\_H

class Time{

public:

explicit Time(int = 0, int = 0, int = 0); // default constructor

// set functions (the Time& return types enable cascading)

Time& setTime(int, int, int); // set hour, minute, second

Time& setHour(int); //set hour

Time& setMinute(int); //set minute

Time& setSecond(int); //set second

unsigned int getHour() const; // return hour

unsigned int getMinute() const; // return minute

unsigned int getSecond() const; // return second

std::string toUniversalString() const; //24-hour time format string

std::string toStandardString() const; //12-hour time format string

private:

unsigned int hour{0};

unsigned int minute{0};

unsigned int second{0};

};

#endif

**2nd class – Time.cpp:**

#include "stdafx.h"

#include <iomanip>

#include <sstream>

#include <stdexcept>

#include "Time.h"

Time::Time(int hr, int min, int sec){

setTime(hr, min, sec);

}

Time& Time::setTime(int h, int m, int s) {

setHour(h);

setMinute(m);

setSecond(s);

return \*this; // enables cascading

}

// set hour value

Time& Time::setHour(int h) { // note Time& return

if (h >= 0 && h < 24) {

hour = h;

}

else {

throw std::invalid\_argument("hour must be 0-23");

}

return \*this; //enables cascading

}

// set minute value

Time& Time::setMinute(int m) {

if (m >= 0 && m < 60) {

minute = m;

}

else {

throw std::invalid\_argument("minute must be 0-59");

}

return \*this;

}

Time& Time::setSecond(int s) {

if (s >= 0 && s < 60) {

second = s;

}

else {

throw std::invalid\_argument("second must be 0-59");

}

return \*this;

}

// get hour value

unsigned int Time::getHour() const { return hour; }

// get minute value

unsigned int Time::getMinute() const { return minute; }

// get second value

unsigned int Time::getSecond() const { return second; }

// return Time asa string in universal-time format (HH:MM:SS)

std::string Time::toUniversalString() const {

std::ostringstream output;

output << std::setfill('0') << std::setw(2) << getHour() << ":"

<< std::setw(2) << getMinute() << ":" << std::setw(2) << getSecond();

return output.str();

}

// return Time as a string in standard-time format (HH:MM:SS AM or PM)

std::string Time::toStandardString() const {

std::ostringstream output;

output << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12)

<< ":" << std::setfill('0') << std::setw(2) << getMinute() << ":"

<< std::setw(2) << getSecond() << (hour < 12 ? " AM" : " PM");

return output.str();

}

**1st class – ConsoleApplication2.cpp:**

#include "stdafx.h"

#include <iostream>

#include "Time.h"

int main()

{

Time t; // create Time object

t.setHour(18).setMinute(30).setSecond(22); //cascaded function calls

// output time in universal and standard formats

std::cout << "Universal time: " << t.toUniversalString()

<< "\nStandard time: " << t.toStandardString();

//cascaded function calls

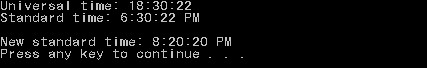
std::cout << "\n\nNew standard time: "

<< t.setTime(20, 20, 20).toStandardString() << std::endl;

return 0;

}

**Result:**



**Important notes:**

