**Time project (better version)**

**3rd class – Time.h:**

#include <string>

//prevents multiple inclusions of heder

#ifndef TIME\_H

#define TIME\_H

//Time class definition

class Time {

public:

explicit Time(int = 0, int = 0, int = 0);

//set functions

void setTime(int, int, int);

void setHour(int);

void setMinute(int);

void setSecond(int);

//get functions

unsigned int getHour() const;

unsigned int getMinute() const;

unsigned int getSecond() const;

std::string toUniversalString() const;

std::string toStandardString() const;

private:

unsigned int hour{ 0 };

unsigned int minute{ 0 };

unsigned int second{ 0 };

};

#endif

**2nd class – Time.cpp:**

#include "stdafx.h"

#include <iomanip>

#include <stdexcept>

#include <sstream>

#include <string>

#include "Time.h"

// Time constructor initializes each data member

Time::Time(int hour, int minute, int second) {

setTime(hour, minute, second);

}

// set new Time value using universal time

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

setHour(h);

setMinute(m);

setSecond(s);

}

// set hour value

void Time::setHour(int h) {

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

hour = h;

}

else {

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

}

}

// set minute value

void Time::setMinute(int m) {

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

minute = m;

}

else {

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

}

}

// set second value

void Time::setSecond(int s) {

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

second = s;

}

else {

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

}

}

//return hour, minute or second values

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

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

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

// return Time as a 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 – CommonApplication5.cpp:**

#include "stdafx.h"

#include <iostream>

#include <stdexcept>

#include "Time.h"

// displays a Time in 24-hour and 12-hour formats

void displayTime(const std::string& message, const Time& time) {

std::cout << message << "\nUniversal time: " << time.toUniversalString()

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

}

int main() {

Time t1;

Time t2{2};

Time t3{ 21, 34 };

Time t4{12, 25, 42 };

std::cout << "Constructed with:\n\n";

displayTime("t1: all arguments defaulted", t1);

displayTime("t2: hour specified; minute and second defaulted", t2);

displayTime("t3: hour and minute specified; second defaulted", t3);

displayTime("t4: hour, minute and second specified", t4);

// attempt to initialize t5 with invalid values

try {

Time t5{ 27, 24, 99 };

}

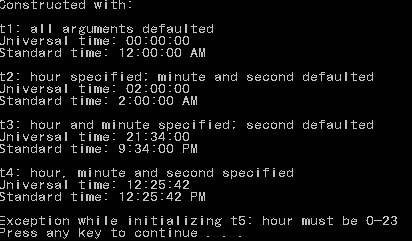
catch (std::invalid\_argument& e) {

std::cerr << "Exception while initializing t5: " << e.what() << std::endl;

}

}

**Result:**



**Important notes:**

* Keep in mind how the ostringstream object was used