// Workshop 2 - Copy and Move Semantics

// Timekeeper.h

// 2019/01/10 - Chris Szalwinski

// 2019/09/13 - Cornel

// 2020/01/14 - Cornel

#ifndef SDDS\_TIMEDEVENTS\_H

#define SDDS\_TIMEDEVENTS\_H

#include <iostream>

#include <chrono>

#include <string>

namespace sdds {

const int MAX\_RECORDS = 7;

class TimedEvents {

int nr {0};

std::chrono::steady\_clock::time\_point ts;

std::chrono::steady\_clock::time\_point te;

struct {

std::string msg{};

std::string units{};

std::chrono::steady\_clock::duration duration{};

} events[MAX\_RECORDS];

public:

void startClock();

void stopClock();

void recordEvent(const char\*);

friend std::ostream& operator<<(std::ostream& out, const TimedEvents& events);

};

}

#endif

// Workshop 2 - Copy and Move Semantics

// Timekeeper.h

// 2019/01/10 - Chris Szalwinski

// 2019/09/13 - Cornel

// 2020/01/14 - Cornel

#include <iomanip>

#include "TimedEvents.h"

namespace sdds

{

// start timer

//

void TimedEvents::startClock()

{

ts = std::chrono::steady\_clock::now();

}

// stop timer

//

void TimedEvents::stopClock()

{

te = std::chrono::steady\_clock::now();

}

// record an event

//

void TimedEvents::recordEvent(const char\* msg)

{

auto ms = std::chrono::duration\_cast<std::chrono::nanoseconds>(te - ts);

if (nr < MAX\_RECORDS)

{

this->events[nr].msg = msg;

this->events[nr].units = "nanoseconds";

this->events[nr].duration = ms;

nr++;

}

}

// report the set of times recorded

//

std::ostream& operator<<(std::ostream& out, const TimedEvents& events)

{

out << "--------------------------\n";

out << "Execution Times:\n";

out << "--------------------------\n";

for (int i = 0; i < events.nr; i++)

out << std::setw(20) << std::left

<< events.events[i].msg << ' ' << std::setw(12) << std::right

<< events.events[i].duration.count() << ' '

<< events.events[i].units << '\n';

out << "--------------------------\n";

return out;

}

}