Creative Software Programming, Assignment 10-2

Handed out: Nov 18, 2021

Due: 23:59, Nov 23, 2021 **(NO SCORE for late submissions!)**

- *Only files submitted by **git push to this course project at https://hconnect.hanyang.ac.kr** (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git) will be scored.*
- *Place your files under the directory structure **<Assignment name>/<Problem no.>/<your files>** just like the following example.*
  - ***DO NOT push CMake intermediate output files** (CMakeCache.txt, cmake_install.cmake, Makefile, CMakeFiles/*).*
  - ***DO NOT use cmake_minimum_required()** command in CMakeLists.txt.*

```
+ 2020_ITE0000_2019000001
  + 2-1/
    + 1/
      - 1.cpp
      - CMakeLists.txt
    + 2/
      - 2.cpp
      - CMakeLists.txt
    + …
```

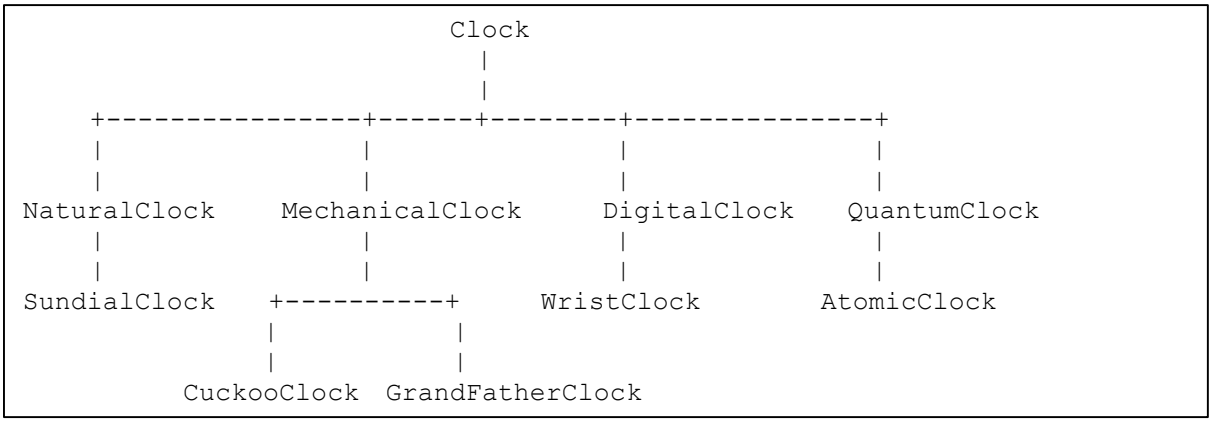
- *The submission time is determined not when the commit is made **but when the git push is made**.*
- *Your files must be committed to the **master branch**. Otherwise, it will not be scored.*
- *Your program should output correct results even for inputs other than those used in the example.*
- *Basically, assignments are scored based on the output results. If it is not possible to check whether a requirement is implemented because the output is not correct, no score is given for the requirement, even if it is implemented internally. However, even if the output result is correct, no score is given for a requirement if the internal implementation does not satisfy the requirement.*

1. Write a program that implements several types of clocks described below.

   A. Simulate the clock's behavior during the input seconds, and print out the current reference time after the simulation and the error of each clock accumulated during the simulation.

   B. The following is the clock type and class hierarchy diagram.

   C.



   D. The clock's behavior is implemented in clock_time.h and clock_time.cpp in the attached zip file. Use these two files without any modification to create the following clock classes.

   E. Clock, NaturalClock, MechanicalClock, DigitalClock, QuantumClock are abstract base classes.

   F. SundialClock, CuckooClock, GrandFatherClock, WristClock, AtomicClock are concrete derived classes.

   G. Class Clock has the following protected member variables:

      i.
```
ClockTime _clockTime;
double _driftPerSecond;
double _totalDrift;
```

   H. The constructor of Clock must take arguments of initial hours, minutes, seconds, and errors per second as follows:

      i.
```
Clock(int hour, int minute, int second, double driftPerSecond)
```

   I. The constructors of NaturalClock, MechanicalClock, DigitalClock, and QuantumClock must take the same arguments as Clock's constructor, and only pass the arguments to the constructor of the parent class.

   J. The constructors of SundialClock, CuckooClock, GrandFatherClock, WristClock, and

AtomicClock take only three parameters: hours, minutes, and seconds, and only pass the arguments and the error per second for each clock type to the constructor of their parent class.

K. The error per second for each clock type:

SundialClock	0.0
CuckooClock	0.0
GrandFatherClock	0.000694444
WristClock	0.000347222
AtomicClock	0.000034722

L. Class Clock has three member functions:

i.
```
void reset();
void tick();
virtual void displayTime() = 0;
```

ii. reset() and tick() are not virtual functions; they exist only in the Clock class.

iii. displayTime() is a pure virtual function, and all concrete derived classes must implement this function.

iv. Clock::reset() simply calls the reset() of _clockTime, a protected member of the Clock class.

v. Clock::tick() calls the increment() of _clockTime, a protected member of the Clock class, and needs to implement some additional functionalities.

vi. displayTime() function of each concrete class basically uses the display() of _clockTime, which is a protected member of class Clock, but you need to do some additional formatting of the displayed text (see the example below).

M. The simulation proceeds in the following order.

i. Create and insert SundialClock, CuckooClock, GrandFatherClock, WristClock, and AtomicClock objects into std::vector<Clock *>. The initial time is 0 hours 0 minutes 0 seconds.

ii. First, reset all clock objects.

iii. Before starting the simulation, print out "Reported clock times after resetting:" and display the current time of each clock using displayTime().

iv. Print out "Running the clocks ...", and then call tick() for the number of seconds input

by the user to run the simulation (1 tick per second).

v. After the simulation, print out "Reported clock times after running:" and display the current time of each clock using displayTime().

vi. Delete clock objects.

N. Input: An integer (representing the simulation time in seconds)

O. Output: The simulation results

P. Files to submit:

i. clock_time.h, clock_time.cpp – Do not modify them.

ii. main.cpp - main() must be in this file.

iii. clocks.h – Class definitions

iv. clocks.cpp – Class member function definitions (implementations)

v. A CMakeLists.txt to generate the executable

```
$ ./clock_time
604800
Reported clock times after resetting:
SundialClock 00:00:00, total drift: 0
CuckooClock 00:00:00, total drift: 0
GrandFatherClock 00:00:00, total drift: 0
WristClock 00:00:00, total drift: 0
AtomicClock 00:00:00, total drift: 0

Running the clocks...

Reported clock times after running:
SundialClock 24:00:00, total drift: 0
CuckooClock 24:00:00, total drift: 0
GrandFatherClock 24:00:00, total drift: 420
WristClock 24:00:00, total drift: 210
AtomicClock 24:00:00, total drift: 20.9999
$
```

2. Write a program that implements several types of clocks described below.

A. Print the alarm from each clock while running the clock.

B. Input: Five integers set to alarm and one simulation time.

C. Output: The simulation results.

D. Files to submit:

i. clock_time.h, clock_time.cpp – Do not modify them.

ii. main.cpp - main() must be in this file.

iii. clocks.h – Class definitions

iv. clocks.cpp – Class member function definitions (implementations)

v. A CMakeLists.txt to generate the executable

```
$ ./clock_time
30 40 50 60 70
604800
Reported clock times after resetting:
SundialClock 00:00:00, total drift: 0
CuckooClock 00:00:00, total drift: 0
GrandFatherClock 00:00:00, total drift: 0
WristClock 00:00:00, total drift: 0
AtomicClock 00:00:00, total drift: 0

Running the clocks...

Sundinal alarm at 00:00:30
Cuckoo alarm at 00:00:40
GrandFather alarm at 00:00:50
Wrist alarm at 00:01:00
Atomic alarm at 00:01:10

Reported clock times after running:
SundialClock 24:00:00, total drift: 0
CuckooClock 24:00:00, total drift: 0
GrandFatherClock 24:00:00, total drift: 420
WristClock 24:00:00, total drift: 210
AtomicClock 24:00:00, total drift: 20.9999
$
```