# 창의적 소프트웨어 설계 실습 문제 10 – hw10-2

**제출 기한**

11월 26일 월 23:59 PM

1. hw10-2(mkdir hw10-2)라는 폴더를 만들고 GitLab에 push
2. hw10-2 디렉토리에 CMakeLists.txt과 각 문제에서 요구하는 파일들을 작성
3. **‘cmake’ 와 ‘make’명령을 수행하여 숙제가 모두 빌드**
4. 최종 버전을 GitLab에 commit
5. **시간과 파일명, 입력과 출력 방식 반드시 지키기**

## 과제 1.

아래의 요구 조건을 만족하고 아래 실행 예와 같이 동작하는 프로그램을 작성하시오.

* Shape 클래스는 순수 가상 함수 (pure virtual function)인 double getArea()와 double getPerimeter(), std::string getTypeString()를 가진다.
* Shape을 상속 받는 Circle 클래스의 constructor는 double형 radius를 받으며, 원의 면적과 둘레를 계산하도록 getArea()와 getPerimeter()를 구현한다. 원주율 PI는 3.141592로 #define하여 사용한다. getTypeString()은 “Circle”을 리턴한다.
* Shape을 상속 받는 Rectangle 클래스의 constructor는 double형 width와 height를 받으며, 사각형의 면적과 둘레를 계산하도록 getArea()와 getPerimeter()를 구현한다. getTypeString()은 “Rectangle”을 리턴한다.
* Circle과 Rectangle에 필요한 멤버 변수를 적절히 추가하여 사용할 것.
* 사용자부터 ‘c’를 입력받으면 이후에 입력되는 하나의 실수는 Circle 객체의 radius이며, ‘r’을 입력받으면 이후에 입력되는 두 실수는 Rectangle 객체의 width와 height를 의미한다. ‘0’이 입력되면 입력이 끝났음을 의미한다.
* 사용자로부터 정해지지 않은 개수의 Circle과 Rectangle들의 정보를 임의의 순서로 받아, j이에 맞게 생성되는 Circle혹은 Rectangle 객체를 std::vector<Shape\*> 타입의 벡터 shapes에 저장한다.
* 사용자가 ‘0’을 입력하여 입력이 종료되면, 지금까지 생성된 Circle혹은 Rectangle객체들의 종류와 면적과 둘레를 각 객체의 getTypeString(), getArea(), getPerimeter() 함수를 호출하여 아래와 같이 순서대로 출력한다.
* 사용이 끝난 shapes의 각 요소는 할당받은 메모리를 해제해야 하며, 프로그램 코드 전체에 걸쳐 type casting 연산자는 사용하면 안 된다.

파일명 : print\_shape (print\_shape.h print\_shape.cc print\_shape\_main.cc)

입력 : 각 객체를 의미하는 문자와 그 객체의 constructor에 필요한 실수들

출력 : 입력받았던 각 객체의 넓이와 둘레

|  |
| --- |
| **$ ./print\_shape**  **c 2.4**  **c 10**  **r 5.2 8.1**  **r 2 4**  **c 7.7**  **0**  **Circle, 18.09556992, 15.0796416**  **Circle, 314.1592, 62.83184**  **Rectangle, 42.12, 26.6**  **Rentangle, 8, 12**  **Circle, 186.26498968, 48.3805168**  **$** |

## 과제 2.

아래에서 설명하는 여러 종류의 시계들을 구현하여 사용자에게 입력 받은 진행 시간 (초) 동안의 시계의 동작을 시뮬레이션 하고, 시뮬레이션이 끝난 후의 현재 기준 시간과 시뮬레이션 동안 누적된 각 시계의 오차를 출력하는 프로그램을 작성하시오.

* 아래는 시계의 종류와 클래스 계층도이다.

Clock

|

|

+-------------------+--------+---------+------------------+

| | | |

| | | |

NaturalClock MechanicalClock DigitalClock QuantumClock

| | | |

| | | |

SundialClock +-------------+ WristClock AtomicClock

| |

| |

CuckooClock GrandFatherClock

* 실제로 시계의 동작의 구현은 아래에 제공되는 clock\_time.h와 clock\_time.cpp에 구현되어 있으며, 이 두 파일을 변형없이 그대로 이용해 아래 시계 클래스들을 작성해야 한다.



* Clock, NaturalClock, MechanicalClock, DigitalClock, QuantumClock는 abstract base class이다.
* SundialClock, CuckooClock, GrandFatherClock, WristClock, AtomicClock는 concrete derived class이다.
* Clock 클래스는 아래 세 개의 protected 멤버 변수를 가진다.

ClockTime \_clockTime;

double \_driftPerSecond;

double \_totalDrift;

* Clock 클래스의 constructor는 아래와 같이 시, 분, 초와 1초당 발생하는 오차를 인자로 받아야 한다.

Clock(int hour, int minute, int second, double driftPerSecond)

* NaturalClock, MechanicalClock, DigitalClock, QuantumClock의 생성자는 Clock의 생성자와 똑같은 인자를 받아야 하며, 부모 클래스의 생성자에게 받은 인자를 그대로 넘겨주는 역할만 수행한다.
* SundialClock, CuckooClock, GrandFatherClock, WristClock, AtomicClock의 생성자는 시, 분, 초 3개의 인자 만을 받으며, 각자의 부모 클래스의 생성자에게 받은 인자와 각 시계 종류별 1초당 오차를 넘겨주면 된다.
* 각 시계 종류별 1초당 오차는 다음과 같다.

|  |  |
| --- | --- |
| SundialClock | 0.0 |
| CuckooClock | 0.0 |
| GrandFatherClock | 0.000694444 |
| WristClock | 0.000347222 |
| AtomicClock | 0.000034722 |

* Clock 클래스는 아래 세 개의 멤버함수를 가진다.

void reset();

void tick();

virtual void displayTime() = 0;

* reset()과 tick()은 virtual function이 아니며, 오직 Clock 클래스에만 존재한다.
* displayTime()은 pure virtual function으로, 모든 concrete derived class들은 이 함수를 구현해야 한다.
* Clock::reset()은 단순히 Clock 클래스의 protected member인 \_clockTime의 reset()함수를 호출하면 된다.
* Clock::tick()은 Clock 클래스의 protected member인 \_clockTime의 increment()함수를 호출하고, 추가적으로 필요한 구현을 해야한다.
* 각 concrete class의 displayTime() 함수는 기본적으로 Clock 클래스의 protected member인 \_clockTime의 display()함수를 이용하되, 각 시계 종류에 맞게 약간의 formatting을 해서 현재의 기준 시간과 그 때까지 누적된 오차를 출력하면 된다 (아래 실행 예 참조)

시뮬레이션은 다음과 같은 순서로 진행한다.

* vector<Clock\*>에 SundialClock, CuckooClock, GrandFatherClock, WristClock, AtomicClock의 객체를 각각 하나씩 순서대로 생성하여 넣는다. 생성할 때의 시간은 0시 0분 0초이다.
* 우선 모든 시계 객체를 reset한다.
* 시뮬레이션을 진행하기 전에, “Reported clock times after resetting:”을 출력한 후 각 시계의 현재 시간을 displayTime()을 이용해 출력한다.
* “Running the clocks...”를 출력한 후 사용자에게 입력 받은 시간 (초) 만큼 tick()을 호출하여 시뮬레이션을 진행한다 (1초당 1 tick)
* 시뮬레이션이 끝난 후, “Reported clock times after running:”을 출력한 후 각 시계의 현재 시간을 displayTime()을 이용해 출력한다.
* 시계 객체들을 delete한다.

파일명 : clock\_time (clock\_time.h clock\_time.cpp clock\_time\_main.cc)

입력 : 입력 시간(초)을 나타내는 정수

출력 : 시뮬레이션 내용을 출력

|  |
| --- |
| **$ ./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: 420**  **GrandFatherClock 24:00:00, total drift: 210**  **WristClock 24:00:00, total drift: 20.9999**  **AtomicClock 24:00:00, total drift: 0**  **$** |