Creative Software Programming, Assignment 8-1

Handed out: Nov 3, 2021

Due: 23:59, Nov 3, 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 works as follows:

   A. Implement the following class Number, Square, and Cube as directed in the comments.

```
class Number
{
    protected:
        int _num;
    public:
        void setNumber(int num)
        {
           _num = num;
        }
    int getNumber()
        {
      return _num;
     }
};

class Square: public Number
{
public:
    int getSquare(); // Implemented to return the square of the number
specified by setNumber()
};

class Cube: public Square
{
public:
    int getCube (); // Implemented to return the cube of the number
specified by setNumber()
};
```

   B.

   C. This program should take user input repeatedly

   D. Input:

      i. 'number' [number] - Create a Number object and print out the return value of getNumber() as shown in the following example.

      ii. 'square' [number] - Create a Square object and print out the return value of getNumber() and getSquare() as shown in the following example.

      iii. 'cube' [number] - Create a Cube object and print out the return value of getNumber(), getSquare(), and getCube() as shown in the following example.

      iv. 'quit' – Quit the program.

   E. Output: The result for each command.

   F. Files to submit:

i. main.cpp – main() must be in this file.

ii. number.h – Just copy the above code skeleton.

iii. number.cpp – Implements Square::getSquare() and Cube::getCube().

iv. A CMakeLists.txt to generate the executable

```
$ ./number
number 3
getNumber(): 3
square 2
getNumber(): 2
getSquare(): 4
cube 4
getNumber(): 4
getSquare(): 16
getCube(): 64
quit
$
```

2. Write a program that works as follows:

   A. Implement the following class Planet, Earth and Moon as directed in the comments.

B.

```
class Planet {
public:
       Planet (float gravity);    // Implement to store necessary data as
member variable
       float drop (float height); // Returns the time elapsed to drop an
object from a specific height
protected:
       // Define member variables you need
};

class Earth: public Planet {
public:
       Earth ();     // Implement to call the parent class's constructor
properly
       void simulate(float height);      // Print out information about
this object (ex. 'Earth gravity = 9.81') and the result of drop
simulation('Drop from 30m, 2.4731 seconds.').
};

class Moon: public Planet {
public:
       Moon ();   // Implement to call the parent class's constructor
properly
       void simulate(float height);      // Print out information about
this object (ex. ' Moon gravity = 1.62') and the result of drop
simulation('Drop from 30m, 6.08581 seconds.').
};
```

C. This program should take user input repeatedly

D. $\text{time} = \sqrt{\dfrac{2 * \text{height}}{\text{gravity}}}$

   i. #include <cmath> → sqrt()

E. Input:

   i. 'Earth' [drop height] - Create an Earth object and print out its information (by calling the print() member function) and the time elapsed to drop an object.

   ii. 'Moon' [drop height] - Create a Moon object and print out its information (by calling the print() member function) and the time elapsed to drop an object.

   iii. 'quit' – Quit the program.

F. Output: The result for each command.

G. Files to submit:

   i. main.cpp – main() must be in this file.

   ii. drop.h – Just copy the above code skeleton. DO NOT modify the code skeleton.

iii. drop.cpp – Implements member functions.

iv. A CMakeLists.txt to generate the executable

```
$ ./drop
Earth 30
Earth gravity = 9.81
Drop from 30m, 2.4731 seconds.
Moon 30
Moon gravity = 1.62
Drop from 30m, 6.08581 seconds.
quit
$
```