Creative Software Programming, Assignment 10-1

Handed out: Nov 17, 2021

Due: 23:59, Nov 17, 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. Define class Shape that has a constructor taking width and height as parameters (double type).
    B. Define class Triangle and class Rectangle which inherit from class Shape. Each subclass also has a constructor taking width and height as parameters (double type).
    C. All classes have a member function, double getArea().
        i. Shape::getArea() is a pure virtual function.
    D. Take inputs from the user and create objects of proper type according to the input by new operator, and then put those objects into std::vector<Shape*> arr.
    E. When the user enters 0, call the getArea() of each element of arr to print out calculated area. Each element of arr must be deallocated after use.
    F. Do not use the type casting operator throughout the code.
    G. This program should take user input repeatedly
    H. Input:
        i. 'r' [width] [height] – Create a rectangle.
        ii. 't' [width] [height] – Create a triangle.
        iii. 0 – Print the results and quit the program.
    I. Output: Areas of the shapes.
    J. Files to submit:
        i. main.cpp – main() must be in this file.
        ii. shape.h – Class definitions
        iii. shape.cpp – Class member function definitions (implementations)
        iv. A CMakeLists.txt to generate the executable

```
$ ./shapes
r 10 5
t 2 4
t 10 5
r 4 3
0
50
4
```

```
25
12
$
```

2. The following program prints out 2. Modify the program using one of the C ++ type casting operators only once so that it prints out 2.5 (actual division result).

   A. Do not use C's casting operator or C++'s reinterpret_cast.

   B.
```
#include <iostream>
using namespace std;
int main()
{
    int a = 10;
    int b = 4;
    cout << a / b << endl;
    return 0;
}
```

   C. Input: None

   D. Output: The division result.

   E. Files to submit:

      i. A C++ source file.

      ii. A CMakeLists.txt to generate the executable

```
$ ./division
2.5
$
```

3. Write a program that works as follows using the following code:

A.
```
class B
{
public:
    virtual ~B() {}
};
class C : public B
{
public:
    void test() { std::cout << "C::test()" << std::endl; }
};
class D : public B
{
public:
    void test() { std::cout << "D::test()" << std::endl; }
};
```

B. Take arbitrary number of "B", "C", and "D" strings, and then creates one object of class B, C, or D for each string, and put those object into std::vector<B*> arr.

C. When the user enters 0, the for statement traverses each element of arr,

   i. Call C::test() if the element is a C type object

   ii. Call D::test() if the element is a D type object (using dynamic_cast)

   iii. Do nothing for the B type object

   iv. Both test() are NOT virtual functions.

D. Note that this problem is intended to practice how to use dynamic_cast, so remember that using dynamic_cast in this way is not desirable.

E. Each element of arr must be deallocated after use.

F. This program should take user input repeatedly

G. Input: B or C or D or 0

H. Output: The result described in the problem.

I. Files to submit:

   i. A C++ source file.

   ii. A CMakeLists.txt to generate the executable

```
$ ./dynamic_cast
B
C
D
B
```

```
C
0
C::test()
D::test()
C::test()
$
```