Creative Software Programming, Assignment 8-2

Handed out: Nov 4, 2021

Due: 23:59, Nov 9, 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 NonSquare and Square that inherits class Shape.

   B.

```
class Shape {
public:
        Shape(int width, int height); //Implement to store necessary data as member variables
        int getArea(); //Returns the area of this rectangle
        int getPerimeter(); //Returns the perimeter of this rectangle
protected:
        //Define member variables you need
};
class Square: public Shape {
public:
        Square (int width); //Implement to call the parent class's constructor properly
        void print(); //Print out information about this object(ex. '5x5Square')
};
class NonSquare: public Shape {
public:
        NonSquare (int width, int height); //Implement to call the parent class's constructor properly
        void print(); //Print out information about this object (ex. '2x7NonSquare')
};
```

   C. This program should take user input repeatedly

   D. Input:

      i. 'nonsquare' [width] [height] - Create a NonSquare object and print out its information (by calling the print() member function), area, and perimeter.

      ii. 'square' [length of one side] - Create a Square object and print out its information (by calling the print() member function), area, and perimeter.

      iii. '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. rect.h – Just copy the above code skeleton. DO NOT modify the code skeleton.

iii. rect.cpp – Implements member functions.

iv. A CMakeLists.txt to generate the executable

```
$ ./rectangle
nonsquare 3 5
3x5 Nonsquare
Area: 15
Perimeter: 16
square 7
7x7 Square
Area: 49
Perimeter: 28
quit
$
```

2. Write a program for drawing 2D shapes.

A. Define class Square, Rectangle, Diamond that inherits from class Shape in the following code.

B.
```
class Shape {
	public:
		Shape();
		Shape(/* required parameters */);

		double GetArea() {};
		int GetPerimeter() {};
		void Draw(int canvas_width, int canvas_height) {};

	protected:
		// Define common properties for all shape types
};
```

C. Complete the definition of class Shape and write the definition of other classes. Define member functions GetArea(), GetPerimeter(), and Draw() in each class.

D. Note

i. Take canvas size (width, height) from the user first

ii. Properties common to all shapes must be member variables of the Shape class.

iii. Define constructors that take necessary information for each class

iv. In subclass' constructor, call the parent's constructor to set common properties.

v. Define member functions GetArea(), GetPerimeter(), and Draw() in each class.

vi. Ignore shape parts outside the canvas.

vii. Empty spaces are printed with '.' and spaces in the shape are printed with brush characters.

E. This program should take user input repeatedly

F. Input:

i. 'rect' [top-left x] [top-left y] [width] [height] [brush] - Create a Rectangle object and call its Draw().

ii. 'square' [top-left x] [top-left y] [length of one side] [brush] - Create a Square object and call its Draw().

iii. 'diamond' [top-center x] [top-center y] [distance from center to each corner] [brush] - Create a Diamond object and call its Draw().

iv. 'quit' – Quit the program.

G. Output: The result for each command.

H. Files to submit:

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

ii. shapes.h – Class definitions

iii. shapes.cpp – Class member function definitions (implementations)

iv. A CMakeLists.txt to generate the executable

```
$ ./draw_shape
10 10 // canvas size: 10 x 10
rect 4 4 5 3 *
Area: 15
Perimeter: 16
 0123456789
0..........
1..........
2..........
```

```
3.........
4....*****.
5....*****.
6....*****.
7.........
8.........
9.........  // Draw a rectangle of width 5 and height 3 with (4, 4) at top left
```

```
diamond 2 5 2 ?
Area: 12.5
Perimeter: 12
 0123456789
0..........
1..........
2..........
3..........
4..........
5..?.......
6.???......
7?????.....
8.???......
9..?.......  // Draw a diamond with (2, 5) at top center, having distance 2 from center to each corner
square 5 5 7 +
Area: 49
Perimeter: 28
 0123456789
0..........
1..........
2..........
3..........
4..........
5.....+++++
6.....+++++
7.....+++++
8.....+++++
9.....+++++ // Draw a square length 7 with (5, 5) at top left
quit
$
```

3.