창의적 소프트웨어 프로그래밍 Lab 9

Handed out : Thu, Oct 14, 2022

Due : Thu, Oct 17, 2022, 23:59 (NO SCORE for late submissions!)

Submit your file on LMS.

Thu, Oct 20, 2022 : Midterm

Fri, Oct 21, 2022 : NO class

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. This program should take user input repeatedly

C. 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.

D. Output: The result for each command.

E. 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 Rectangle, Square, and NonSquare as directed in the comments.

```
class Rectangle {
public:
	Rectangle(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 Rectangle {
public:
	Square (int width); // Implement to call the parent class's constructor properly
	void print(); // Print out information about this object (ex. '5x5 Square')
};

class NonSquare: public Rectangle {
public:
	NonSquare (int width, int height);  // Implement to call the parent class's constructor properly
	void print(); // Print out information about this object (ex. '2x7 NonSquare')
};
```

   B. This program should take user input repeatedly

   C. 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.

   D. Output: The result for each command.

   E. 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
$
```

3. Write a program for drawing 2D shapes.

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

```
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
};
```

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

C. 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.

D. This program should take user input repeatedly

E. 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.

F. Output: The result for each command.

G. 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
$
```