

## 창의적 소프트웨어 프로그래밍 Lab 13

**Handed out : Fri, Nov 4, 2022**

**Due : Mon, Nov 7, 2022, 23:59 (NO SCORE for late submissions!)**

**Submit your file on LMS.**

1. Write a program that works as follows:
  - A. Class Animal has two member variables, `std::string name` and `int age`, and has a constructor that initializes the values of `name` and `age` by taking a string and an integer as arguments.
  - B. Class Zebra that inherits Animal has a member variable, `int numStripes`, and has a constructor that initializes the values of `name`, `age`, and `numStripes` by taking a string, an integer, and an integer as arguments.
    - i. The member variable of the parent class is initialized through the constructor of the parent class.
  - C. Class Cat that inherits Animal has a member variable, `std::string favoriteToy`, and has a constructor that initializes the values of `name`, `age`, and `favoriteToy` by taking a string, an integer, and a string as arguments.
    - i. The member variable of the parent class is initialized through the constructor of the parent class.
  - D. Each class has a member function `printInfo()`
    - i. `Animal::printInfo()` does nothing.
    - ii. `Zebra::printInfo()` and `Cat::printInfo()` print out the type, name, age, and number of stripes (or favorite toy) as shown in the following example.
  - E. Create Zebra or Cat objects according to user input, and put them into `std::vector<Animal*> animals`.
  - F. When the user enters 0, call the `printInfo()` function of each element of `animals`. Each element of `animals` must be deallocated after use.
  - G. Do not use the type casting operator throughout the code.

H. This program should take user input repeatedly

I. **Input:**

- i. 'z' [name] [age] [numStripes] – Create a zebra
- ii. 'c' [name] [age] [favoriteToy] – Create a cat
- iii. 0 – Stop taking inputs and call printInfo() functions..

J. **Output:** The result for calling printInfo() functions

K. Files to submit:

- i. main.cpp – main() must be in this file.
- ii. animal.h – Class definitions
- iii. animal.cpp – Class member function definitions (implementations)
- iv. A CMakeLists.txt to generate the executable

```
$ ./Animals
z Tom 2 21
z Amy 3 22
c Kitty 4 mouse
c King 3 tower
z Elen 5 30
0
Zebra, Name: Tom, Age: 2, Number of stripes: 21
Zebra, Name: Amy, Age: 3, Number of stripes: 22
Cat, Name: Kitty, Age: 4, Favorite toy: mouse
Cat, Name: King, Age: 3, Favorite toy: tower
Zebra, Name: Elen, Age: 5, Number of stripes: 30
$
```

2. Write a program for drawing multiple 2D shapes.

A. Complete the following classes using inheritance.

```
class Canvas {
public:
    // 1 <= row <= 10, 1 <= col <= 10
    Canvas(int row, int col);
    ~Canvas();

    // Dot with the brush at (x,y). If (x,y) is outside the canvas, just
    ignore it.
    void DrawPixel(int x, int y, char brush);

    //Print out the canvas
    void Print();

private:
    // Define data member to save drawn shapes
};

class Shape {
public:
    virtual ~Shape();
    virtual void Draw(Canvas* canvas) {};
protected:
    // Define common properties of shapes
};

class Rectangle : public Shape { /* Define required members */ }
class UpTriangle : public Shape { /* Define required members */ }
class DownTriangle : public Shape { /* Define required members */ }
class Diamond : public Shape { /* Define required members */ }
```

B.

C. Note

- i. Take canvas size (width, height) from the user first.
  1.  $1 \leq \text{width} \leq 10, 1 \leq \text{height} \leq 10$
- ii. Take user input repeatedly to create different objects of child classes of class Shape as shown in the following example.
- iii. Once an object is created, store it in a `std::vector<Shape*>` object. All shape objects should be stored in the single vector object.
- iv. Redraw all shapes whenever displaying the canvas.
- v. All shapes in the canvas must be able to 'overlay'. The shape entered later

overwrites the shape entered earlier.

- vi. 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. 'add' [shape] [shape parameters]
  - 1. 'rect' [top-left x] [top-left y] [width] [height] [brush]
  - 2. 'diamond' [top-center x] [top-center y] [distance from center to each corner] [brush]
  - 3. 'tri\_up' [top-center x] [top-center y] [height] [brush]
  - 4. 'tri\_down' [bottom-center x] [bottom -center y] [height] [brush]
- ii. 'draw' – Draw the canvas
- iii. 'dump' – Print out information about all shapes in the vector.
- 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. canvas.h – Class definitions
- iii. canvas.cpp – Class member function definitions (implementations)
- iv. A CMakeLists.txt to generate the executable

```

$ ./draw_shape
10 10
0123456789
0.....
1.....
2.....
3.....
4.....
5.....
6.....
7.....
8.....
9..... // Draw a empty canvas once when you first create the
canvas
add rect 4 4 3 3 *
draw
0123456789
0.....
1.....
2.....
3.....
4...***...
5...***...
6...***...
7.....
8.....
9..... // Draw a rectangle of width 3 and height 3 with (5, 5)
at top left
add tri_down 3 3 3 @
draw
0123456789
0.....
1.@@@@@...
2..@@@...
3...@.....
4...***...
5...***...
6...***...
7.....
8..... // Draw an inverted triangle of height 3 with (3, 3) an
the bottom center
9.....
add tri_up 5 7 3 #
draw
0123456789
0.....
1.@@@@@...
2..@@@...
3...@.....
4...***...
5...***...
6...***...
7.....#....

```

```

8....###...
9...#####.. // Draw a triangle of height 3 with (7, 7) an the top
center
add diamond 2 5 2 ?
draw
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
add rect 5 5 8 4 +
// Shapes are only drawn inside the canvas
dump
0 rect 4 4 3 3 *
1 tri_down 3 3 3 @
2 tri_up 7 7 3 #
3 diamond 2 5 2 ?
4 rect 5 5 8 4 +
draw
0123456789
0.....
1.@@@@@....
2..@@@.....
3...@.....
4....***...
5..?.*+++++
6.???*+++++
7?????*+++++
8.???#+++++
9..?#####..
quit
$

```