Lecture 5

Object-Oriented Programming I

Introduction to OOP

Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

- Brief overview of object-oriented programming (15.1)
- C++ Features
  - Class (1.5)
  - Inheritance (15.1)
  - Template (107 page, 16.1.1, 16.1.2)
  - Standard Template Library (STL) (3.1-3.4)

Brief Overview of Object-Oriented Programming

- Object-oriented programming (OOP) is a programming paradigm that uses **"objects"** – data structures consisting of **datafields** and **interfaces** (procedures which access the datafields) – to design applications.
  - in Wikipedia _ http://en.wikipedia.org/wiki/Object_oriented

- OOP is based on three fundamental concepts:
  - Encapsulation
    - Encapsulation = Data encapsulation = Data protection
    - Hiding the details of datafields (and interfaces) in the implementation of the procedures.
    - C++ provides various ways to realize encapsulation.
  - Inheritance
    - Hierarchy in the real world is reflected in the programming language.
    - 서울대생은 대학생. 대학생은 대한민국 국민. 주민등록번호를…
    - Results in less amount of coding.
      - Example: 정삼각형은 삼각형, 그러므로 정삼각형의 면적은 삼각형 면적 구하는 procedure를 사용할 수 있음.
  - Dynamic Binding (Virtual Function)
    - Run-time polymorphism
    - Seemingly same function calls produce different responses according to the type of the arguments.
      - Example: t는 삼각형이고 q는 사각형일 때 area(t); area(q); 는 각각 삼각형과 사각형 면적 계산 procedure를 호출함.

C++ Features

- Class
  - Allows to define your own data type with associated procedures.
  - It consists of **datafields** (member variables) and **interfaces** (member functions).

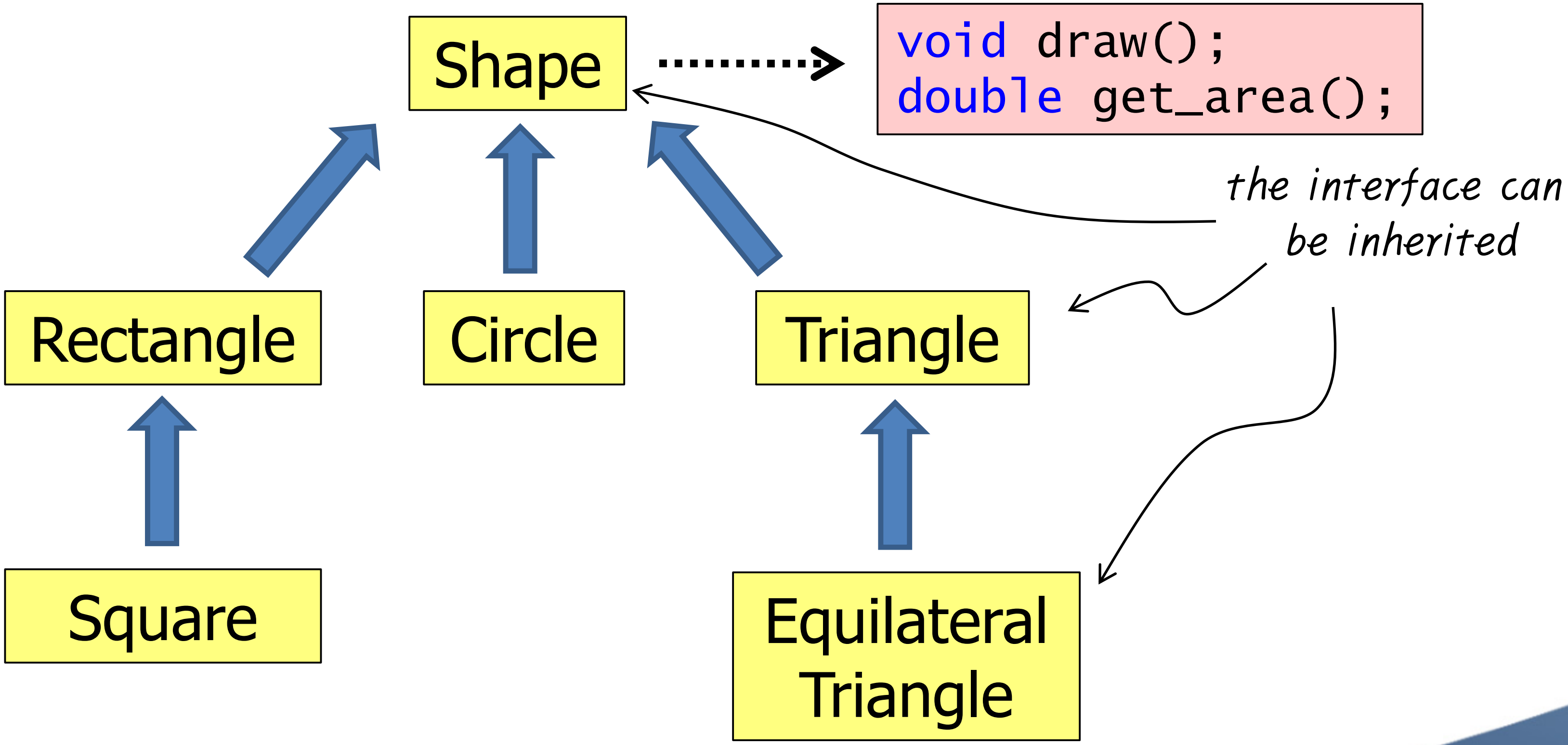
```
class Box {
public:
   Box(double h, double w, double l) : height(h), width(w), length(l) {}

   double volume() { return height*width*length; }
   void print() {
      cout << height << " " << width << " " << length << endl;
   }

private:
   double height, width, length;
};

void main() {
   Box b(10,20,30);
   b.print();
   std::cout << b.volume() << std::endl;
}
```

interfaces

datafields

C++ Features

- Inheritance
  - C++ allows to define inheritance among classes.



```
void draw();
double get_area();
```

the interface can be inherited

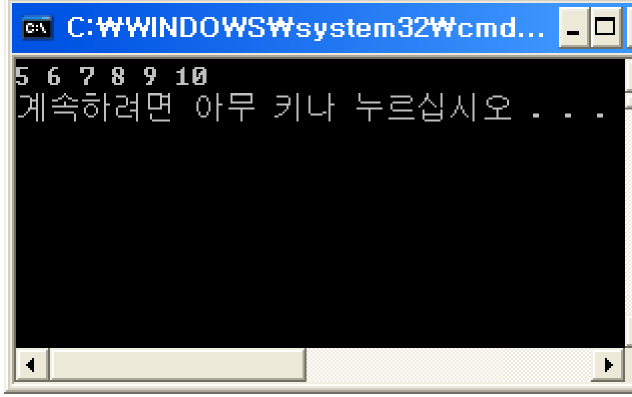
C++ Features

- Template
  - Compile-time polymorphism
  - **Templates** allow functions and classes to operate with generic types.
    - This allows a function or class to work on **many different data types without being rewritten** for each one.

```
#include <iostream>
#include <string>

template<typename T>
T minimum(T a, T b) { return (a < b) ? a : b; }

void main() {
   int i0 = 3, i1 = 5;
   char c0 = ‘a’, c1 = ‘b’;
   string s0 = “apple”, s1 = “animal”;

   std::cout << minimum(i0, i1) << std::endl;
   std::cout << minimum(c0, c1) << std::endl;
   std::cout << minimum(s0, s1) << std::endl;
}
```

C++ Features

- Standard Template Library (STL)
  - It brings huge innovation in C++ programming.
  - It is **generic library** which provides
    - Container (ex. vector, string)
    - Iterators
    - Algorithms
    - Functors
  - It is very **general, efficient, and easy to use.**



```
5 6 7 8 9 10
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <iostream>
#include <vector>

void main() {
   std::vector<int> ivec;
   for(int i=5;i<=10;++i)
      ivec.push_back(i);

   for(std::vector<int>::size_type i=0;i!=ivec.size();++i)
      std::cout << ivec[i] << " ";
   std::cout << std::endl;
}
```