창의적 소프트웨어 설계

8주차 실습 – Polymorphism

노인우, inwoo13@hanyang.ac.kr
한중수, soohan@hanyang.ac.kr

Overview

목표

- Inheritance & Polymorphism
- Virtual Functions
- Casting
  - Upcasting
  - Downcasting

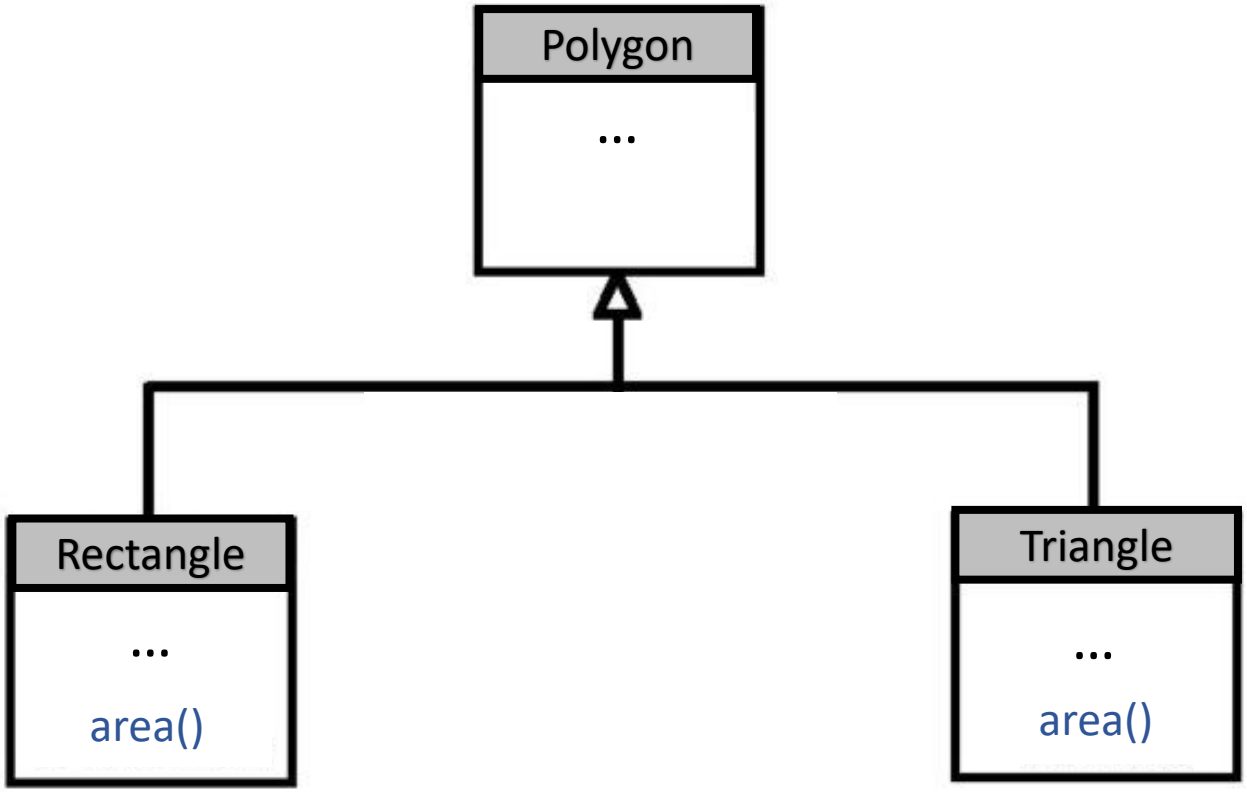
Inheritance & Polymorphism

- Inheritance without Polymorphism is possible!
  - Addition or Extension of base class

- Polymorphism without Inheritance is impossible!
  - Treat objects from different classes the same way
  - Needs virtual inheritance
  - class that declares or inherits a virtual function is called *polymorphic class*

Inheritance & Polymorphism

- Inheritance
  - Addition or Extension of base class



Example

```
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b) { width=a; height=b; }
};

class Rectangle: public Polygon {
  public:
    int area() { return width*height; }
};
```

Example (Cont.)

```
class Triangle: public Polygon {
  public:
    int area() { return width*height/2; }
};

int main () {
  Rectangle rect;
  Triangle trgl;
  rect.set_values (4,5);
  trgl.set_values (4,5);
  cout << rect.area() << '\n';
  cout << trgl.area() << '\n';
  return 0;
}
```

Virtual Functions

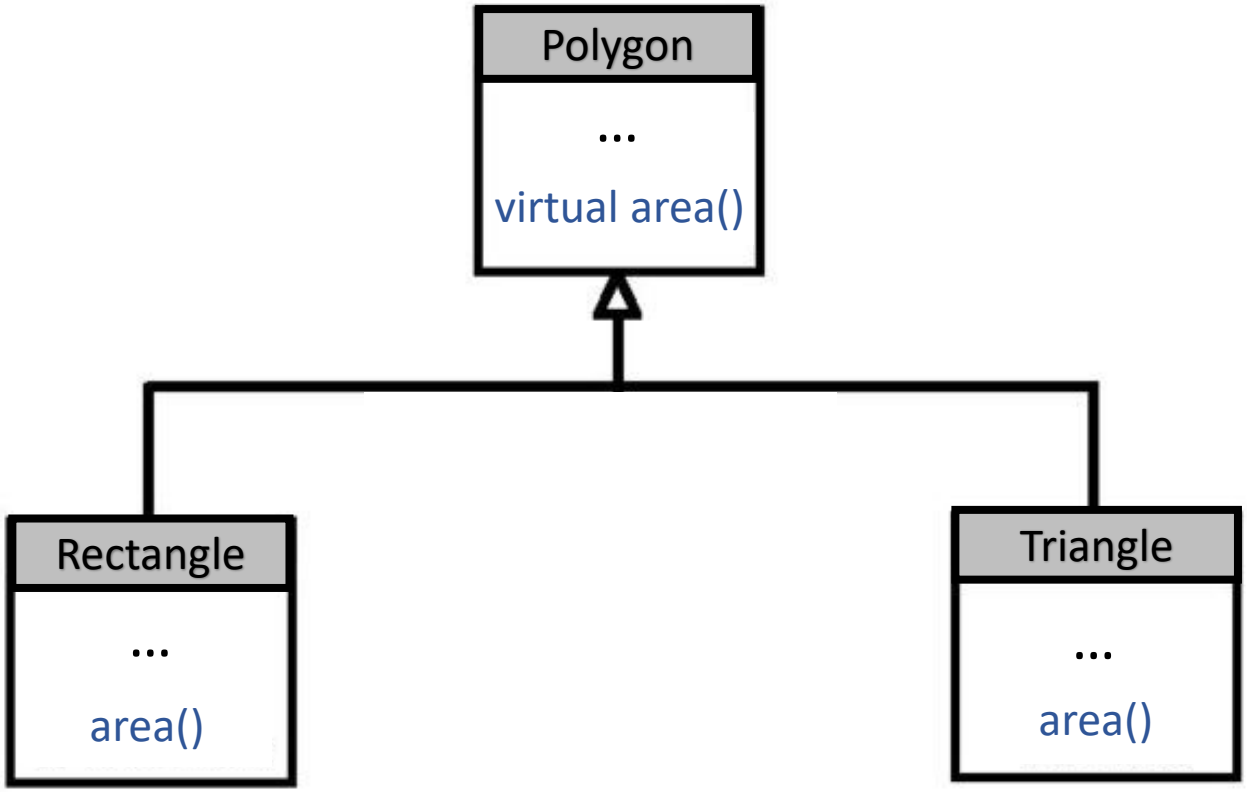
- Virtual Functions
  - Can be redefined in a derived class
  - Preserve calling properties through references
  - Abstract implementation!

```
class BaseClass {
  protected:
    int a, b;
  public:
    virtual int func () { return 0; }
};
```

```
class PolyClass: public BaseClass {
  public:
    int func () { return 1; }
};
```

Inheritance & Polymorphism

- Polymorphism
  - Treat objects from different classes the same way
  - Needs virtual inheritance



Example

```
#include <iostream>
using namespace std;
class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area () { return 0; }
};
class Rectangle: public Polygon {
  public:
    int area () { return width * height; }
};
```

Example (Cont.)

```
class Triangle: public Polygon {
  public:
    int area ()
      { return (width * height / 2); }
};
int main () {
  Rectangle rect;
  Triangle trgl;
```

```
  cout << ppoly1->area() << '\n';
  cout << ppoly2->area() << '\n';
return 0;
}
```

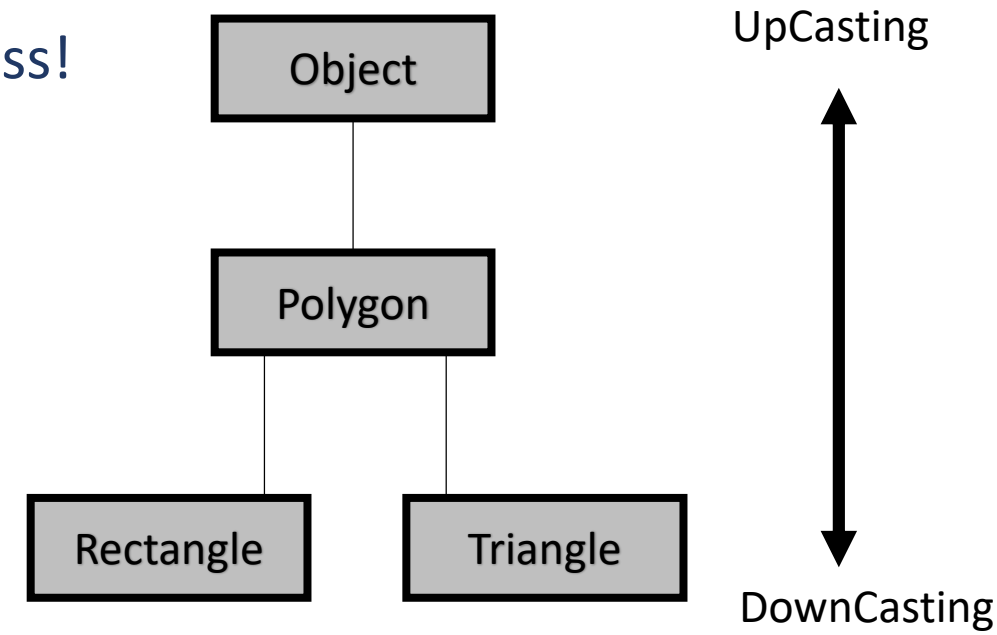
Inheritance & Polymorphism

- Key features
  - Pointer to a derived class is type-compatible with a pointer to its base class
  - Can preserve calling properties through references

Casting

- UpCasting
  - Up to Base Class!
  - Possible! Because derived class includes members of base class
  - ABC abc = abcd;

- DownCasting
  - Down to Derived Class!
  - ABCD abcd = abc;



Example

```
#include <iostream>

using namespace std;

class Base {
  public:
    void showBase() { cout << "Base Function" << endl; }
};

class Derived: public Base {
  public:
    void showDerived() { cout << "Derived Function" << endl; }
};
```

Example

```
int main(void) {
  Derived d2;
  Derived *d1;
  Base *b = &d2;          // UpCasting

  d1 = b;                 // Needs DownCasting

  d1->showDerived();
  d1->showBase();
}
```

```
example3.cpp: In function ‘int main()’:
example3.cpp:21:6: error: invalid conversion from ‘Base*’ to ‘Derived*’ [-fpermissive]
                d1 = b;
                     ^
```

참고자료

1. Polymorphism, https://en.wikipedia.org/wiki/Polymorphism_(computer_science)
2. Polymorphism, http://www.cplusplus.com/doc/tutorial/polymorphism
3. Casting, https://m.blog.naver.com/PostView.nhn?blogId=madplay&logNo=220203111905&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F
4. Casting, http://www.cs.utexas.edu/~cannata/cs345/Class%20Notes/14%20Java%20Upcasting%20Downcasting.htm


Q&A