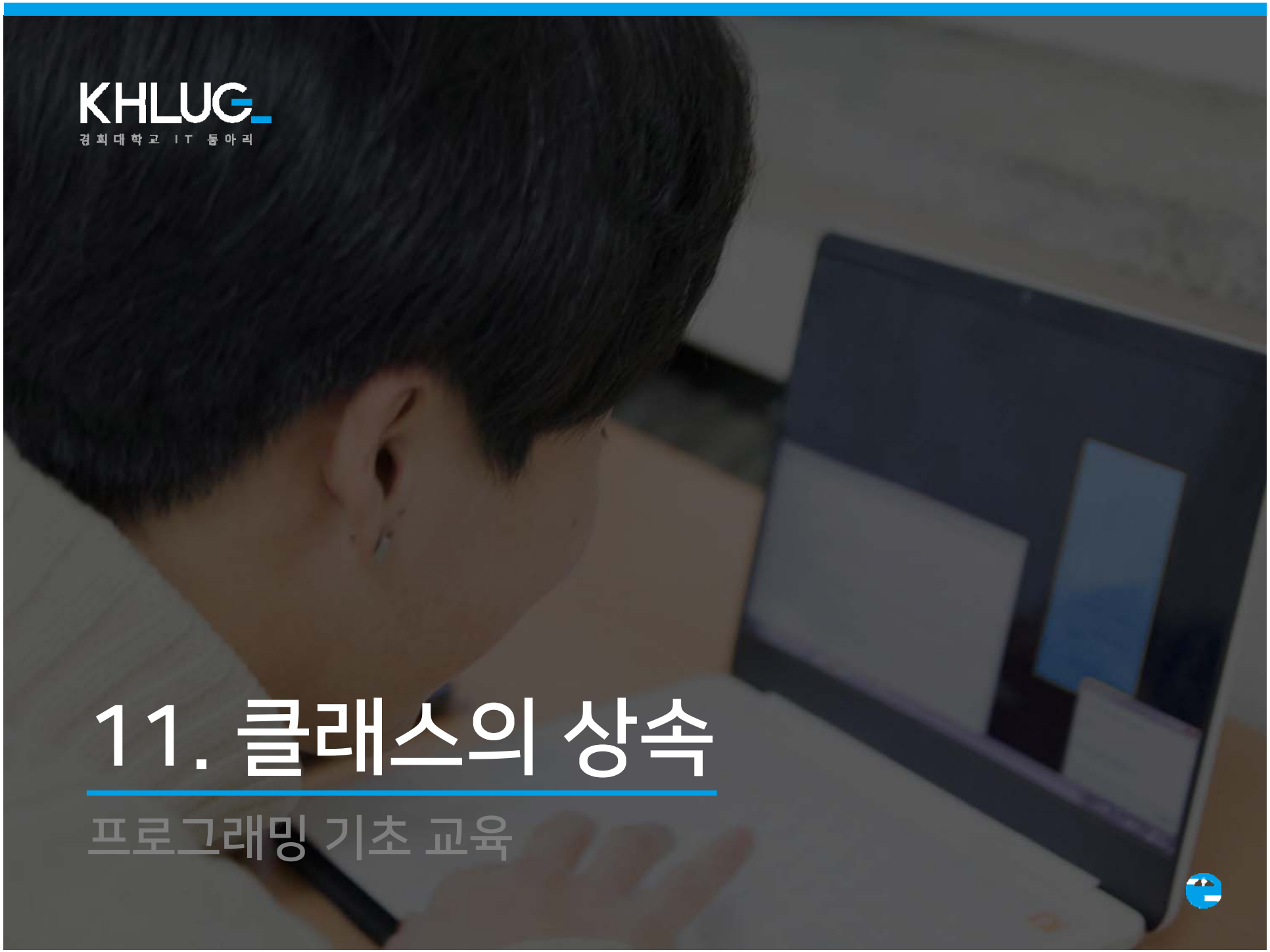
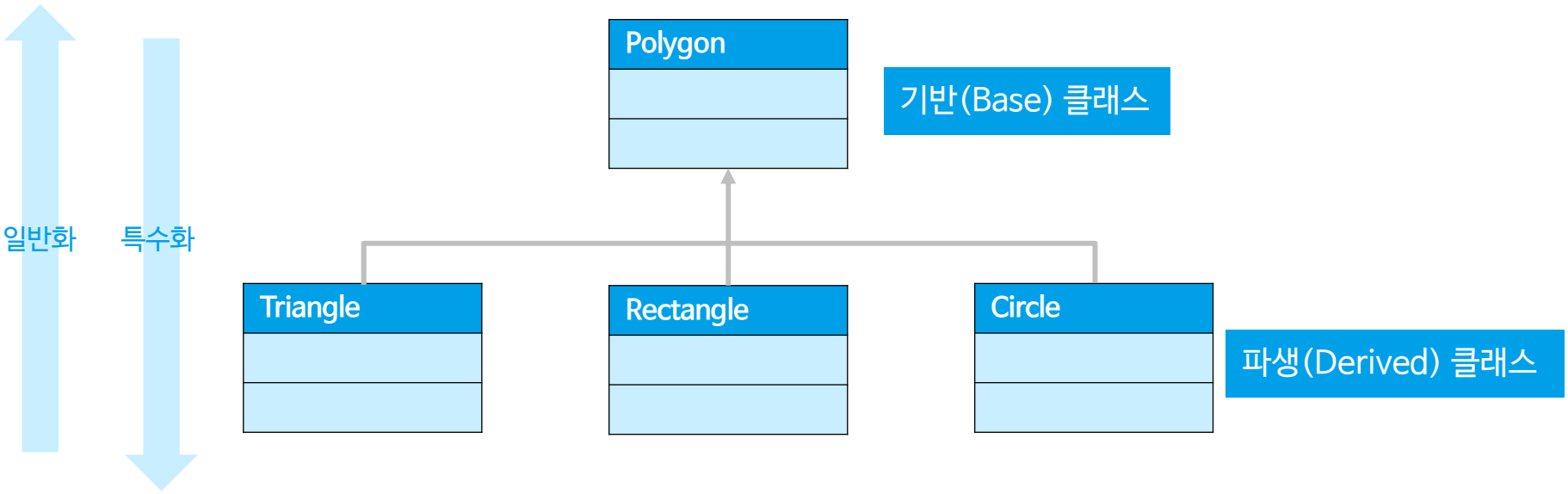


11. 클래스의 상속

프로그래밍 기초 교육

상속

- 클래스 간의 구조를 더욱 확장하여 기존 클래스의 특성을 가지면서 더욱 구체적인 멤버가 추가된 클래스를 만드는 데 상속(Inheritance)을 사용한다.
- is-a(~은 ~) 관계라고도 표현하는데, 예를 들면 사각형 is-a 도형(사각형은 도형)이다.



기반 클래스

- 비슷한 종류의 구체적인 객체들로 파생될 수 있어 상속을 해주는 클래스는 기반 클래스(Base Class) 또는 부모 클래스(Parent Class) 또는 상위 클래스(Super Class) 라고 한다.
- 기반 클래스에는 파생 클래스들이 갖는 공통적인 속성을 멤버로 만들어야 한다.

파생 클래스는 기반 클래스로부터 멤버들을 물려(상속) 받아 사용한다.

```
class Polygon {
protected:
    double area;
    double perimeter;

public:
    void printArea() const;
    void printPerimeter() const;
};

void Polygon::printArea() const {
    cout << "Area      : "
        << this->area << endl;
}

void Polygon::printPerimeter() const {
     cout << "Perimeter: "
        << this->perimeter<< endl;
}
```

protected 멤버는 상속 관계에 있는 객체에만 접근을 허용한다.

도형은 공통적으로 넓이(area)와 둘레(perimeter)를 가진다.

멤버 함수에 const를 붙이면 해당 함수 안에서 멤버 함수의 값이 변경되지 않는다.

파생 클래스

- 기반 클래스로부터 멤버를 상속 받는 클래스를 파생 클래스(Derived Class) 또는 자식 클래스(Child Class) 또는 하위 클래스(Sub Class) 라고 한다.
- 파생 클래스에는 다른 파생 클래스와 달리 해당 클래스만 가지고 있는 멤버를 만든다.

도형마다 가지고 있는 변도 다르고, 넓이와 둘레의 계산 방법도 다 다르다.

public으로 상속했다.

```
class Rectangle : public Polygon {
private:
    double sideA;
    double sideB;

    void calcArea();
    void calcPerimeter();

public:
    Rectangle(double, double);
};

Rectangle::Rectangle(double a, double b)
    : sideA(a), sideB(b) {
    this->calcArea(); this->calcRerimeter();
}

void Rectangle::calcArea() {
    this->area = this->sideA * this->sideB;
}

void Rectangle::calcPerimeter() {
    this->perimeter =
        (this->sideA + this->sideB) * 2;
}
```

Trangle 클래스는 부모 클래스 Polygon의 protected 멤버인 perimeter, area, printPerimeter(), printArea()를 사용할 수 있다.

접근자

상속될 때는 모든 멤버 변수와 멤버 함수가 상속되지만, 아래의 멤버 변수는 상속되지 않는다.

- 생성자
- 소멸자
- 멤버가 아닌 함수 (프렌드 함수 등)
- 할당된 연산자 (오버로딩된 연산자 등)
- 가상 함수

- 상속될 때 접근자를 정해주어 상속하게 되며, 이에 따라 기반 클래스로부터 파생 클래스에서 상속 받은 멤버의 접근 범위가 결정된다.

상속 종류	기반 클래스 멤버의 접근자	파생 클래스에서의 접근
private	private	상속되나 접근할 수 없음
	protected	private
	public	private
protected	private	상속되나 접근할 수 없음
	protected	protected
	public	protected
public	private	상속되나 접근할 수 없음
	protected	protected
	public	public

접근자 덮어씌우기

- 상속 받는 멤버의 접근자를 오버라이딩(Overidding, 덮어씌우기)할 수 있다.

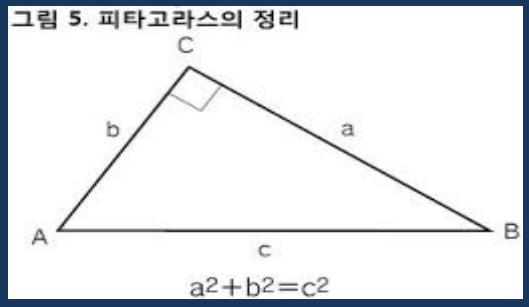
```
class A {
public:
    int member;
    void print();
};

class B : private A {
protected:
    using A::member;
}
```

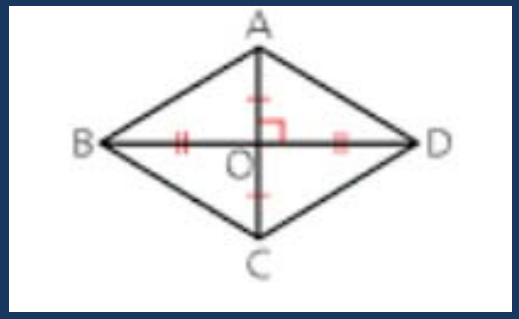
private로 상속 받은 클래스 A의 public 멤버는
클래스 B에서 private로 사용되어야 하나,
클래스 B에서 protected로 만들었기 때문에
클래스 B에서 protected로 사용할 수 있다.

클래스 상속 사용해보기

1. 앞의 슬라이드에서 사각형을 상속받아 원의 반지름을 상속받아 둘레와 넓이를 출력하시오(원의둘레= 반지름x2x원주율,원의지름 = 반지름x반지름x원주율, 원주율은 3.141592로한다)

2.앞의 슬라이드에서 사각형을 상속받아 직사각형을 만들었다. 앞의 슬라이드를 참조하여 사각형을 상속받는 마름모 클래스를 만들고 두 대각선을 입력받아 넓이와 둘레를 출력하시오.(마름모는 네 변의 길이가 모두 같은 사각형이다.)





파생 클래스의 생성자와 소멸자

- 파생 클래스가 인스턴스화 될 때, 기반 클래스의 생성자가 먼저 호출되고 파생 클래스의 생성자가 호출된다.
- 반대로 인스턴스가 사라질 때는 파생 클래스의 소멸자가 먼저 호출되고 기반 클래스의 소멸자가 호출된다.

```
Derived::Derived(parameters)
    : BaseClass (parameters) {
    //
}

Derived::Derived
    (const Derived& derivedObject)
    :Base(derivedObject) {
    //
}
```

기반 멤버 초기화 리스트 (base-member-initialization-list)를 통해 파생 클래스의 생성자로 들어온 매개변수를 기반 클래스의 생성자로 넘겨줄 수 있다.

복사 생성자는 아래와 같이 만든다. 파생 클래스의 인스턴스를 기반 클래스의 생성자로도 넘겨줘야 한다.

오버라이딩

- 클래스의 상속에서 오버라이딩(Overriding)은 기반 클래스의 멤버 함수를 파생 클래스에 재정의(Redefine)하는 것을 말한다.

이렇게 한 이름으로 여러 다른 모양을 사용할 수 있는 것은 클래스의 특징: 다형성(Polymorphism)을 보여준다.

오버로딩(Overloading)

- 함수의 이름이 같다.
- 그러나 매개변수 목록이 다르다.

오버라이딩(Overriding)

- 함수의 이름이 같다.
- 그리고 매개변수 목록도 같다.

Rectagle에 Polygon에 있는 printArea()와 같은 이름과 매개변수 목록을 가진 멤버 함수를 만들었지만, 내용은 다르다.

```
void Polygon::printArea() const {
    cout << "Area      : "
         << this->area << endl;
}

void Rectangle::printArea() const {
    cout << "Area(Rectangle): "
         << this->area << endl;
}
```

main()에서

```
Rectangle rectangle(4, 5);
rectangle.printArea();
```

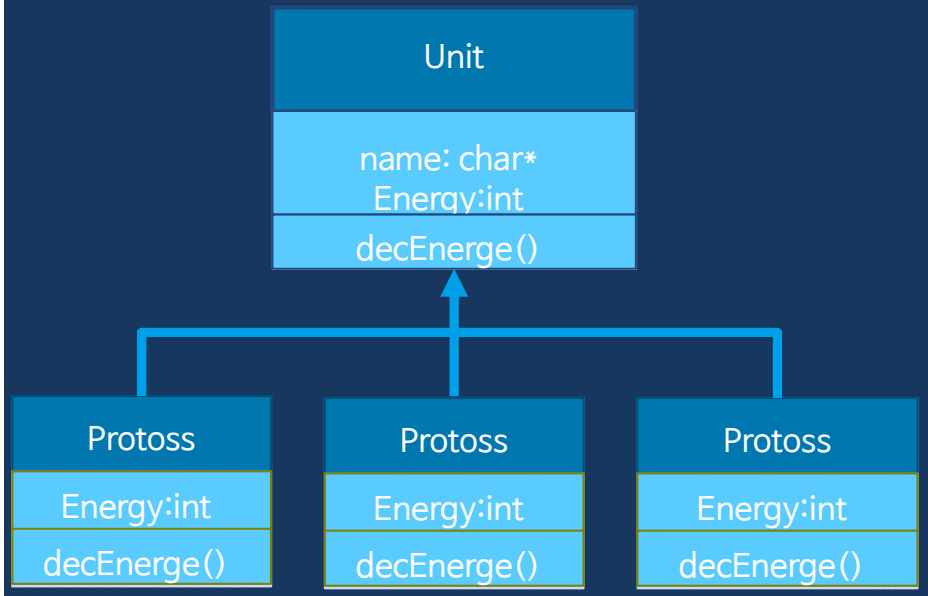
Polygon의 printArea()가 아니라 Rectangle의 printArea()가 호출된다.

```
rectangle.Polygon::printArea();
```

Polygon의 printArea()를 호출할 수도 있다.

오버라이딩 사용해보기

아래의 그림을 참조하여 스타크래프트 클래스를 적절히 상속시키고 decEnerge()를 오버로딩 하시오. (decEnerge()는 에너지를 감소시키는 함수이며 저그,테란 프로토스 각각 3,4,5를 감소시킨다. 저그의 초기에너지는 200,테란은 300,프로토스는 400이다.)




```
class Unit {
//Unit class 예시
protected:
char* name;
int energy;

public:

Unit(char*_name, int _energy)//생성자
{...}

~Unit(){...}//소멸자

void printEnerey() const{
cout << "현재에너지: " << energy
<< endl;
}

void decEnergy(){ //empty }
};
```

2017.05.18. 프로그래밍 기초 (2017-1)
with D.com