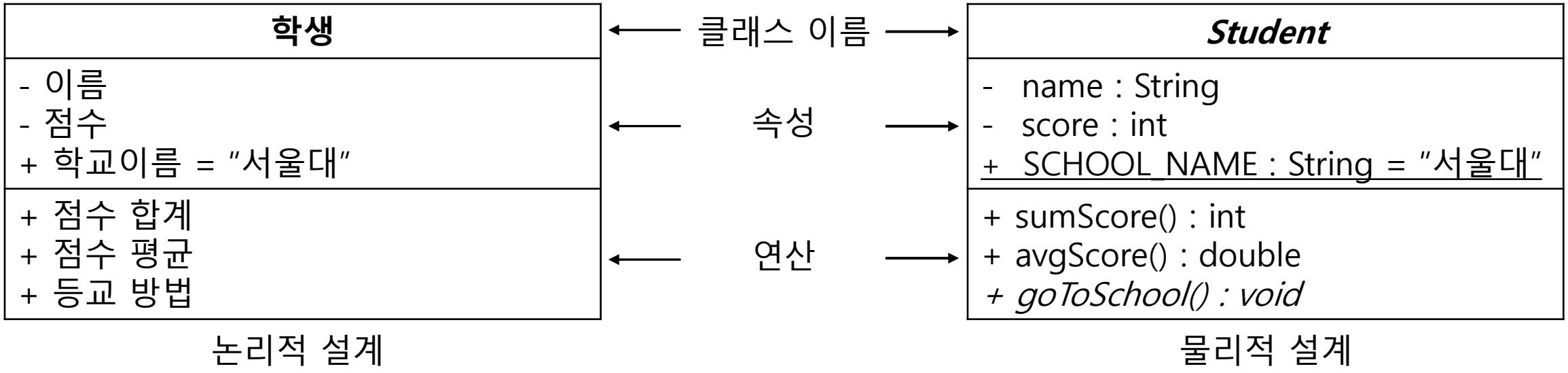
▶ 클래스 다이어그램

정적(구조) 다이어그램으로 UML모델링에서 가장 일반적으로 사용
시스템의 구조와 구조 간 상호 관계를 나타내며
시스템의 논리적 및 물리적 구성요소 설계 시 주로 활용

✓ 클래스의 표현



▶ 클래스 다이어그램

✓ 접근제한자

기호	예약어	적용 범위
+	public	전체
#	protected	같은 패키지 + 상속 관계
~	default	같은 패키지
-	private	같은 클래스

✓ 클래스 다이어그램 표기 방법

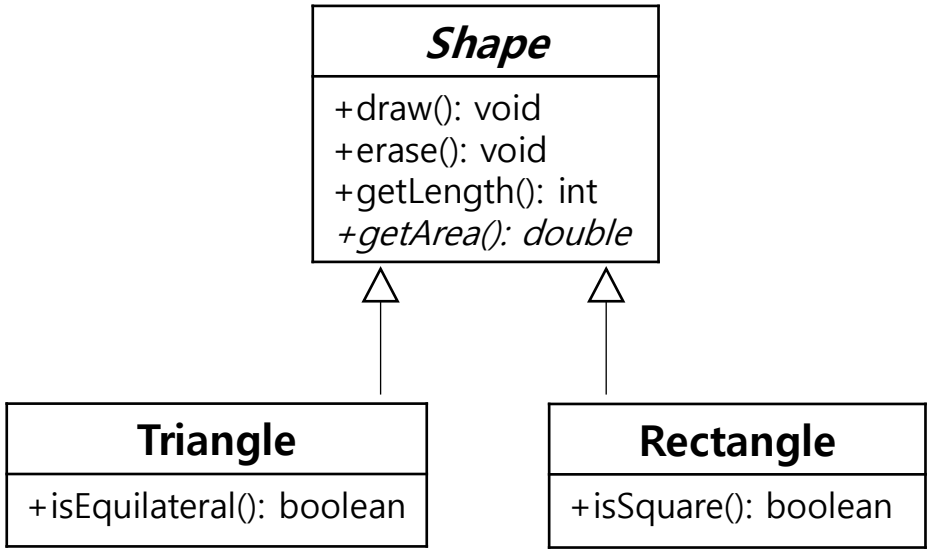
표현법	적용 범위	예약어
attribute / method (밑줄)	속성, 연산	static
FIELD (대문자)	속성	final
Class / *method* (기울임)	클래스 명, 연산	abstract

▶ 일반화 관계와 실체화(인터페이스 실현) 관계

✓ 일반화 관계

보다 **일반적인 클래스와 보다 구체적인 클래스 간의 관계**를 뜻하는 것으로

한 클래스(상위 클래스)가 다른 클래스(하위 클래스)보다 일반적인 개념/대상 임을 의미하는 관계

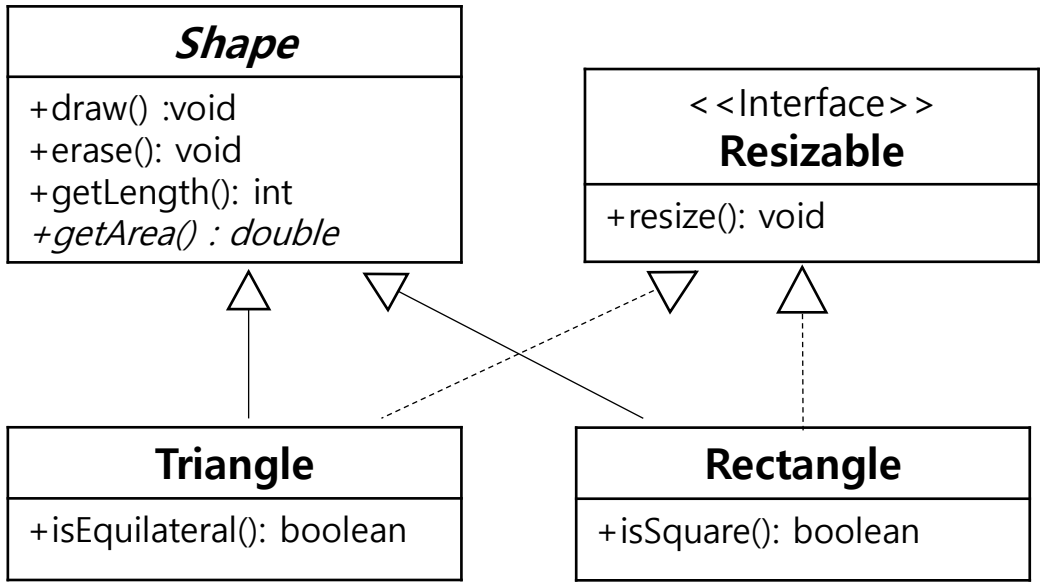



```
public abstract class Shape {
        public void draw() {…}
        public void erase() {…}
        public int getLength() {…}
        public abstract double getArea();
}

public class Triangle extends Shape {
        public boolean isEquilateral() {…}
        public double getArea() {…}
}
public class Rectangle extends Shape {
        public boolean isSquare() {…}
        public double getArea() {…}
}
```

일반화 관계와 실체화(인터페이스 실현) 관계

✓ 실체화(인터페이스 실현) 관계

인터페이스에 명세 된 기능을 클래스에 의해서 **구현한 관계** 의미



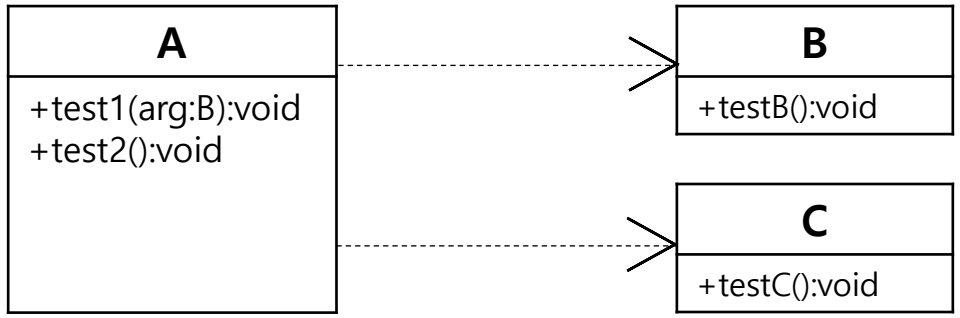

```
public interface Resizable {
        void resize();
}

public class Triangle extends Shape
                        implements Resizable  {
        public boolean isEquilateral() {…}
        public double getArea() {…}
        public void resize() {…}
}
public class Rectangle extends Shape
                        implements Resizable {
        public boolean isSquare() {…}
        public double getArea() {…}
        public void resize() {…}
}
```

의존 관계와 인터페이스 의존 관계

✓ 의존 관계

두 클래스의 **연산 간의 호출 관계**를 표현한 것으로 제공자의 변경이 이용자에 영향을 미칠 수 있음을 의미(제공자의 변경이 이용자의 변경 유발)

이용자는 의존 관계를 통해서 제공자의 연산을 호출할 수 있음

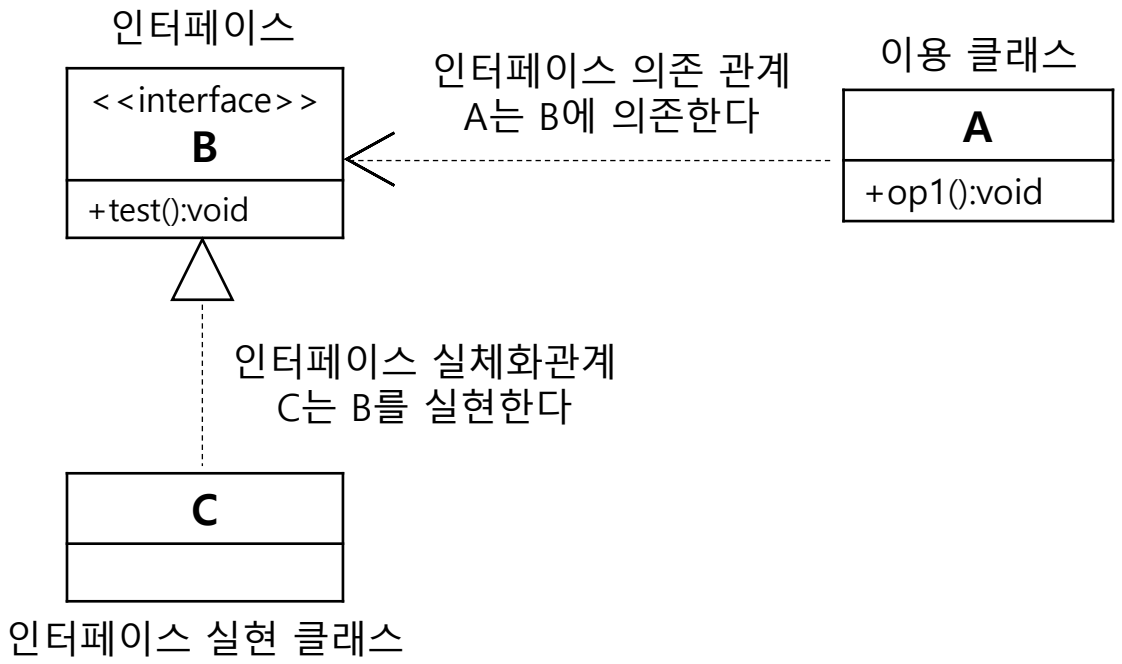



```
public class A{
        public void test1(B arg){
                arg.testB();
        }
        public void test2(){
                C = new C ();
                c.testC();
        }
}
```

의존 관계와 인터페이스 의존 관계

✓ 인터페이스 의존 관계

인터페이스와 인터페이스 이용자 간의 이용관계를 표현할 때 사용 될 수 있음




```
public interface B{
        void test();
}

public class C implements B{
        public void test() {…}
}

public class A{
        public void op1(){
                B b= new C();
                b.test();
        }
}
```

▶ 집합 관계와 합성 관계

✓ 집합 관계

부분 객체가 다수의 전체 객체에 의해 공유 될 수 있음

→ 전체 객체가 사라져도 부분 객체는 존재한다.

리모콘은 건전지로 구성된다
건전지는 리모콘의 부분이다
건전지는 다른 프로젝트에도 공유된다

✓ 합성 관계

부분 객체가 오직 하나의 전체 객체에 포함될 수 있음

→ 전체 객체가 사라지면 부분 객체도 사라진다.

집은 방으로 구성된다
방은 집의 부분이다