UML과 자바클래스

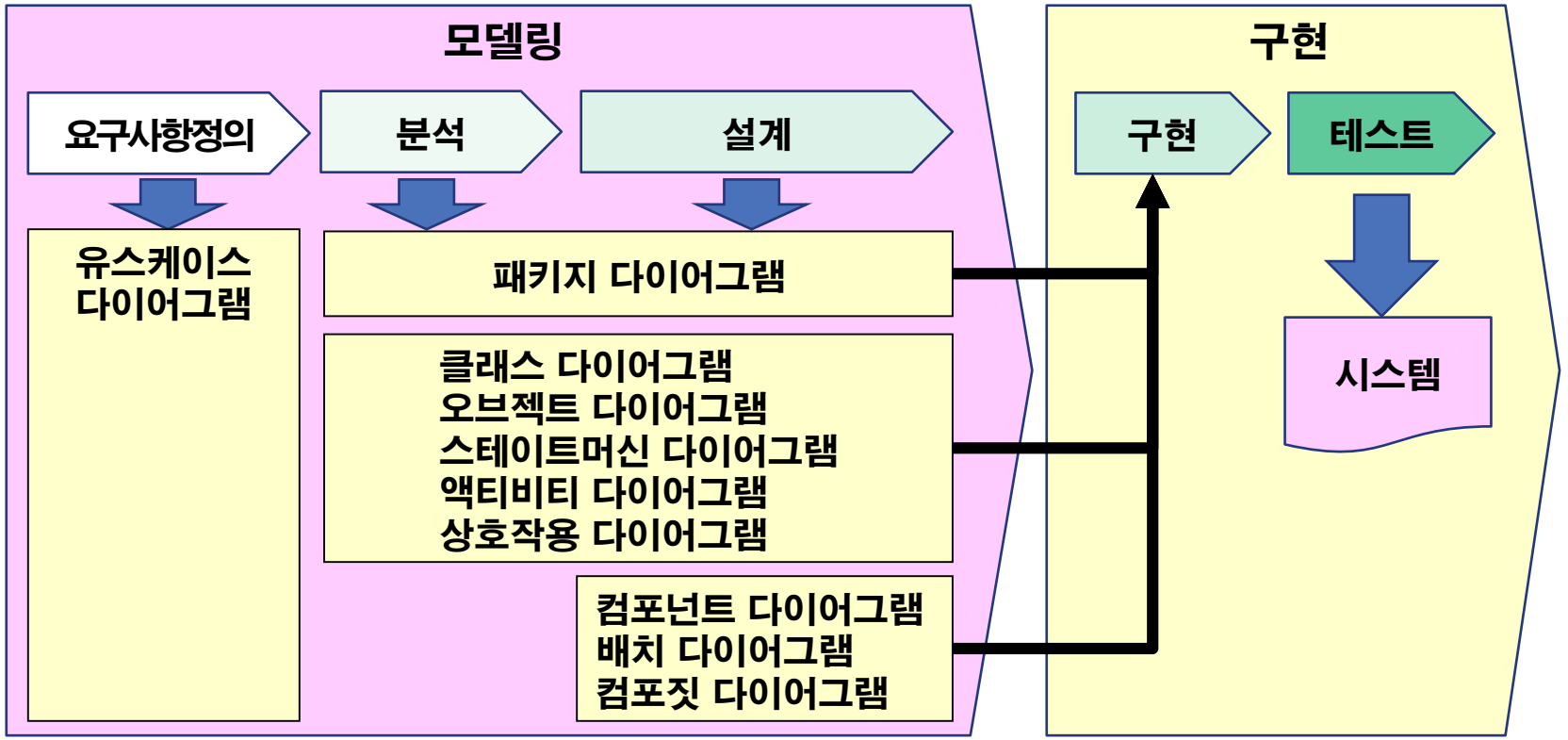
- UML(Unified Modeling Language)
- UML 작성도구
- 클래스 모델링
- 클래스 관계설정

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

UML

- Unified Modeling Language
  - 객체지향 소프트웨어 개발 방법론의 표준화된 모델링 언어
  - 시스템의 산출물을 명세하고, 생성하며, 문서화하기 위한 시각적인 언어 (OMG 2003)

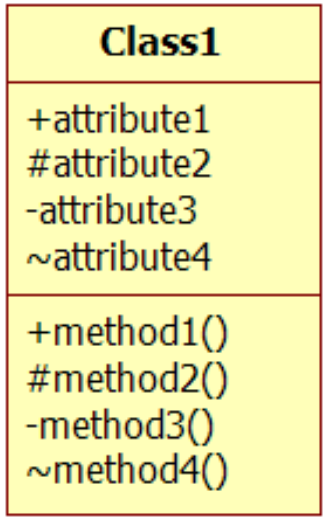
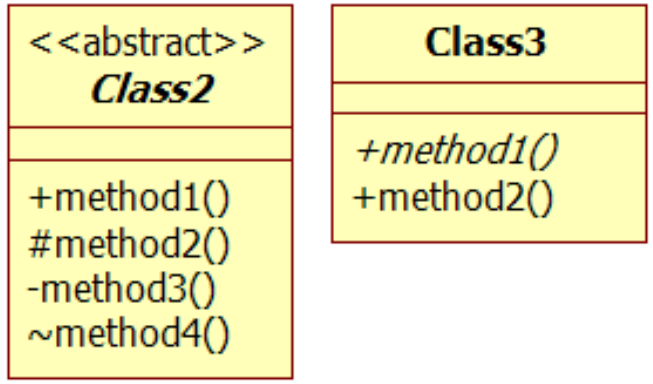
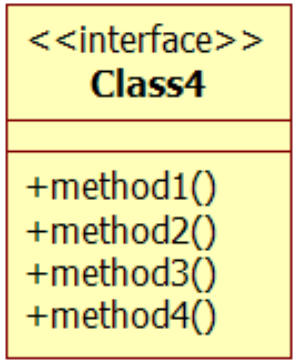
- UML 활용 소프트웨어 개발 과정



UML 작성도구

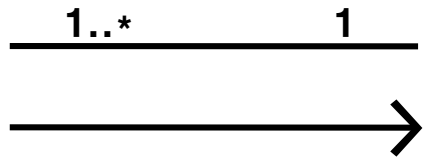
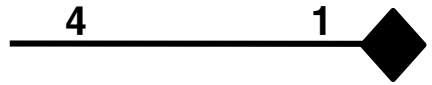
- Rational Rose
  - www.ibm.com/software/awdtools/developer/rose
  - 고성능이지만 가격이 비싼 대표적인 도구
- ArgoUML
  - argouml.tigris.org
  - 오픈 소스 소프트웨어 엔지니어링 도구
- Violet
  - violet.sourceforge.net
  - 공개 UML Editor
- StarUML
  - staruml.sourceforge.net/ko/
  - 국산 Win32 플랫폼에서 무료로 사용할 수 있는 UML/MDA 플랫폼

클래스 모델링

일반 클래스	추상클래스	인터페이스
		

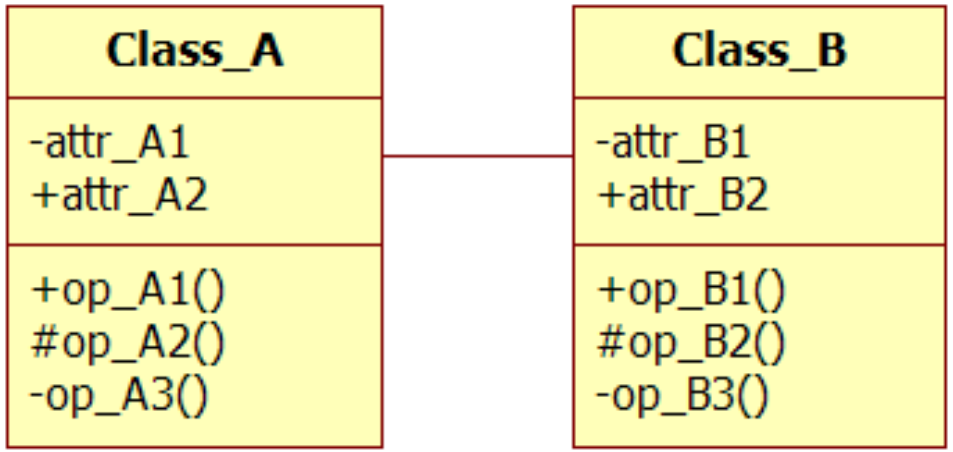
public class Class1
{
public int attribute1;
protected int attribute2;
private int attribute3;
int attribute3;
 	
public void method1(){}
protected void method2(){}
private void method3(){}
void method4(){}
}
	public abstract class Class2
	{
	public void method1(){}
	protected void method2(){}
	private void method3(){}
	void method4(){}
	}
	public class Class3
	{
	public abstract void method1(){}
	public void method4(){}
	}
		public interface Class4
		{
		public void method1();
		public void method2();
		public void method3();
		public void method4();
		}

클래스 관계 설정 (1)

종류	설명	UML 표시
연관관계 association	클래스가 서로 개념적으로 연결되는 관계 (기호, 역할, 다중성, 방향 표시) 예> 학사 = 학생, 학교, …	
집합관계 aggregation	하나의 클래스에 여러 개의 독립적인 클래스들이 구성되는 관계 예> 식탁 = 테이블+의자	
복합관계 composition	집합관계에 있는 클래스들이 하나의 클래스에 영구적인 요소로 분리될 수 없는 관계 예> 엔진 = 카브레터+피스톤+플러그+…	
일반화(상속)관계 generalization (=inheritance)	특정 클래스를 상속해서 새롭게 생성되는 어떤 클래스와의 관계 예> 자동차 = (버스, 트럭, 택시, …)	
의존관계 dependency	한 클래스가 또 다른 클래스를 사용하는 관계 (한 클래스 변경은 다른 클래스에 영향 미침) 예> 수업→교수, TV→리모컨	
실체화관계 realization	추상클래스나 인터페이스를 상속받아 자식클래스나 추상메소드를 구현하는 관계 (= interface inheritance)	

클래스 관계 설정 (2)

❑ 연관 관계 : Association

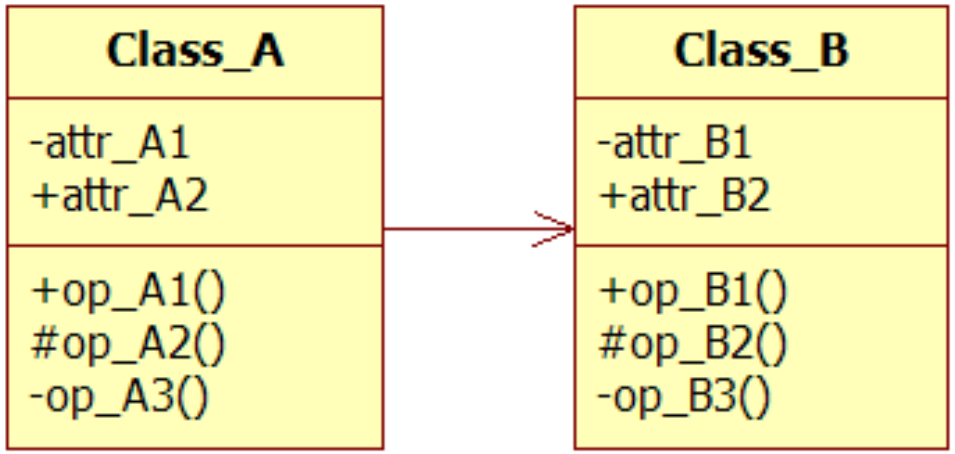



```
public class Class_A
{
    private int attr_A1;
    public int attr_A2;
    public Class_B classB;

    public void op_A1()  { }
    protected void op_A2() { }
    private void op_A3() { }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;
    public Class_A classA;

    public void op_B1() { }
    protected void op_B2() { }
    private void op_B3() { }
}
```

클래스 관계 설정 (3)

❑ 연관 관계 : Directed Association

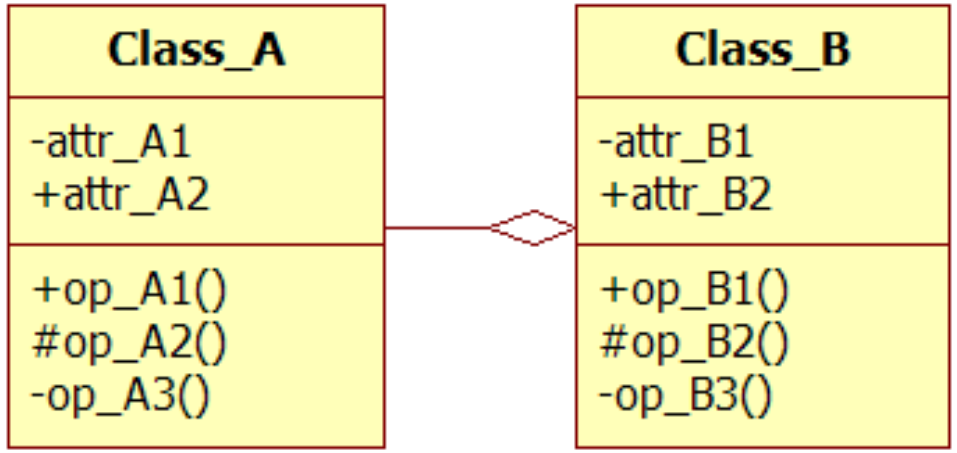



```
public class Class_A
{
    private int attr_A1;
    public int attr_A2;
    public Class_B classB;

    public void op_A1()  { }
    protected void op_A2() { }
    private void op_A3() { }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;

    public void op_B1() { }
    protected void op_B2() { }
    private void op_B3() { }
}
```

클래스 관계 설정 (4)

❑ 집합 관계 : Aggregation

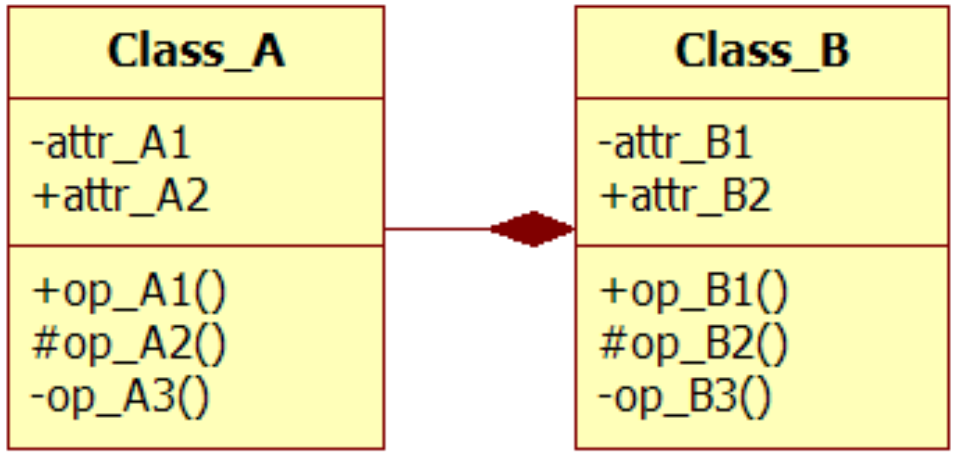



```
public class Class_A
{
    private int attr_A1;
    public int attr_A2;
    public Class_B classB;

    public void op_A1()  { }
    protected void op_A2() { }
    private void op_A3() { }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;
    public Class_A classA;

    public void op_B1() { }
    protected void op_B2() { }
    private void op_B3() { }
}
```

클래스 관계 설정 (5)

❑ 복합 관계 : Composition

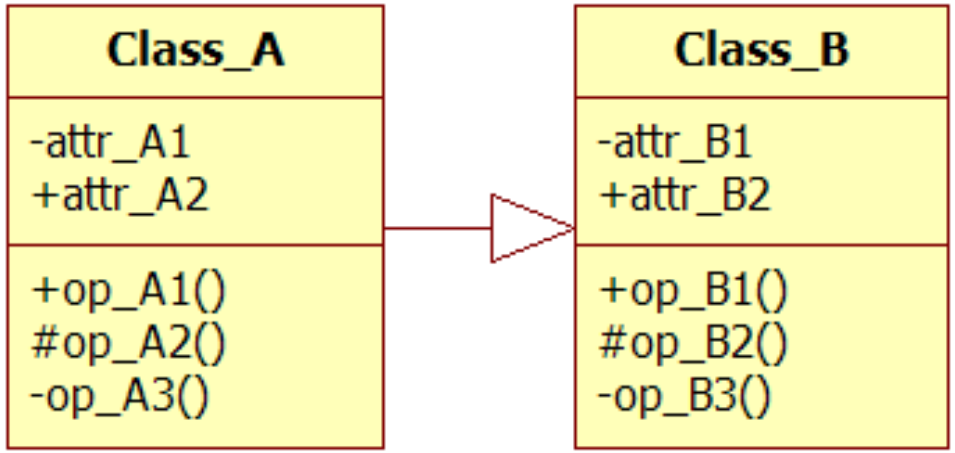



```
public class Class_A
{
    private int attr_A1;
    public int attr_A2;
    public Class_B classB;

    public void op_A1()  { }
    protected void op_A2() { }
    private void op_A3() { }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;
    public Class_A classA;

    public void op_B1() { }
    protected void op_B2() { }
    private void op_B3() { }
}
```

클래스 관계 설정 (6)

❑ 일반화(상속) 관계 : Generalization (=Inheritance)

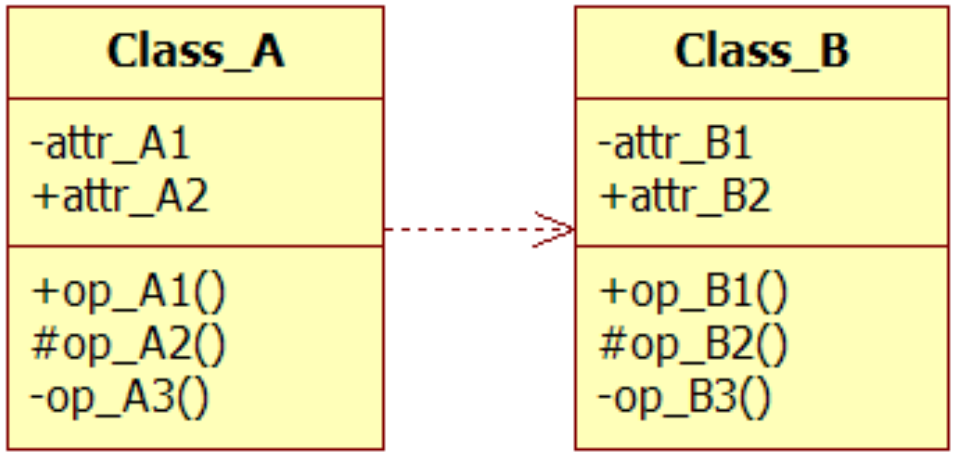



```
public class Class_A
                    extends Class_B
{
    private int attr_A1;
    public int attr_A2;

    public void op_A1()  {  }
    protected void op_A2() {  }
    private void op_A3() {  }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;

    public void op_B1() {  }
    protected void op_B2() {  }
    private void op_B3() {  }
}
```

클래스 관계 설정 (7)

❑ 의존 관계 : Dependency

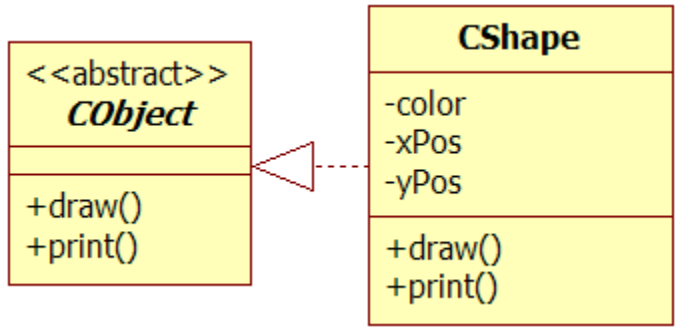


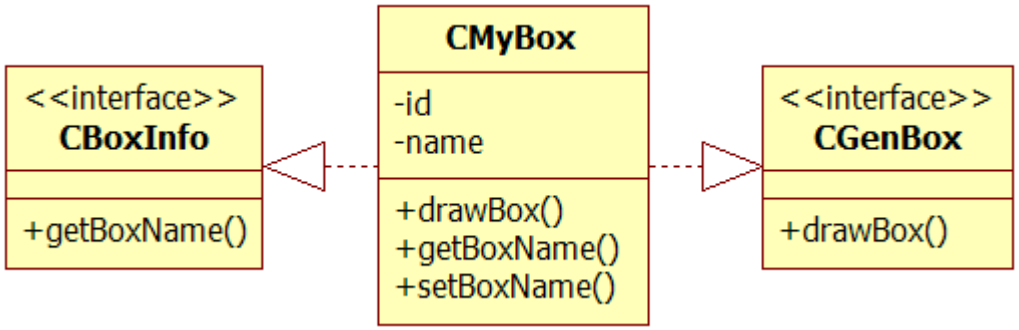

```
public class Class_A
{
    private int attr_A1;
    public int attr_A2;

    public void op_A1()  { }
    protected void op_A2() { }
    private void op_A3() {
        Class_B classB = new Class_B();
        classB.attr_B1 = 10;
        classB.op_B3();
    }
}
```

```
public class Class_B
{
    private int attr_B1;
    public int attr_B2;

    public void op_B1() { }
    protected void op_B2() { }
    private void op_B3() { }
}
```

클래스 관계 설정 (8)

❑ 실체화 관계 : Realization (=Interface Inheritance)






```
public abstract class CObject
{
    public void draw() { }
    public void print() { }
}
public class CShape
                implements CObject
{
    private Object color;
    private Object xPos;
    private Object yPos;
    public void draw() { }
    public void print() { }
}
```

```
public interface CBoxInfo {
    public void getBoxName();
}
public interface CGenBox {
    public void drawBox();
}
public class CMyBox
            implements CBoxInfo, CGenBox {
    private Object id;
    private Object name;
    public void drawBox() { }
    public void getBoxName() { }
    public void setBoxName() { }
}
```