Topic 12
Inheritance – More Topics

- Types of Inheritance
- Multiple Inheritance
- Virtual Abstract class
- Inheritance - Guidelines

Inheritance Types

- What will the visibility of inherited members ?
- For example, is a public inherited member function still public ?
- It depends on the type of inheritance.
- There are three types of inheritance
  - Private inheritance
  - Public inheritance
  - Protected inheritance

Private Inheritance

- All the inherited members become private.

```
class Person {
  private:
    string name ;
  protected:
    string address ;
  public:
    void print() ;
} ;
```

Person	Student
private	private
protected	private
public	private

```
class Student: private Person {
  private:
  public:
} ;
```

```
void f() {
  Student student ;
  student.name ;        // error
  student.address ;     // error
  student.print() ;     // error
}
```

Public Inheritance

- The visibility of each inherited members remain the same.

```
class Person {
  private:
    string name ;
  protected:
    string address ;
  public:
    void print() ;
} ;
```

Person	Student
private	private
protected	protected
public	public

```
class Student: public Person {
  private:

  public:
} ;
```

```
void f() {
  Student student ;
  student.name ;        // error
  student.address ;     // error
  student.print() ;     // ok
}
```

Protected Inheritance

- All the protected/public inherited members become protected.

```
class Person {
  private:
    string name ;
  protected:
    string address ;
  public:
    void print() ;
} ;
```

Person	Student
private	private
protected	protected
public	protected

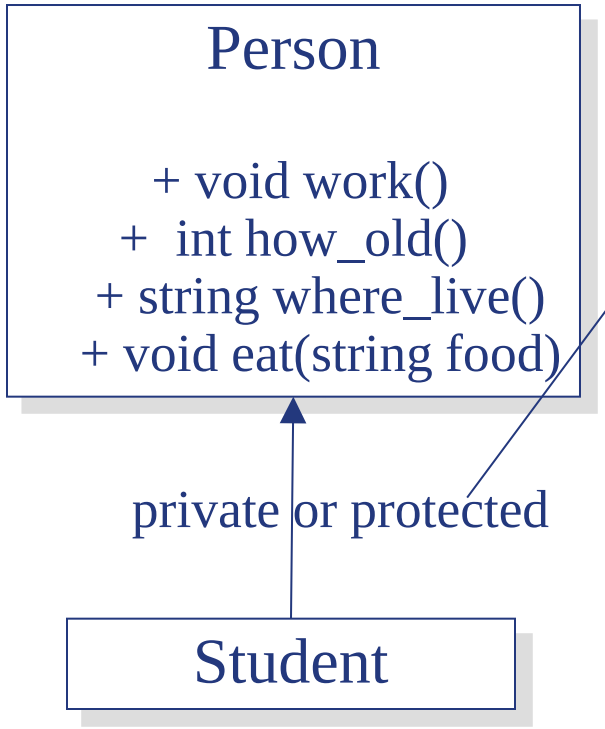
```
class Student: protected Person {
  private:

  public:
} ;
```

```
void f() {
  Student student ;
  student.name ;        // error
  student.address ;     // error
  student.print() ;     // error
}
```

Non-public Inheritance

- A subclass cannot include the interface of a superclass.
- This violates the principle of "IS-A" relationship

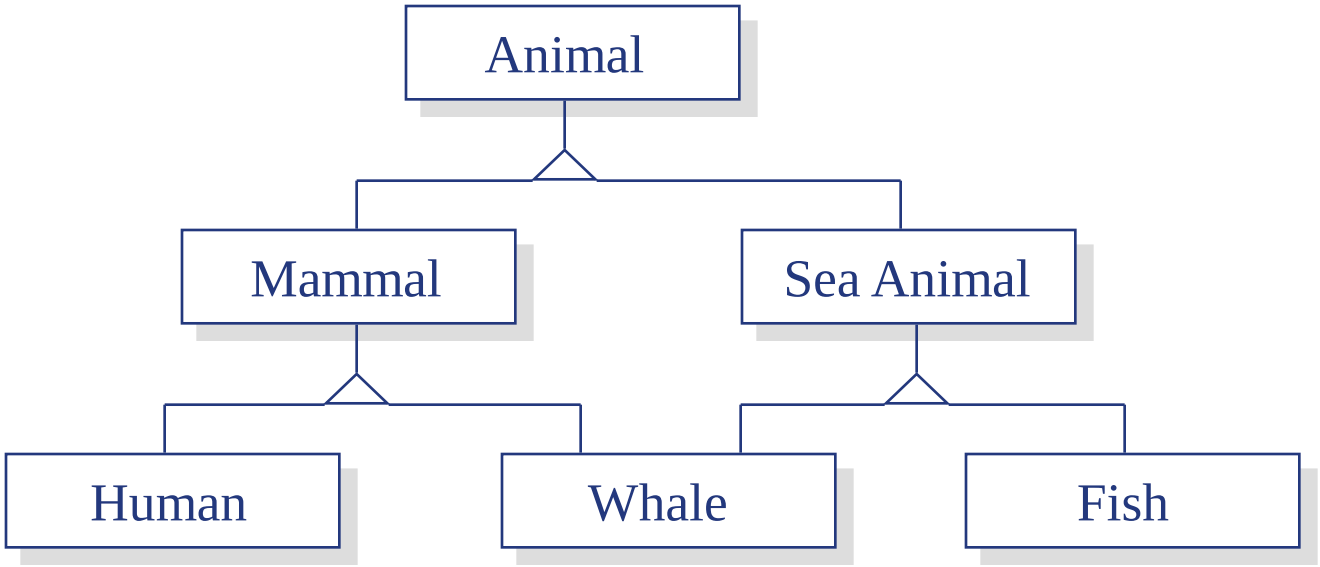


the public interface of class Person is excluded in class Student

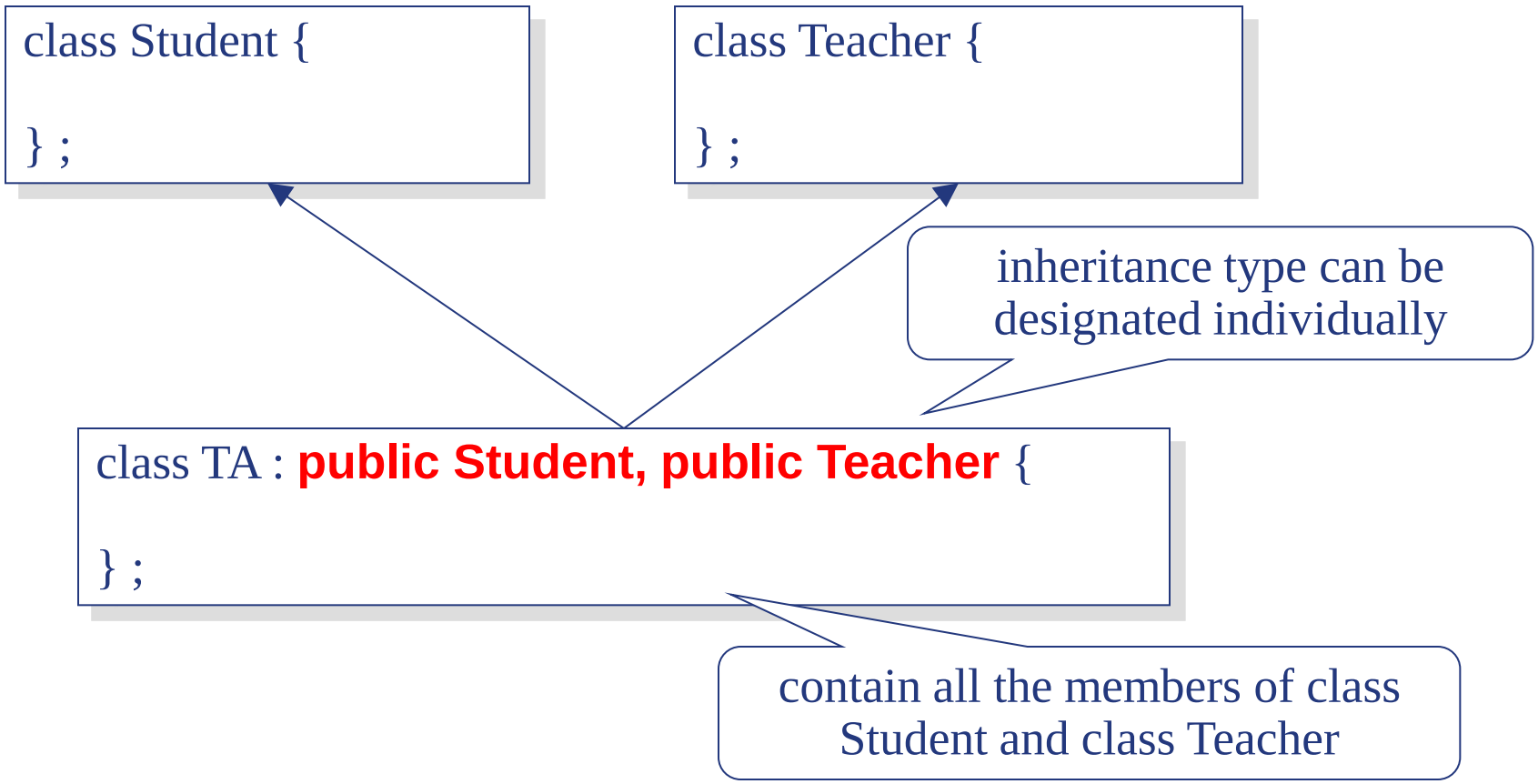
```
void f() {
  Student student ;
  student.work() ;              // error
  student.how_old() ;           // error
  student.where_live() ;        // error
  student.eat("Pizza") ;        // error
}
```

Multiple Inheritance

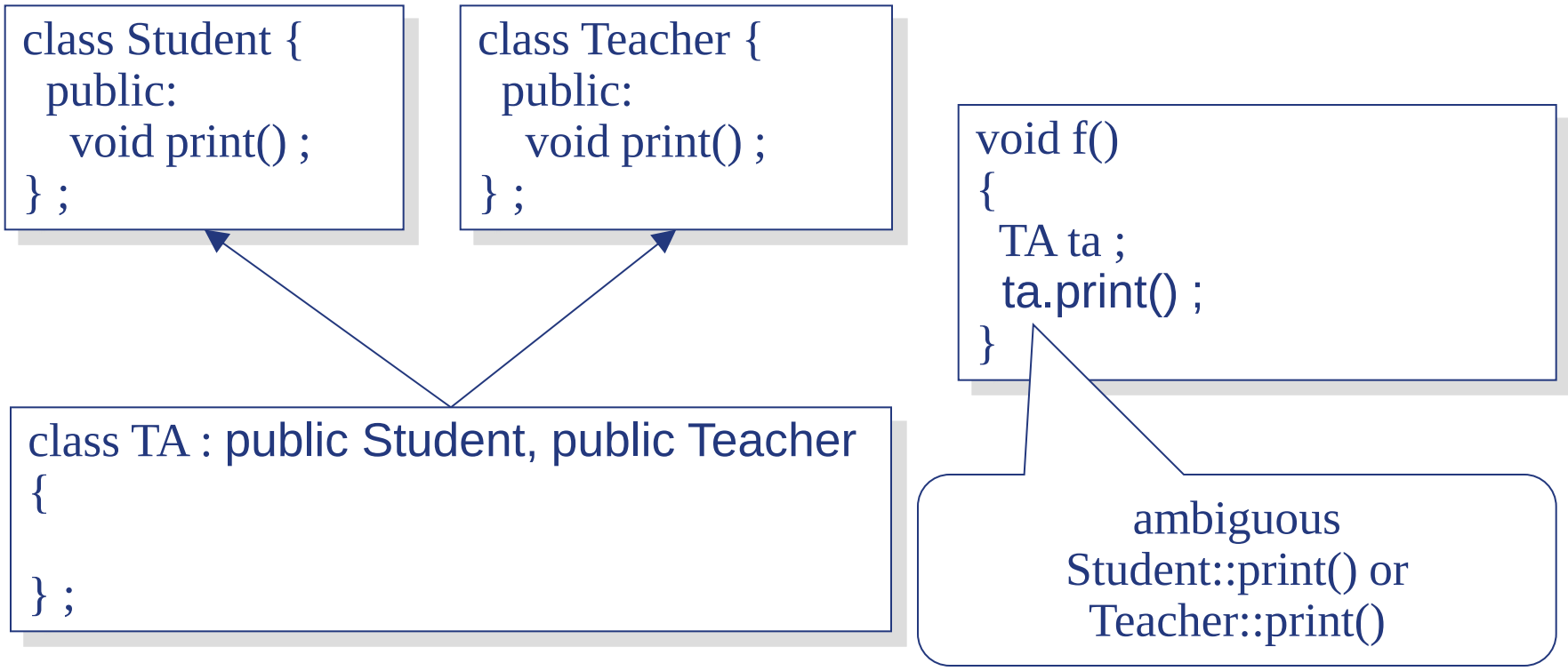
- Class can have more than one superclass and to inherit features from all parents



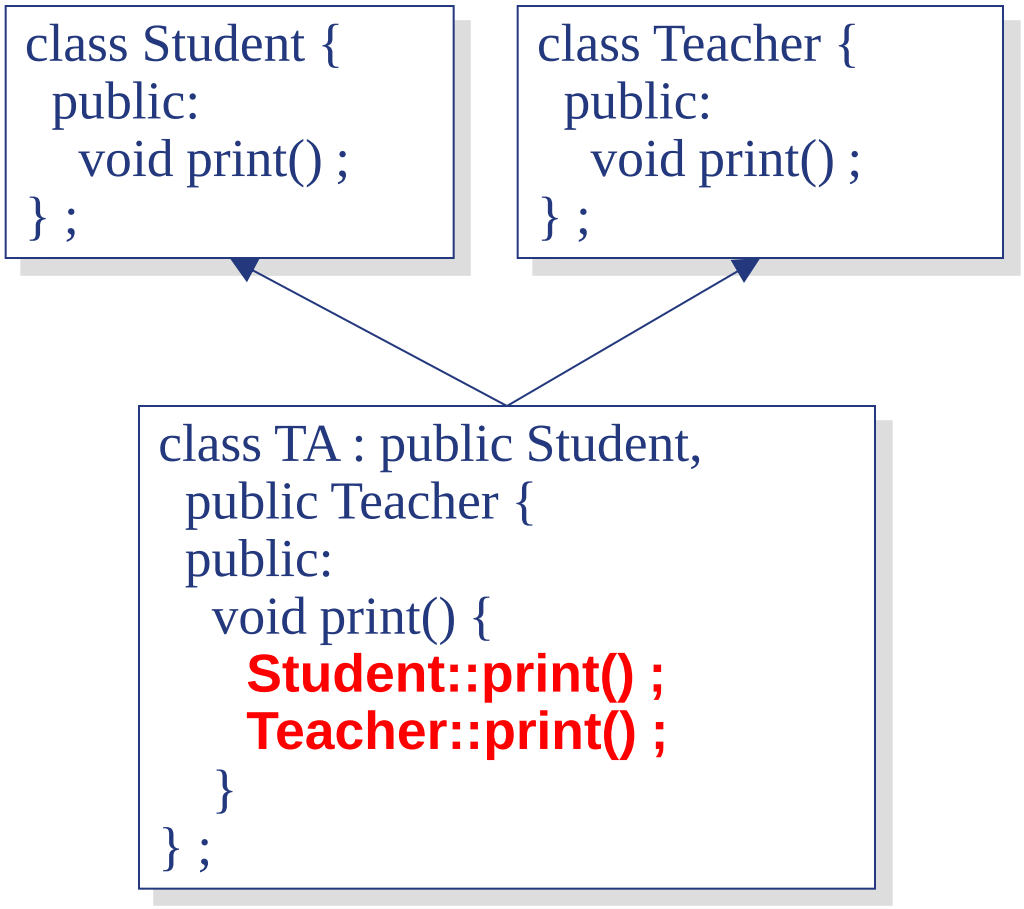
Multiple Inheritance in C++



Conflict Between Inherited Members



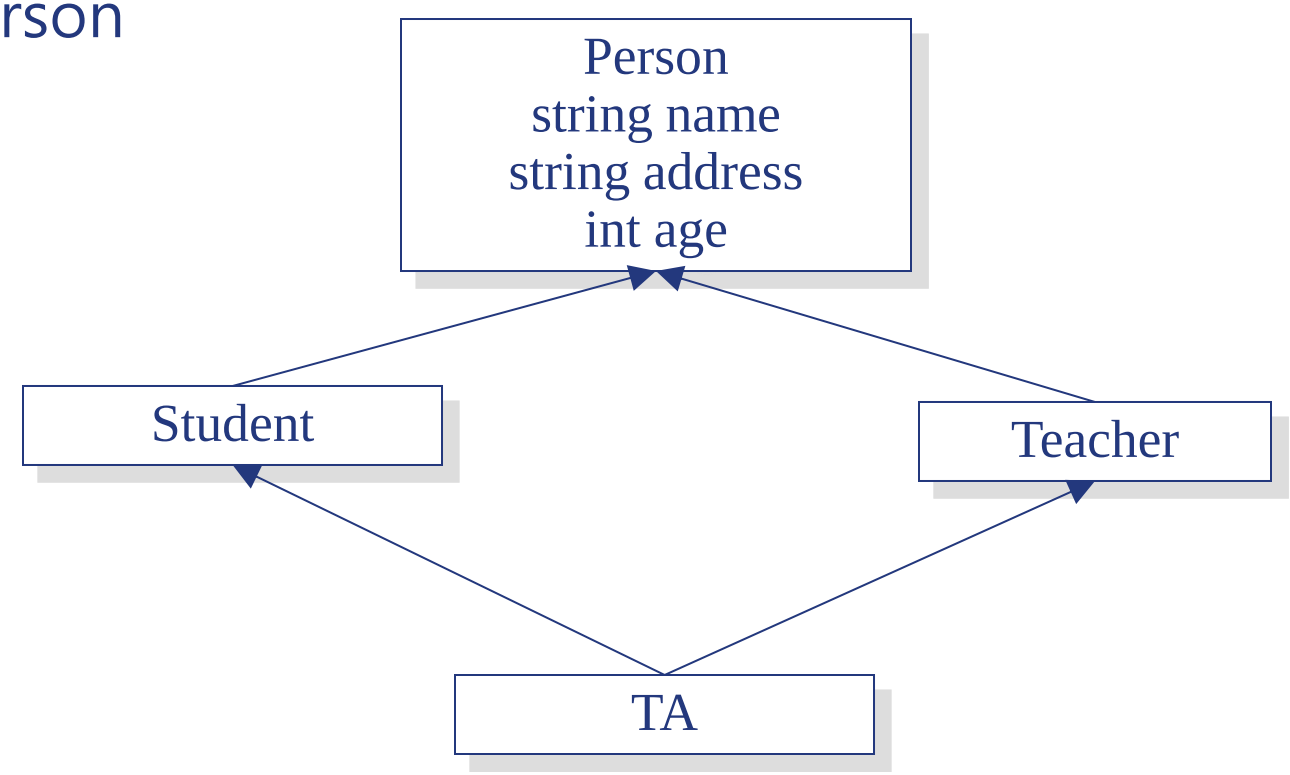
Class Qualification By Scope Operator



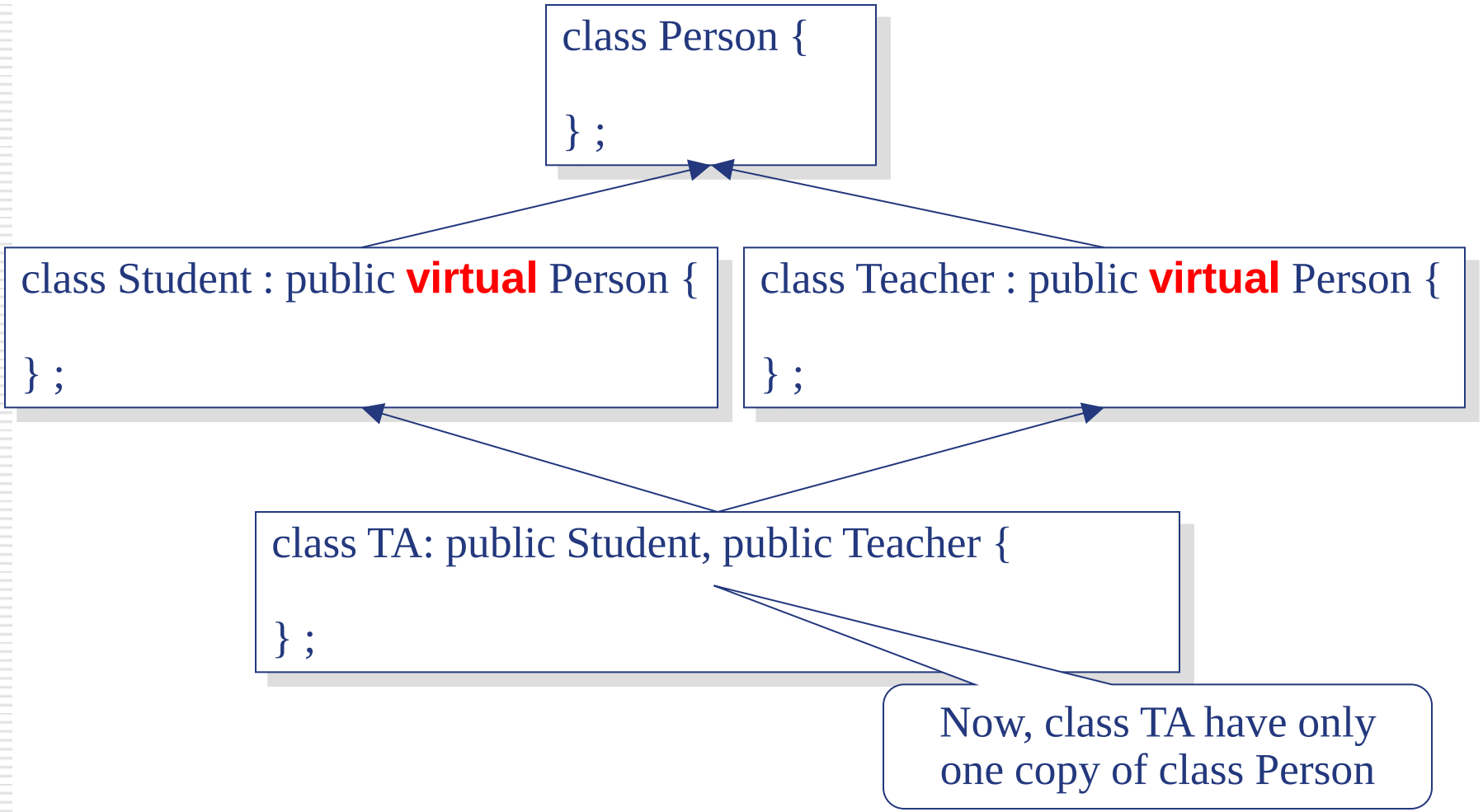

```
void f()
{
  TA ta ;
  ta.Student::print() ;
  ta.Teacher::print() ;
  ta.print() ;
}
```

Virtual Base Class

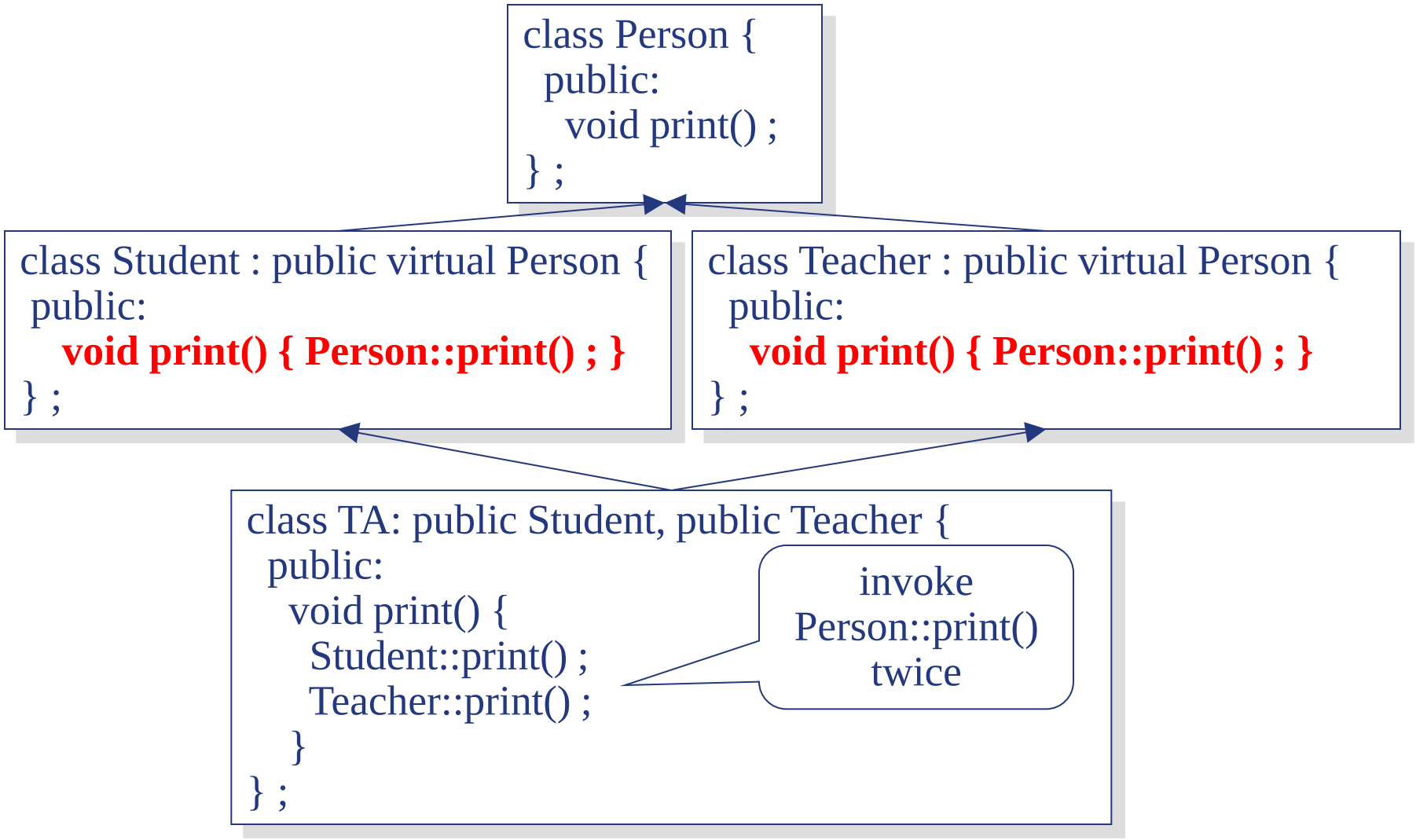
- Class TA contains two copies of members of class Person



Virtual Base Class



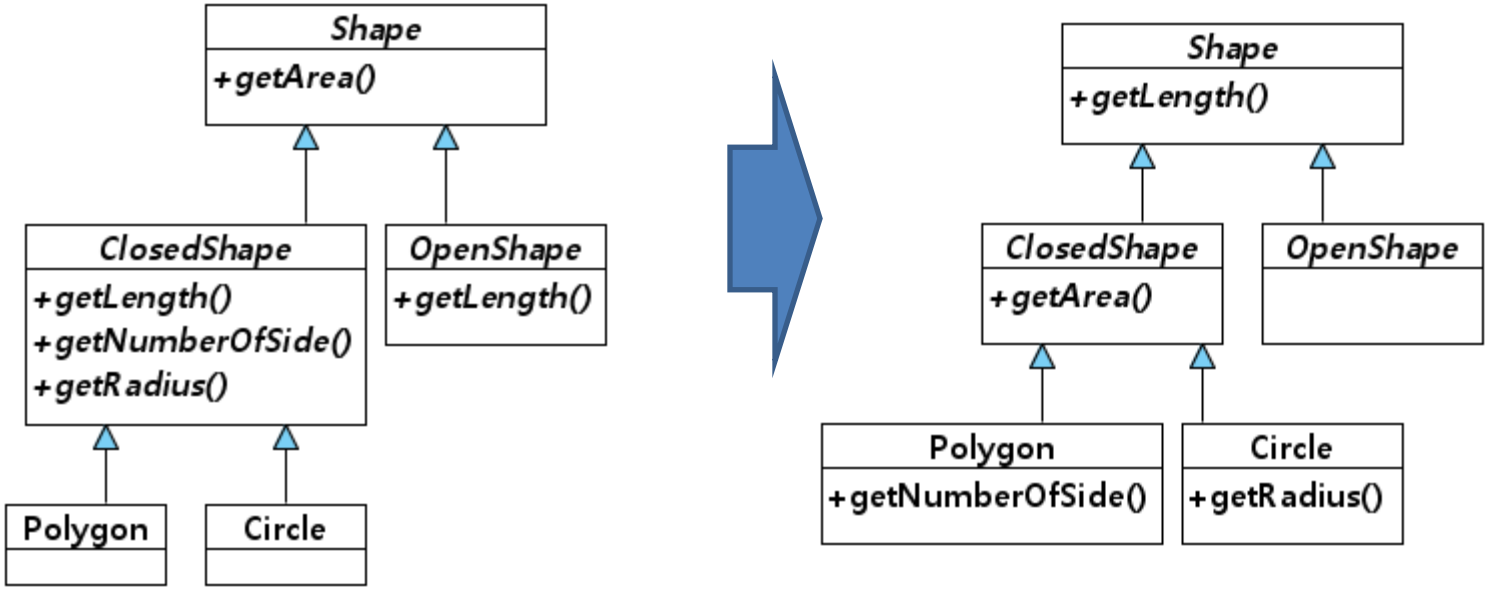
Virtual Base Class



Inheritance - Guidelines

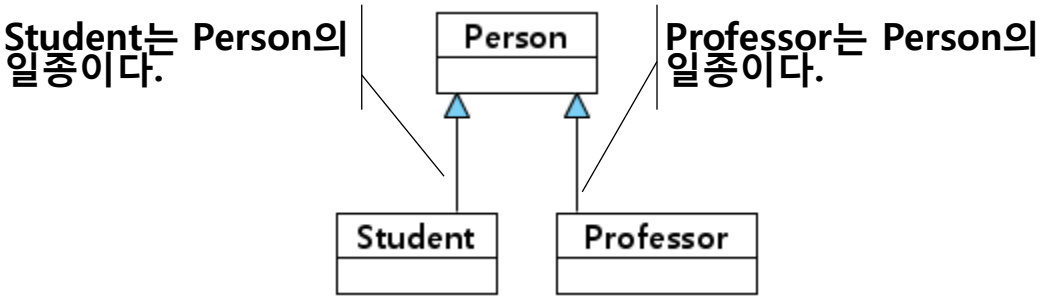
상위 클래스의 모든 멤버는 모든 하위 클래스에게 의미가 있어야 한다

- 모든 하위 클래스에게 동일하게 상위 클래스의 속성과 연산이 상속된다.
- 일부의 속성과 연산이 일부의 하위 클래스에게만 의미 있게 적용된다면 클래스 다이어그램의 수정이 필요하다

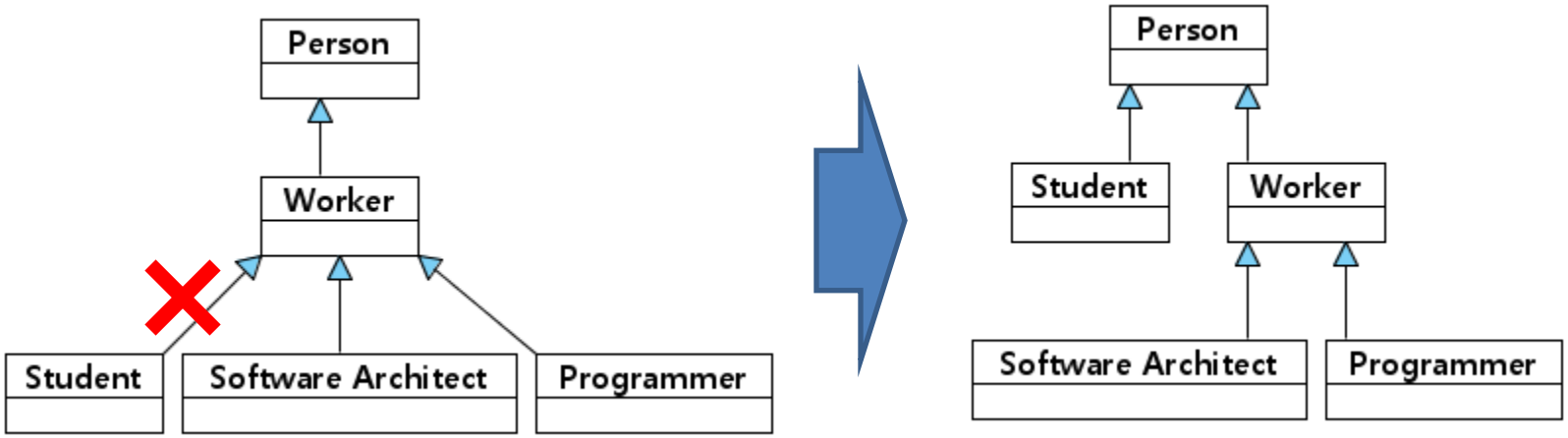


하위 클래스와 상위 클래스는 Is-a가 만족되어야 한다.

- 다형성 등의 특징을 바탕으로 시스템의 확장성을 제공하기 위해서는 상속 관계는 반드시 일반화 관계를 준수해야 한다.

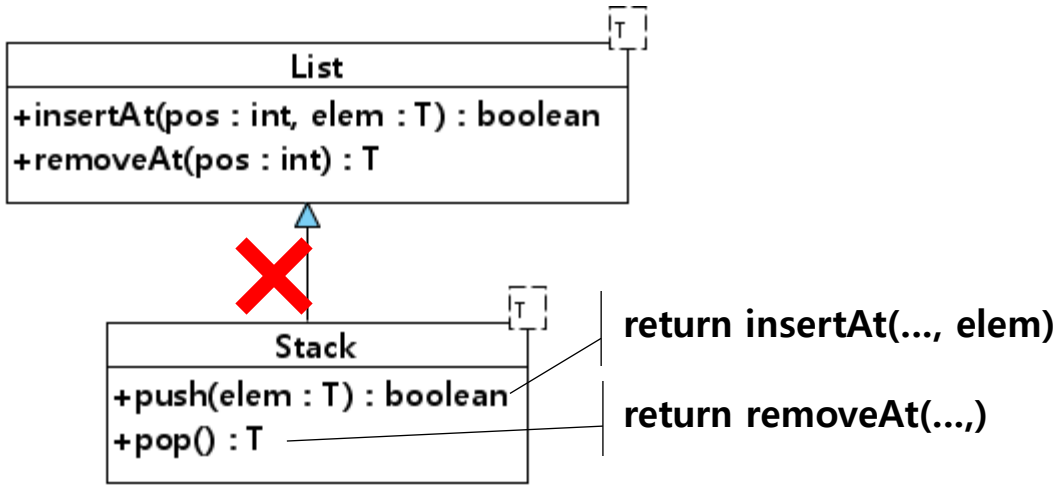


하위 클래스와 상위 클래스는 Is-a가 만족되어야 한다.



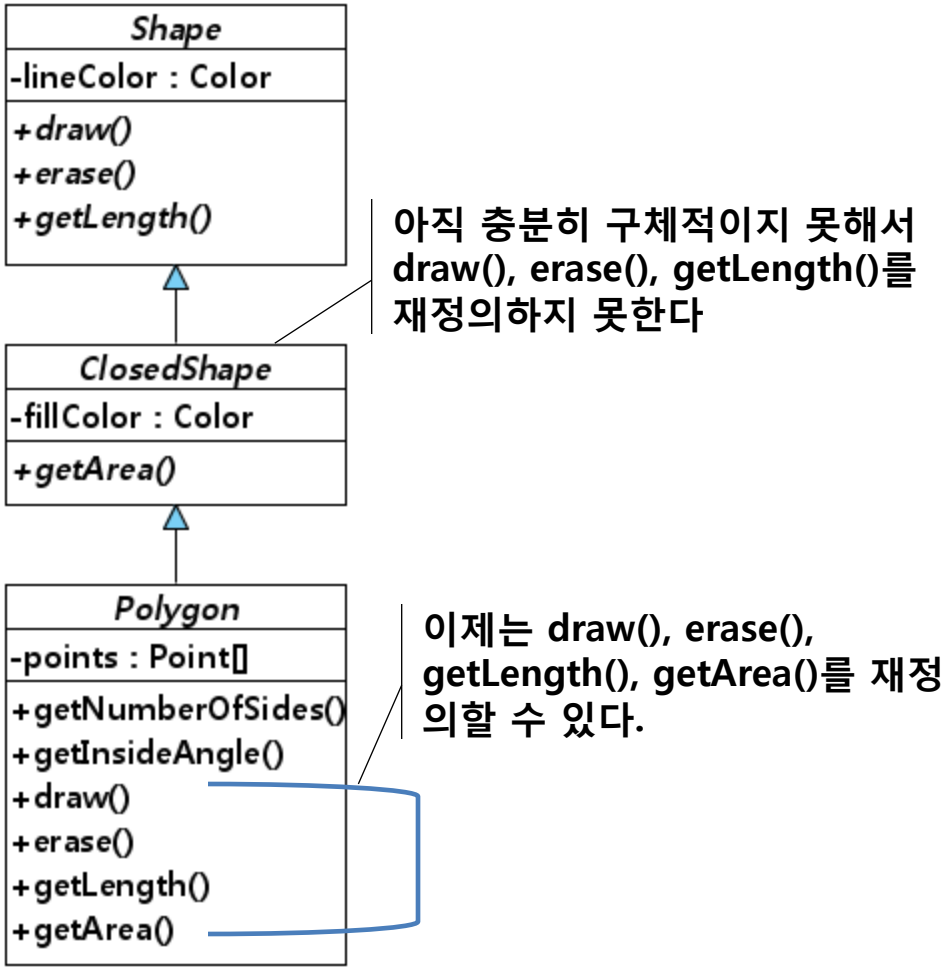
하위 클래스와 상위 클래스는 Is-a가 만족되어야 한다.

- Stack은 List의 일종이 아니다.



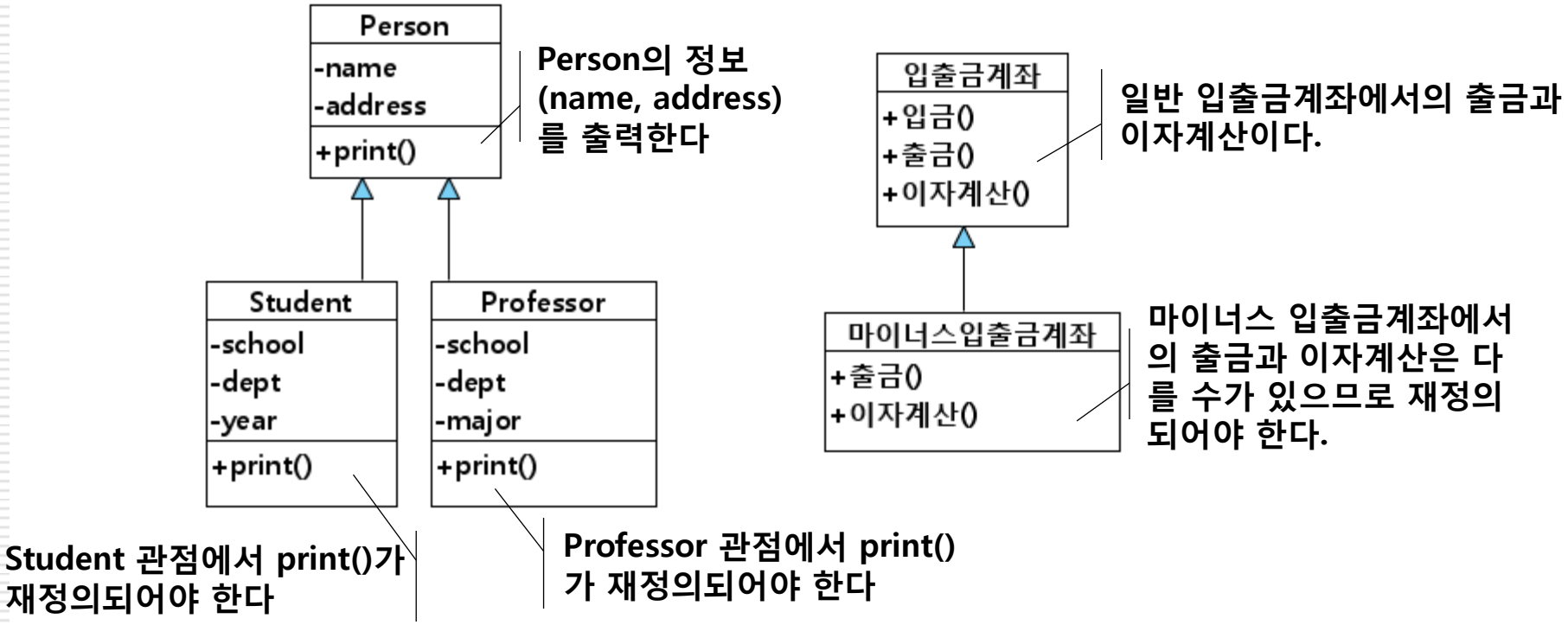
상속 받은 연산은 반드시 하위 클래스의 관점에서 정확해야 한다

- 하위 클래스에서는 상위 클래스로부터 물려 받은 연산을 필요에 따라서 재정의를 할 수 있다.
- 추상 연산의 재정의: 상위 클래스로부터 물려 받은 추상 연산이 하위 클래스에서는 구현이 가능하다면 추상 연산을 재정의할 수가 있다.



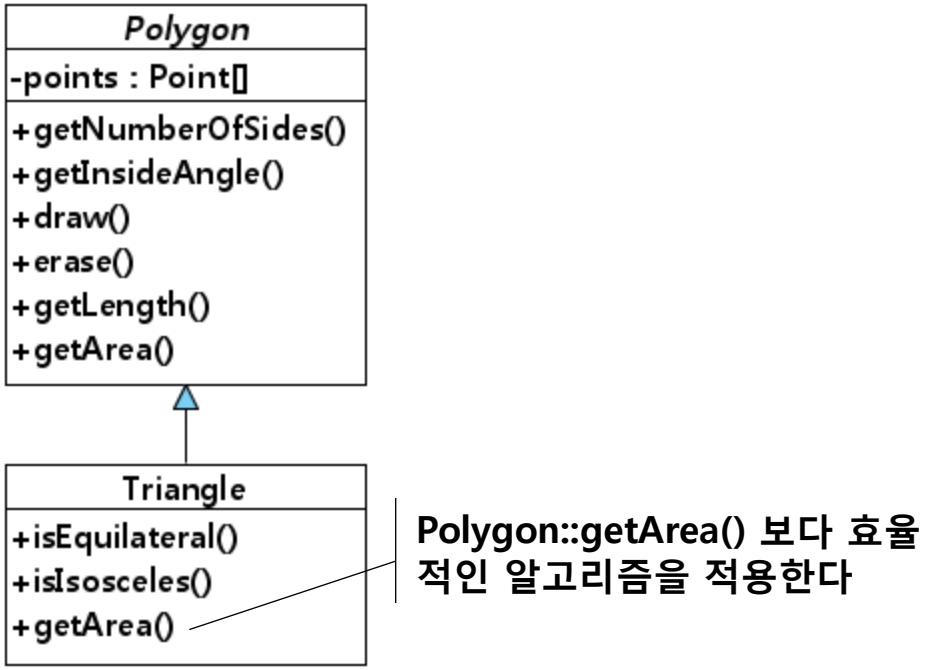
상속 받은 연산은 반드시 하위 클래스의 관점에서 정확해야 한다

- 기능적 재정의: 상위 클래스로부터 물려 받은 연산이 하위 클래스 관점에서는 기능적으로 부적절할 수가 있다.



상속 받은 연산은 반드시 하위 클래스의 관점에서 정확해야 한다

- 비기능적 재정의: 연산의 기능적인 측면에서는 동일하지만 수행 속도 또는 메모리 사용 등의 측면에서 상이한 구현이 필요한 경우에는 하위 클래스에서 재정의를 할 수도 있다

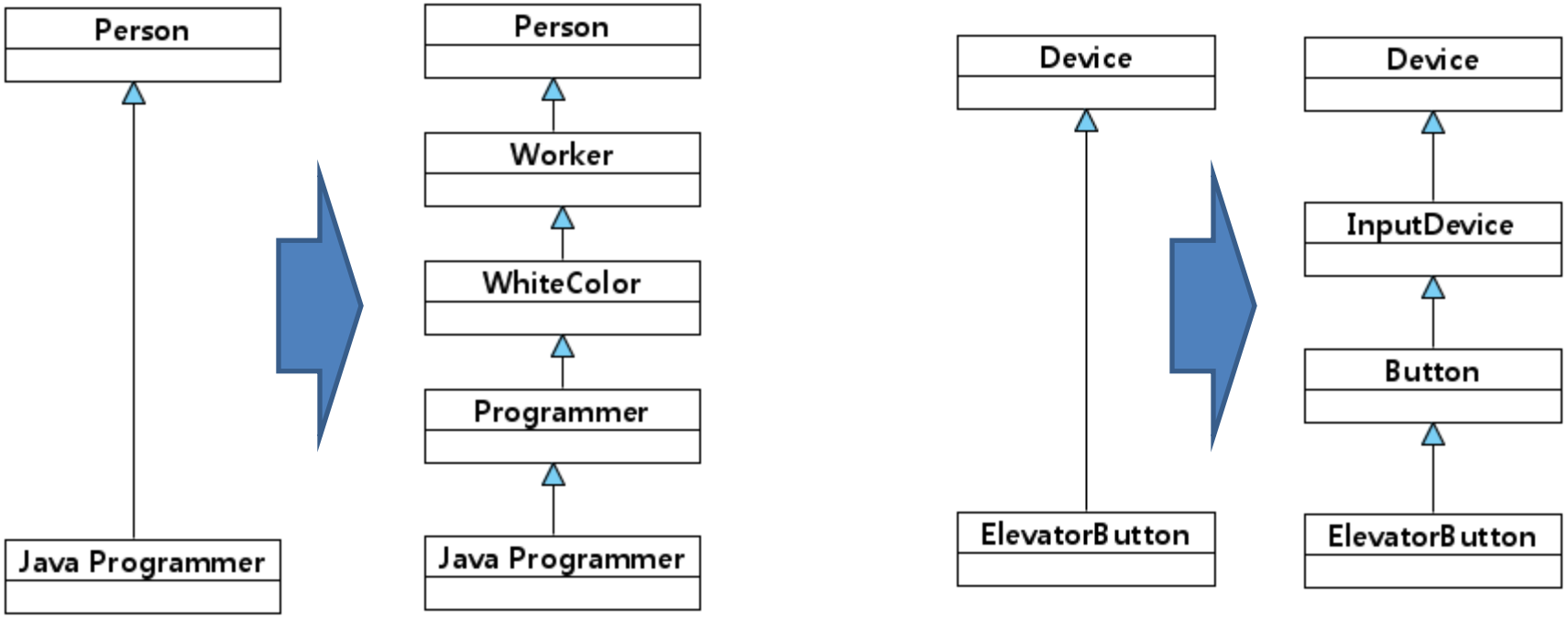


일반화 계층 구조는 균형적이어야 한다

- **적절한 계층 구조는 재사용성과 확장성이 높은 객체지향 시스템을 개발하는 데 큰 도움을 줄 수 있다**
- **상위 클래스와 하위 클래스 간의 의미적 차이가 적절해야 한다.** 상위 클래스와 하위 클래스 간에 의미적 차이가 너무 크면 그 상이에 존재하는 개념/대상에 해당되는 클래스들을 추가할 수 있다.
- **형제 클래스들은 동등한 수준의 개념을 의미해야 한다.** 형제 클래스들은 개념적으로 동등한 수준이어야 한다. 즉 비교가 가능해야 한다.

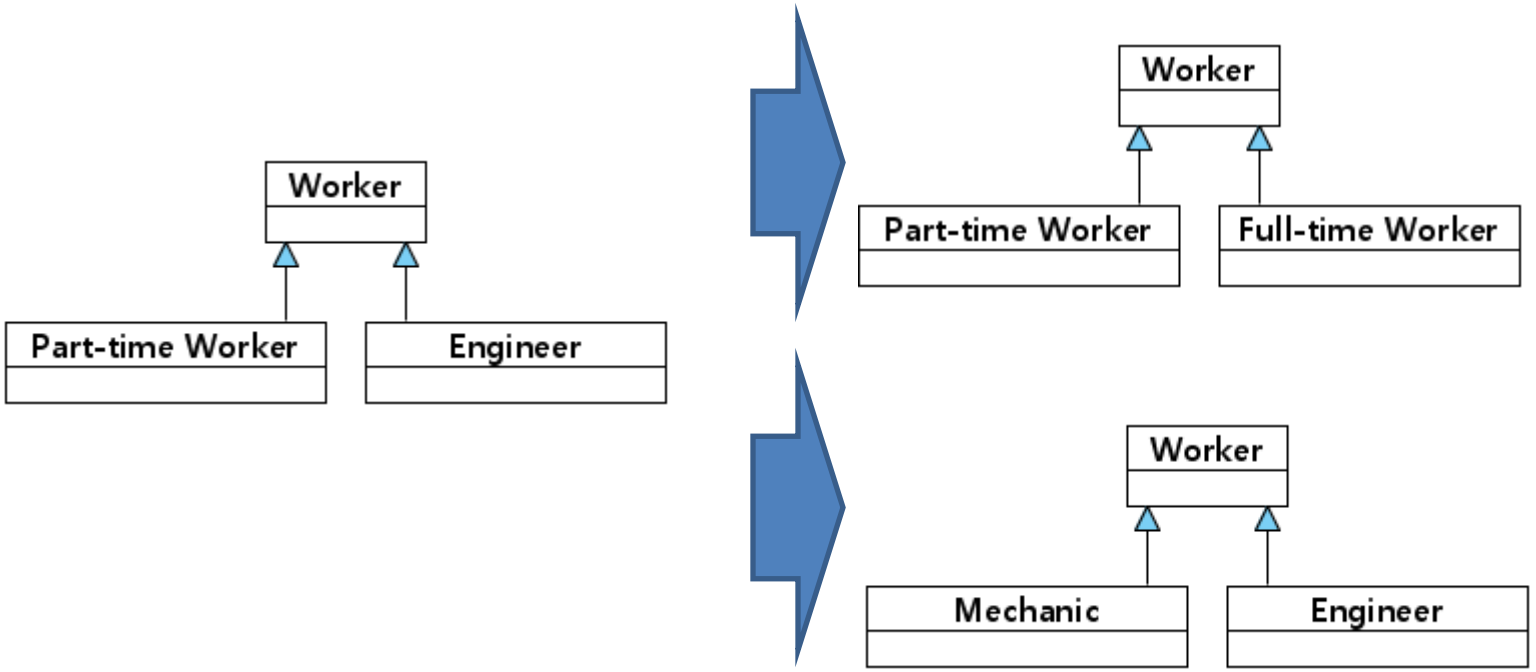
일반화 계층 구조는 균형적이어야 한다

- 상위 클래스와 하위 클래스 간의 의미적 차이가 적절해야 한다.



일반화 계층 구조는 균형적이어야 한다

- 형제 클래스들은 동등한 수준의 개념을 의미해야 한다.



일반화 계층 구조는 균형적이어야 한다