Revisit Java Programming
(Class 2)

Fall, 2020

Table of Contents

Inheritance

- Inherit member variables and methods from **a** parent
  - Simplicity
  - Manageability
  - Producibility

클래스 상속과 객체

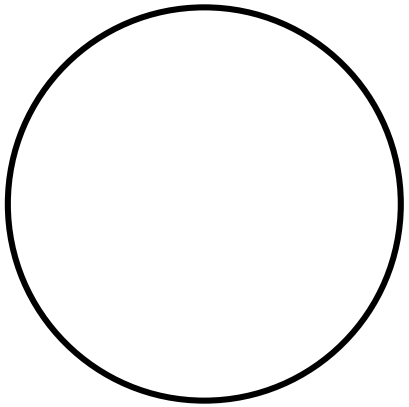
- 자바의 상속 선언

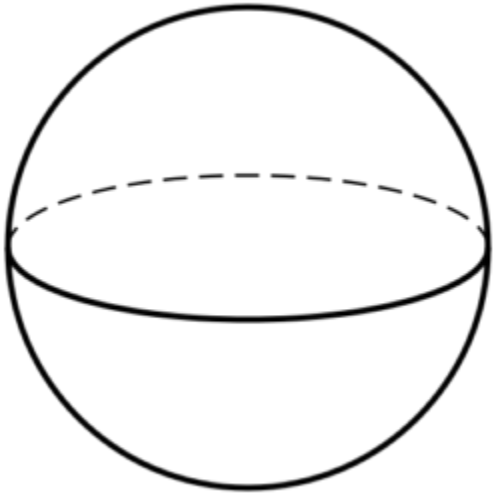
```
public class Person {
        ...
}
public class Student extends Person { // Person을 상속받는 클래스 Student 선언
        ...
}
public class StudentWorker extends Student { // Student를 상속받는 StudentWorker 선언
        ...
}
```

  - 부모 클래스 -> 슈퍼 클래스(super class)로 부름
  - 자식 클래스 -> 서브 클래스(sub class)로 부름
  - extends 키워드 사용
    - 슈퍼 클래스를 확장한다는 개념

Practice 1: Circle and Sphere

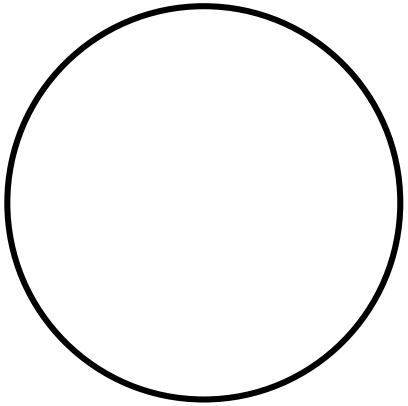
- Circle
  - Member variables
    - x point
    - y point
    - r radius
  - Methods
    - double getArea()
- Sphere
  - Member variables
    - x point
    - y point
    - z point
    - r radius
  - Methods
    - double getVolume()

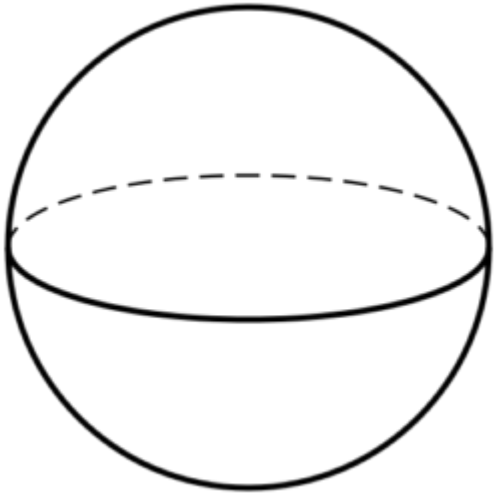




Practice 1: Circle and Sphere

- Feel the necessity of inheritance





Upcasting / Downcasting

- Upcasting
  - revisit a casting from int to long
  - int intVal = 3;
  - long longVal = (long) 3;

  - Circle c = (Circle) new Sphere();

- Downcasting
  - Revisit a casting from long to int

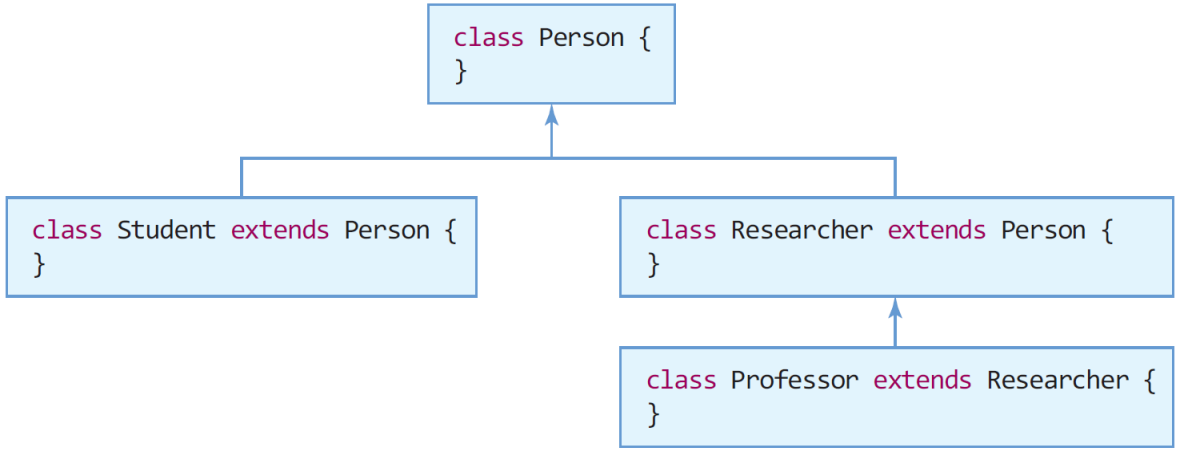
  - Circle c = (Circle) new Sphere();
  - Sphere s = (Sphere) c;

  - Sphere s = (Sphere) new Circle(); ????

Check whether an object is an instance of a class

- Instanceof operator
  - Circle c = (Circle) new Sphere();
  - if(c instanceof Sphere)
    - Sphere s = (Sphere) c;

instanceof 사용 예




```
Person jee= new Student();
Person kim = new Professor();
Person lee = new Researcher();
if (jee instanceof Person)      // jee는 Person 타입이므로 true
if (jee instanceof Student)     // jee는 Student 타입이므로 true
if (kim instanceof Student)     // kim은 Student 타입이 아니므로 false
if (kim instanceof Professor)   // kim은 Professor 타입이므로 true
if (kim instanceof Researcher) // Professor 객체는 Researcher 타입이기도 하므로 true
if (lee instanceof Professor)   // lee는 Professor 타입이 아니므로 false
```

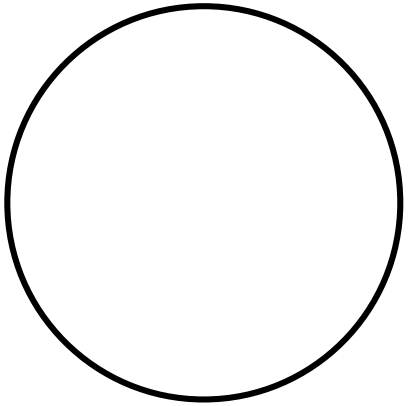
오류

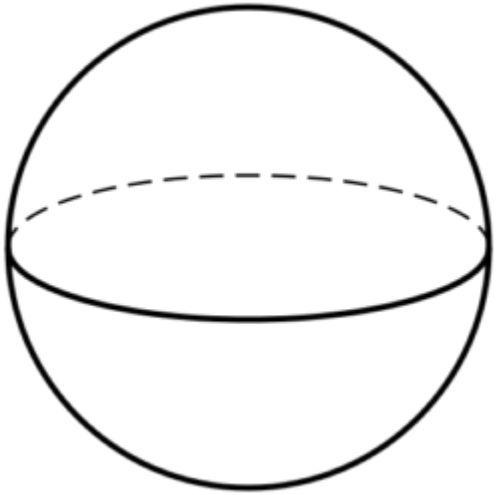
```
if(3 instanceof int) // 문법 오류. instanceof는 객체에 대한 레퍼런스만 사용
```

```
if("java" instanceof String) // true
```

Practice 2: Casting

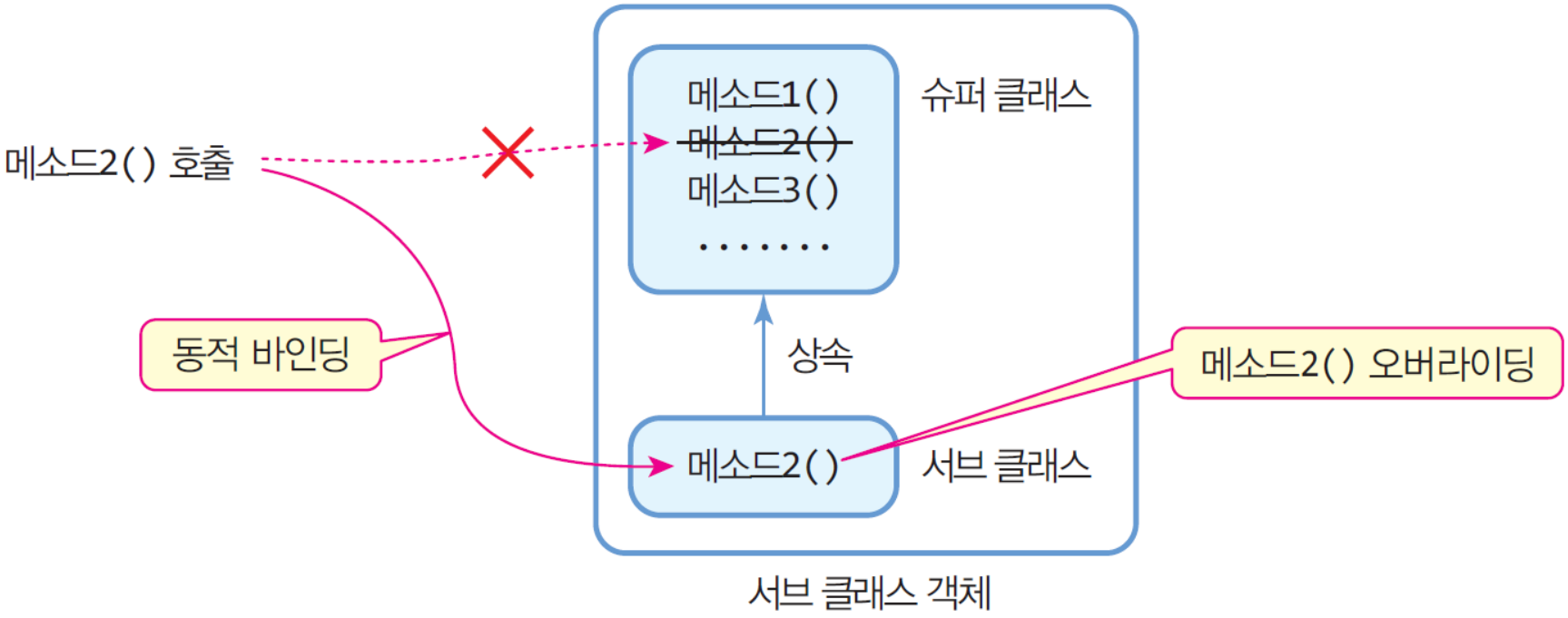
- Practice upcasting / downcasting / instanceof





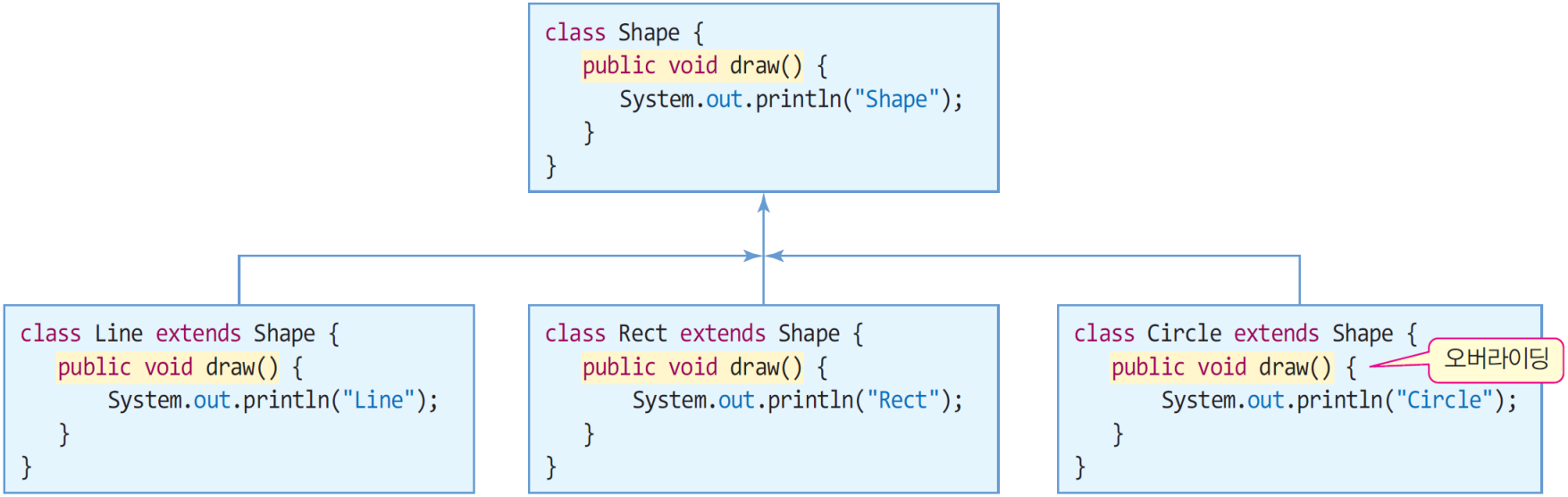
Method Overriding

- Redefining methods of a super class
  - The methods should have same
    - Return type
    - Method name
    - Parameter(s)



Method Overriding

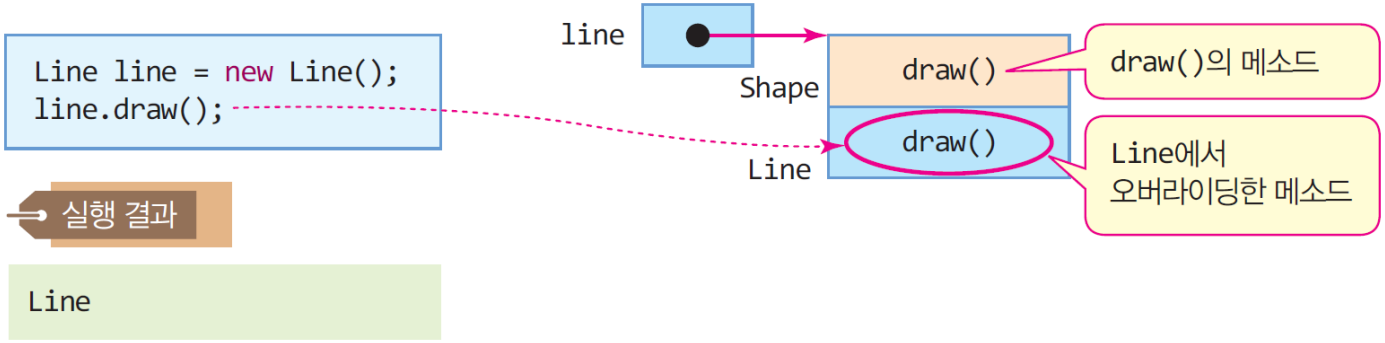
- Redefining methods of a super class
  - The methods should have same
    - Return type
    - Method name
    - Parameter(s)



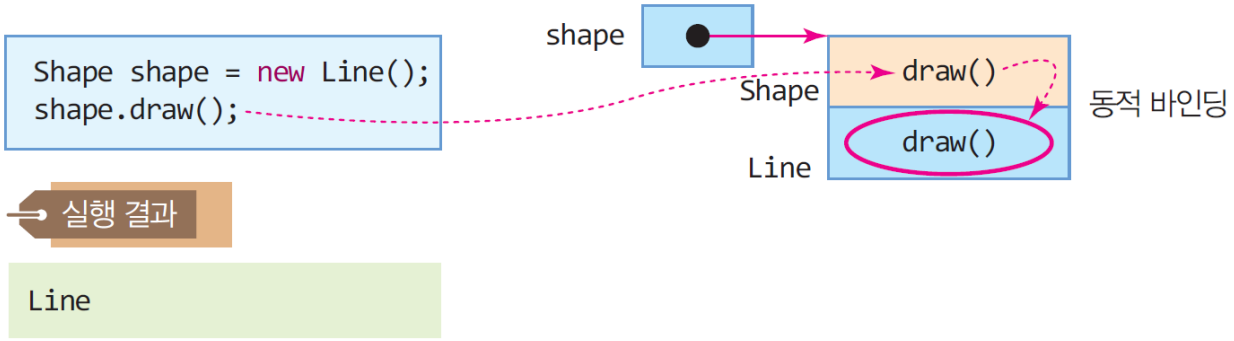
Method Overriding

- Redefining methods of a super class

(1) 서브 클래스 레퍼런스로 오버라이딩된 메소드 호출

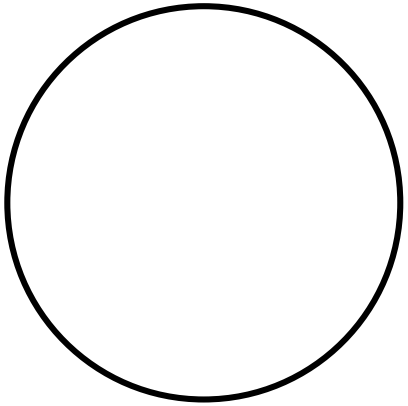


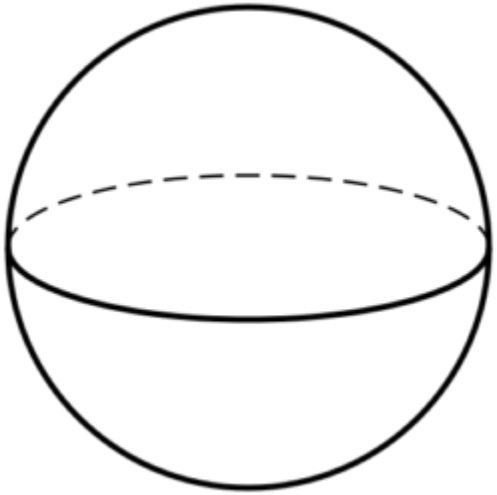
(2) 업캐스팅에 의해 슈퍼 클래스 레퍼런스로 오버라이딩된 메소드 호출(동적 바인딩)



Practice 3: Method Overriding

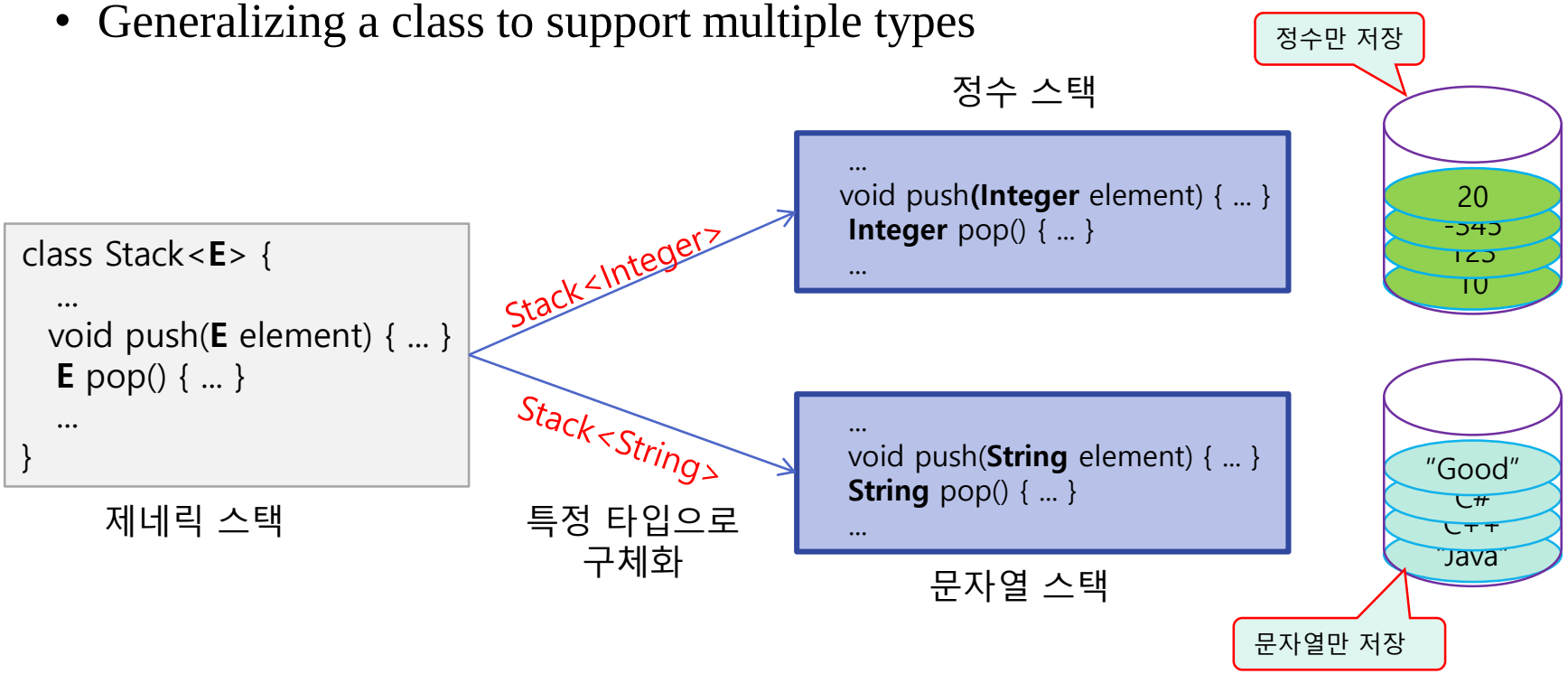
- Practice method overriding





Generic and Parametric Polymorphism

- Motivation
  - Revisit MyStack
  - Implement another MyStack2 handling int[]
  - Implement another MyStack3 handling char[]
  - Implement another MyStack4 handling Circle[]
- Generic
  - Generalizing a class to support multiple types



Practice 5: Generic and Polymorephism

- Implement int[] MyStack

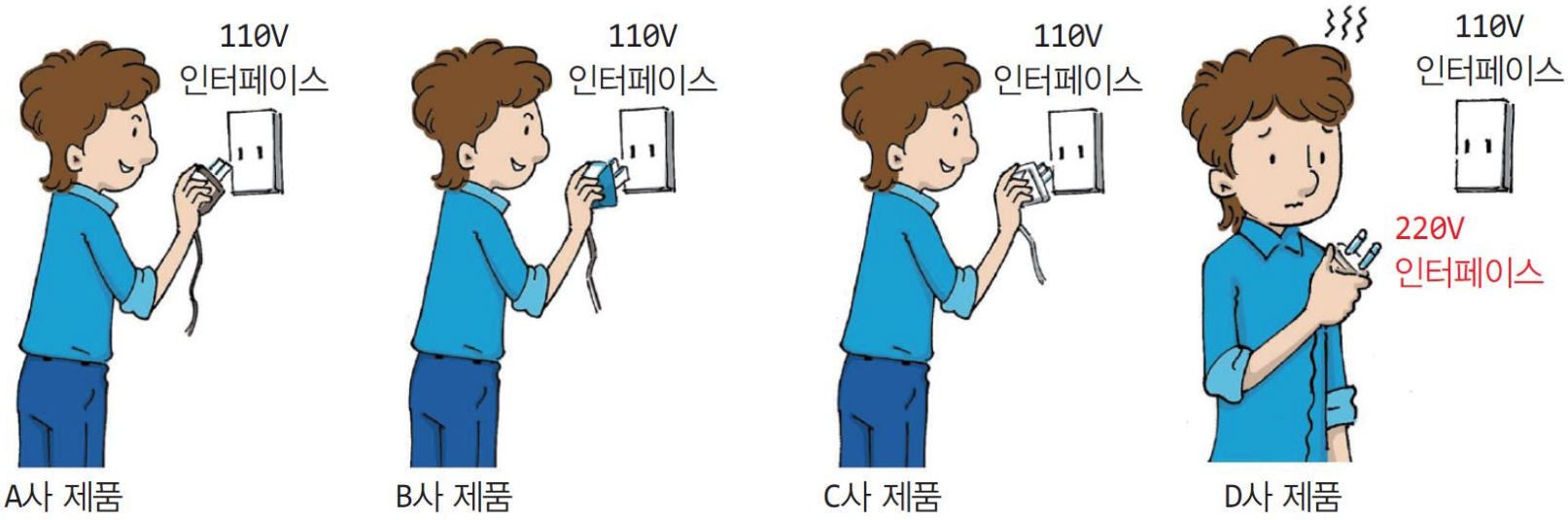
Practice 6: Generic and Polymorephism

- Implement MyStack extends Stack<String>

Practice 7: Generic and Polymorephism

- Implement MyStack<E>

실세계의 인터페이스

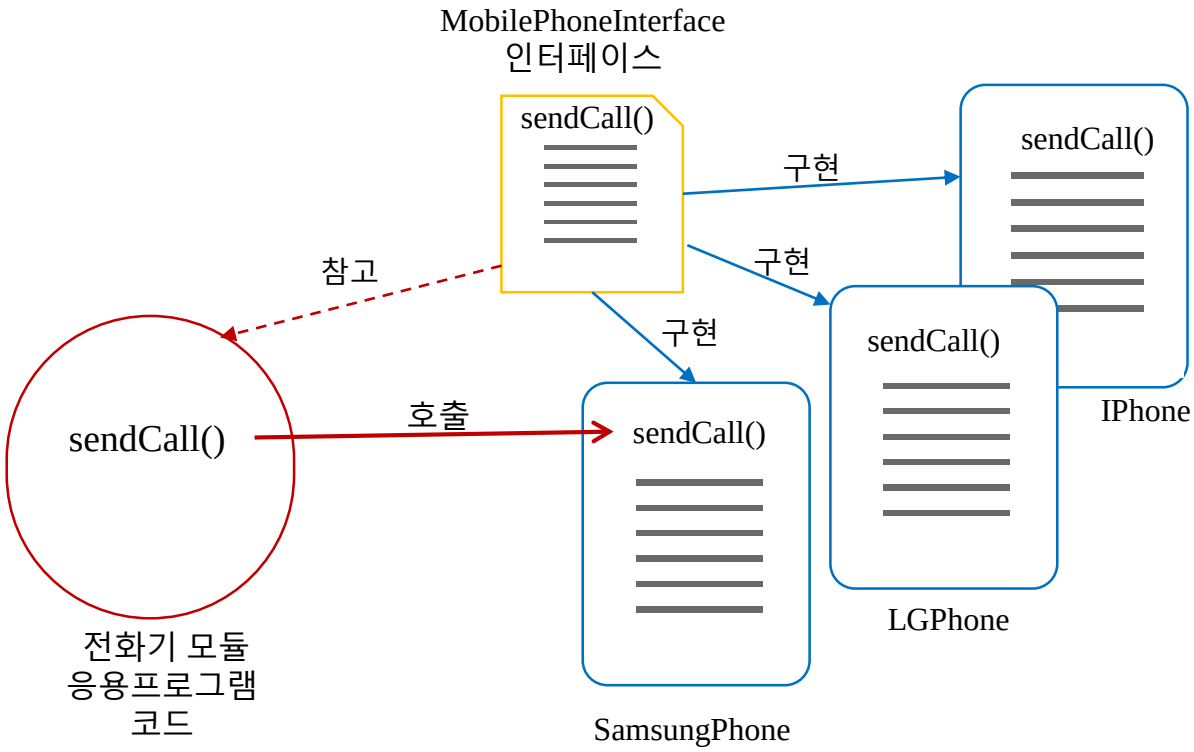


정해진 규격(인터페이스)에 맞기만 하면 연결 가능.
각 회사마다 구현 방법은 다름

정해진 규격(인터페이스)에 맞지 않으면 연결 불가

인터페이스의 목적

인터페이스는 스펙을 주어 클래스들이 그 기능을 서로 다르게 구현할 수 있도록 하는 클래스의 규격 선언이며, 클래스의 다형성을 실현하는 도구이다



자바의 인터페이스

- 자바의 인터페이스
  - 클래스가 구현해야 할 메소드들이 선언되는 추상형
  - 인터페이스 선언
    - **interface** 키워드로 선언
    - Ex) public **interface** SerialDriver {…}
- 자바 인터페이스에 대한 변화
  - Java 7까지
    - 인터페이스는 상수와 추상 메소드로만 구성
  - Java 8부터
    - 상수와 추상메소드 포함
    - default 메소드 포함 (Java 8)
    - private 메소드 포함 (Java 9)
    - static 메소드 포함 (Java 9)
  - 여전히 인터페이스에는 **필드(멤버 변수) 선언 불가**

인터페이스 상속

- 인터페이스가 다른 인터페이스 상속
  - extends 키워드 이용

```
interface MobilePhoneInterface extends PhoneInterface {
    void sendSMS();     // 새로운 추상 메소드 추가
    void receiveSMS();     // 새로운 추상 메소드 추가
}
```

  - 다중 인터페이스 상속

```
interface MP3Interface {
    void play(); // 추상 메소드
    void stop(); // 추상 메소드
}

interface MusicPhoneInterface extends MobilePhoneInterface, MP3Interface {
    void playMP3RingTone(); // 새로운 추상 메소드 추가
}
```

Practice 8: Create IStack

- Create IStack interface has
  - public void push(String elem);
  - public String pop();
- Create MyStack class implements IStack
- Create IStack2 interface has
  - public void push(Integer elem);
  - public Integer pop();
- Create MyStack2 class implements IStack2
- Create Istack<E> interface has
  - public void push(E elem);
  - public E pop();
- Create GeneralStack<E> class implements IStack<E>

Practice 8: Implements Queue

- MyStackQueue extends Stack implements Queue

Summary

- Class
- Inheritance
- Generic
- Interface