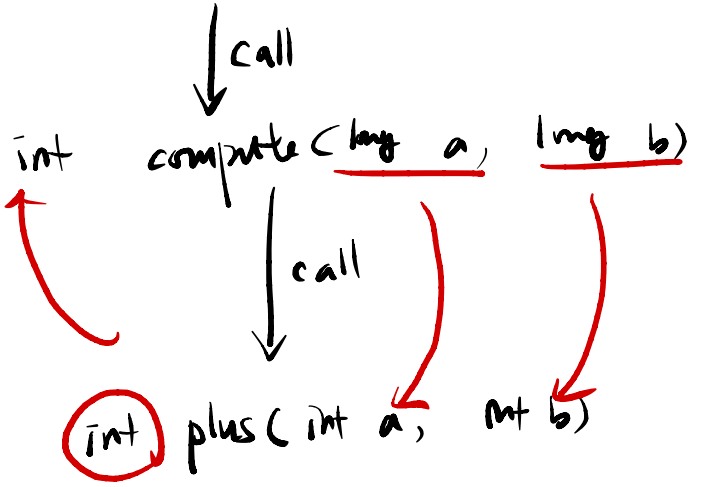
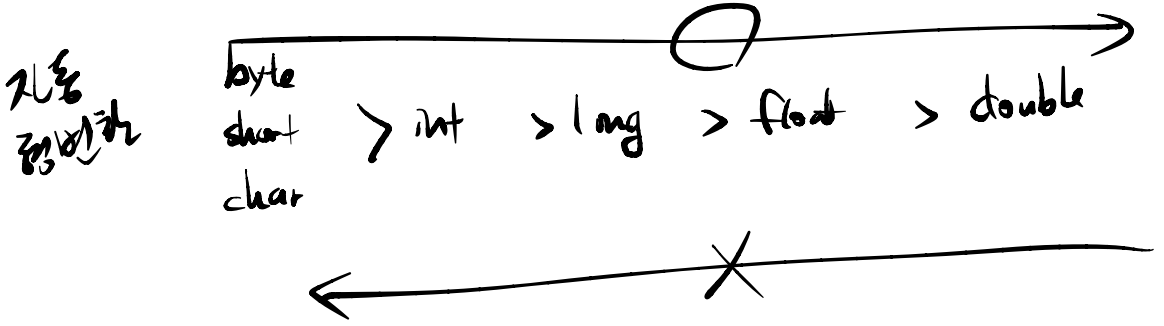
Calculator c1 = MyCalculator::power;

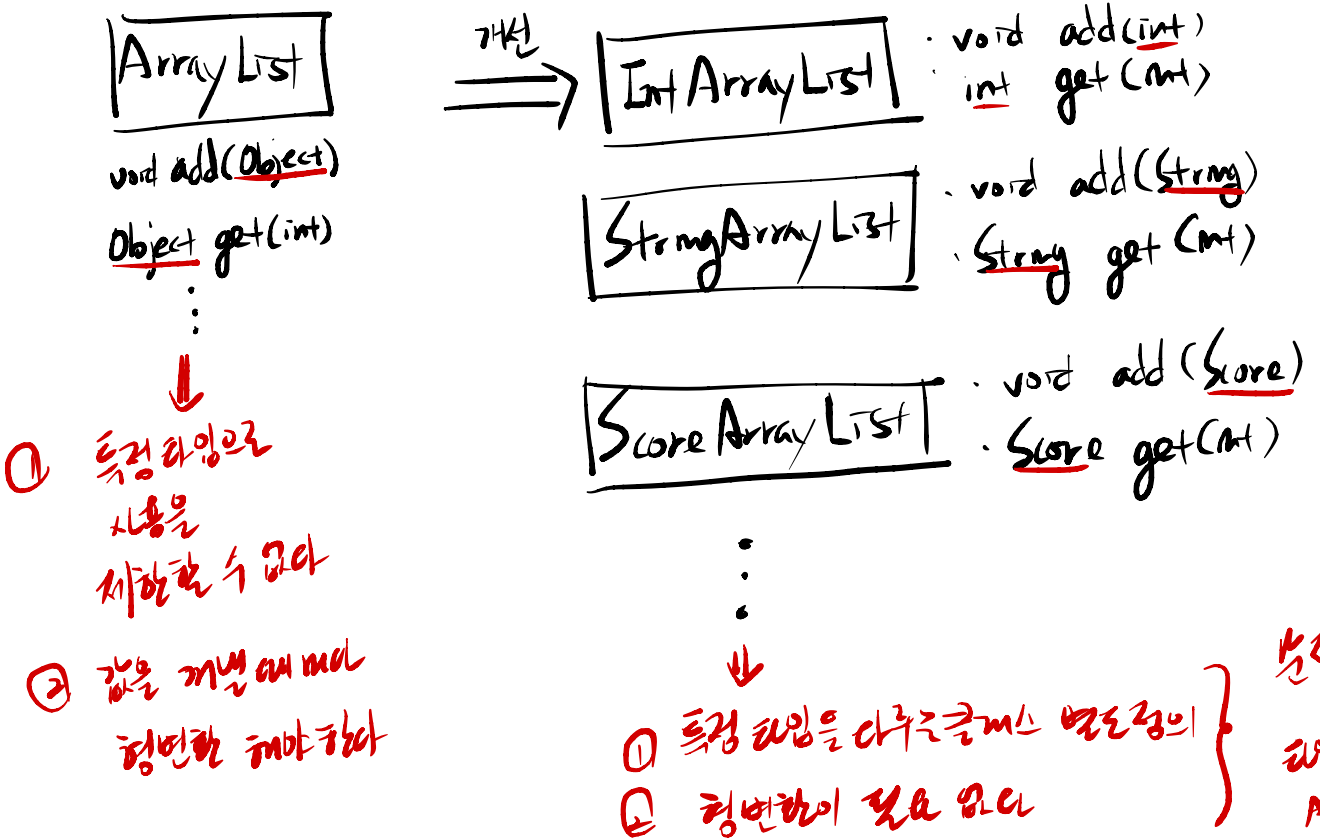
컴파일러에게 이 메서드를 사용해서
인터페이스 구현체를 만들라고 명령

```
new Calculator() {
    public int compute(int x, int y) {
        return MyCalculator.power(x, y);
    }
}
```


call
int compute(long a, long b)
call
int plus(int a, int b)

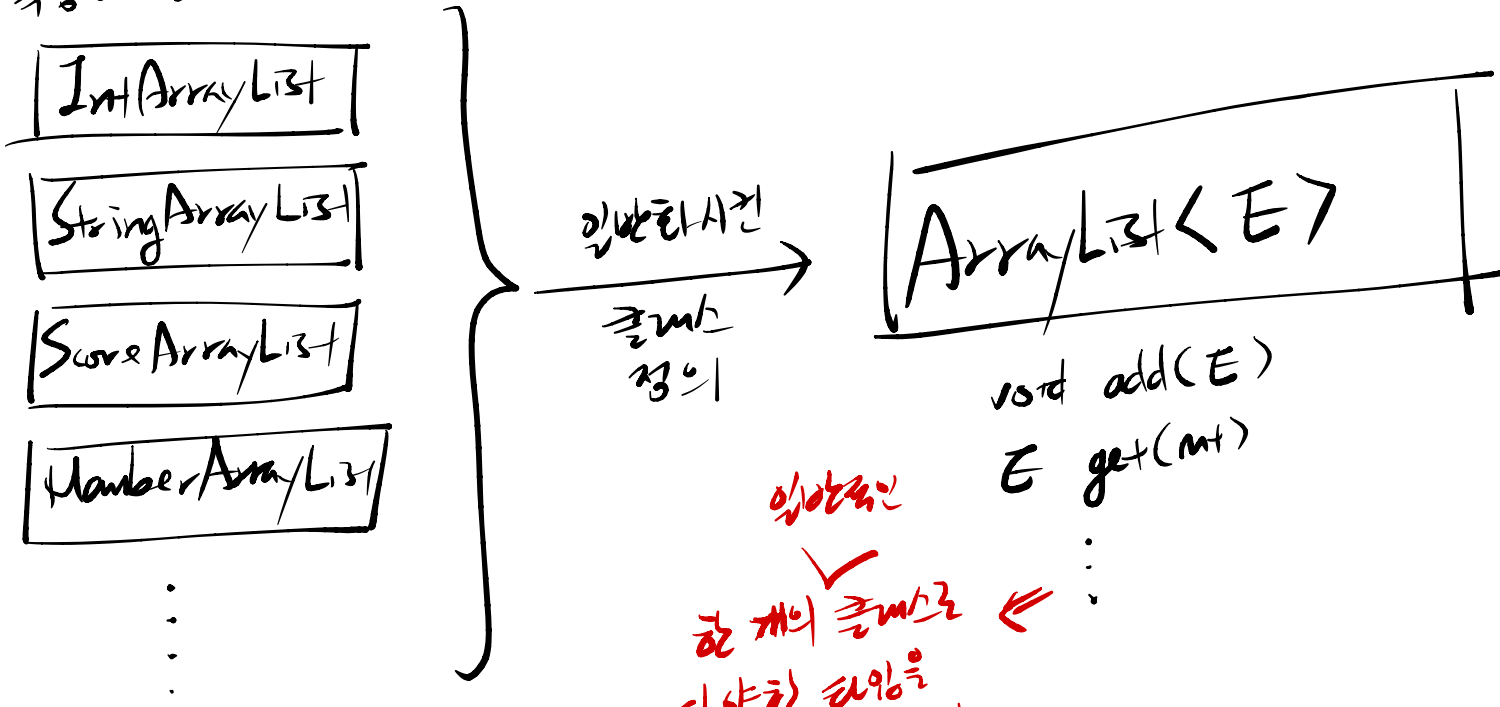

자동 형변환
byte
short
char
> int > long > float > double

* 제네릭 도입 시점

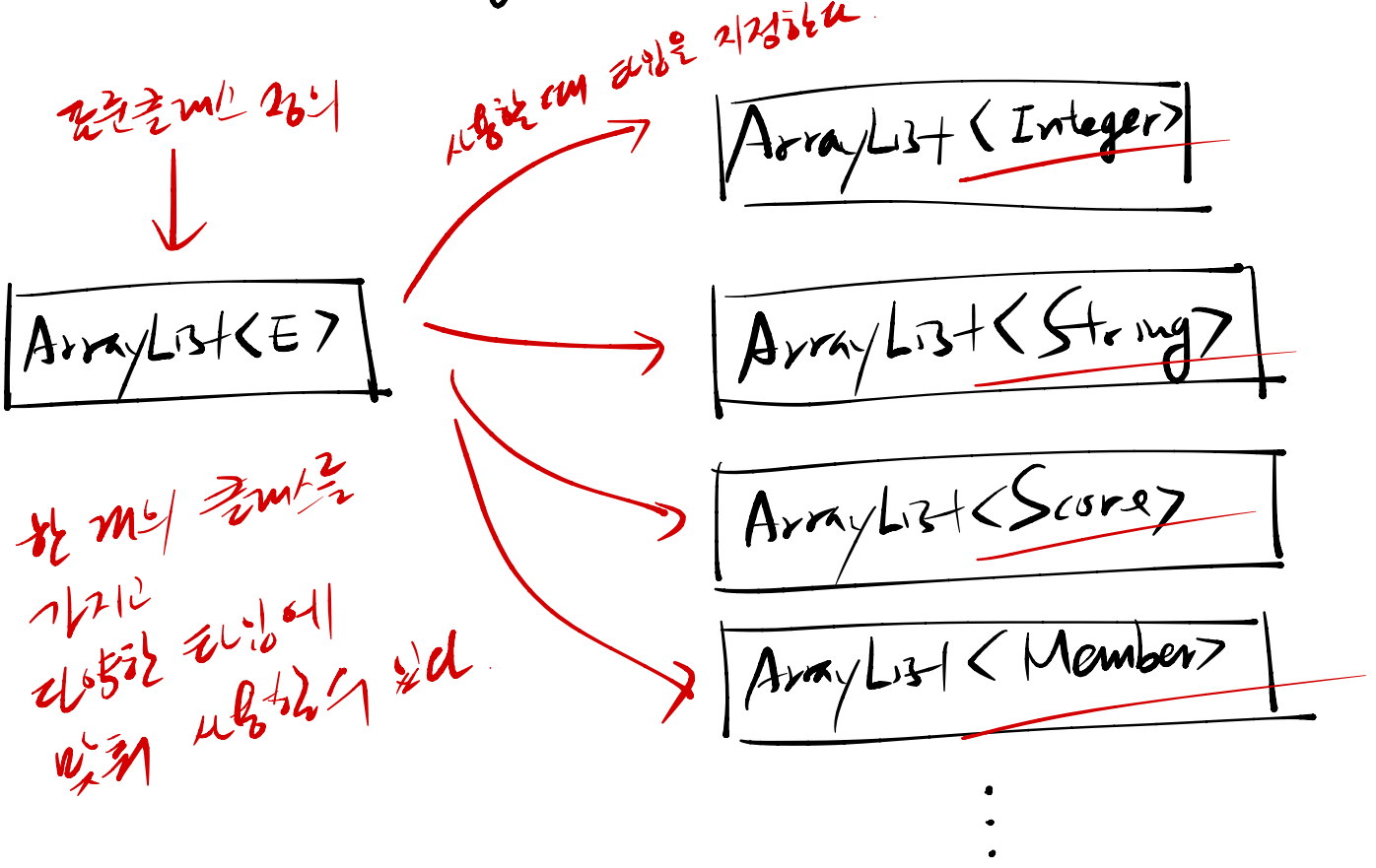

ArrayList
void add(Object)
Object get(int)
개선
IntArrayList
· void add(int)
· int get(int)
StringArrayList
· void add(String)
· String get(int)
ScoreArrayList
· void add(Score)
· Score get(int)
① 특정 타입으로 사용을 제한할 수 없다
② 값을 꺼낼 때 cast 형변환 해야한다
① 특정 타입을 다루는 클래스 별도 정의
② 형변환이 필요 없다
불리한점이 있다
타입별로 ArrayList 정의해야 한다
장점을 얻을려고 클래스를 여러개 만든다

* 제네릭(generic) 문법 사용 x

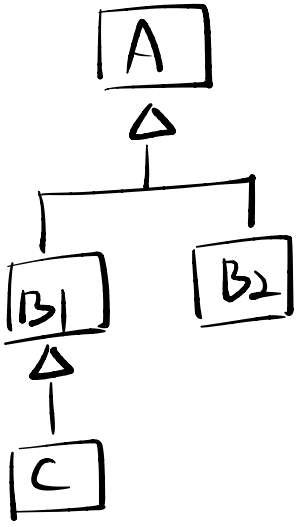
특정 타입을 다루는 클래스



✗ 제네릭(generic) 클래스 사용



★ 제네릭 : 레퍼런스 와 인스턴스



ArrayList<A> list; ⇒ add(A obj){-}

list = new ArrayList<B2>(); ← 허락안함

- list.add(new A()) ✗
- list.add(new B1()) ✗
- list.add(new B2()) ○
- list.add(new C()) ✗

← 레퍼런스에서 허용한 타입과 인스턴스에서 허용한 타입이 다르면 안됨!

✱ 제네릭 <?>

ArrayList<?> list;

list.add(new A());

add(? obj) { — }

↑

파라미터 타입이 정해지지 않았기 때문에

어떤 타입이 값이 가능한지 컴파일러로 판단할 수 없다!