```
public class BankAccount {
    private int balance;
    ...
}


public class MinimumBalanceAccount extends BankAccount {
    ...
    @Override
    public boolean withdraw(int amount) {
        if (getBalance() - amount < minimum) {
            System.out.println("적어도 " + minimum + "원은 남겨야 합니다.");
            return false;
        }

        setBalance(getBalance() - amount);
        return true;
    }
}
```

자식 클래스 **MinimumBalanceAccount** 에서는 **private** 변수인 **balance** 에 대한 권한이 없기 때문에, **public** 메소드인 **getBalance** 와 **setBalance** 메소드를 사용했습니다. 보호 차원에서는 좋지만, 상속한 클래스에서의 접근성마저 떨어져서 유연하지 못하고 불편합니다.

protected 접근 제어자

이럴 때에는 **private** 대신 **protected** 를 사용하면 됩니다!

protected 는 **private** 과 유사하지만 **자식 클래스에선 접근 가능**하도록 합니다. 상속의 유연함과 **private** 의 보호성을 동시에 만족시켜주는 키워드죠!

```
public class BankAccount {
    protected int balance;
    ...
}
```

```
public class MinimumBalanceAccount extends BankAccount {
    ...
    @Override
    public boolean withdraw(int amount) {
        // if (getBalance() - amount < minimum) {
        if (balance - amount < minimum) {
            System.out.println("적어도 " + minimum + "원은 남겨야 합니다.");
            return false;
        }

        // setBalance(getBalance() - amount);
        balance -= amount;
        return true;
    }
}
```

그럼 이제 자식 클래스에서 **balance** 변수에 바로 접근할 수 있습니다.

상속받는 클래스에서만 접근 가능하나, **BankDriver** 같은 외부 클래스에서 접근이 안 되는 것은 여전히 유효 합니다.

상속받은 클래스에서의 접근이 잦을 경우, 내부 변수를 **private** 이 아닌 **protected** 로 선언해야할 때가 꽤 있는데, 부모 클래스나 자식 클래스 모두 스스로 작성한다면 상황에 따라 유연하게 바꾸어가며 작성할 수 있겠죠?

☑ 수업을 완료하셨으면 체크해주세요.

수강생 Q&A 보기 (/questions?assignment_id=421&sort_by=popular)

질문하기 (/questions/new?assignment_id=421&op1=%EA%B0%9D%EC%B2%B4+%EC%A7

이전 강의 super 퀴즈 (해설 노트) (/assignments/489)

다음 강의 Object Class (/assignments/437)