KNU 강원대학교
KANGWON NATIONAL UNIVERSITY

자바프로그래밍2

5주차 실습

상속

2024-2학기 자바프로그래밍2 3분반
TA 강지호

실습 평가 방법

- 점수 : 100점 만점 기준(문제 별 난이도에 따라 점수 부여)
- 채점 기준 : 완성도, 작동 유무, 일부 오류 등에 따라 감점
  - 프로그램이 동작하지 않거나, 코드 공유, ChatGpt 사용 등의 부정행위 적발 시 0점
  - 소스 코드에 허점, 잘못된 들여쓰기, 일부 입출력 오작동 시 정도에 따라 감점
- 제출 기한 : 실습 당일 23시 59분까지(이후 제출 불가능)
  - 실습 시간(14:00-15:50) 내 제출 시 감점X
  - 18:00 까지 제출 시 채점 점수의 5% 감점
  - 20:00 까지 제출 시 채점 점수의 10% 감점
  - 23:59 까지 제출 시 채점 점수의 20% 감점

실습 제출 방법

- 압축 파일명 : n주차_학번_이름.zip
- 소스 파일 : Eclipse에서 Export한 zip 파일 내 소스 파일(.java)
- 보고서 : 각 문제별 문제 번호 및 소스 코드 실행 결과 화면 캡처한 **pdf 파일** - 부재 시 감점
- 소스 파일과 보고서를 압축하여 주차별로 위 압축 파일명과 같이 e-루리에 제출
- e-루리 접속 오류 등 특별한 사유로 인해 제출하지 못하는 경우
  - rkdwlgh01@naver.com 해당 e-mail을 통해 제출

실습 조교 및 질의응답

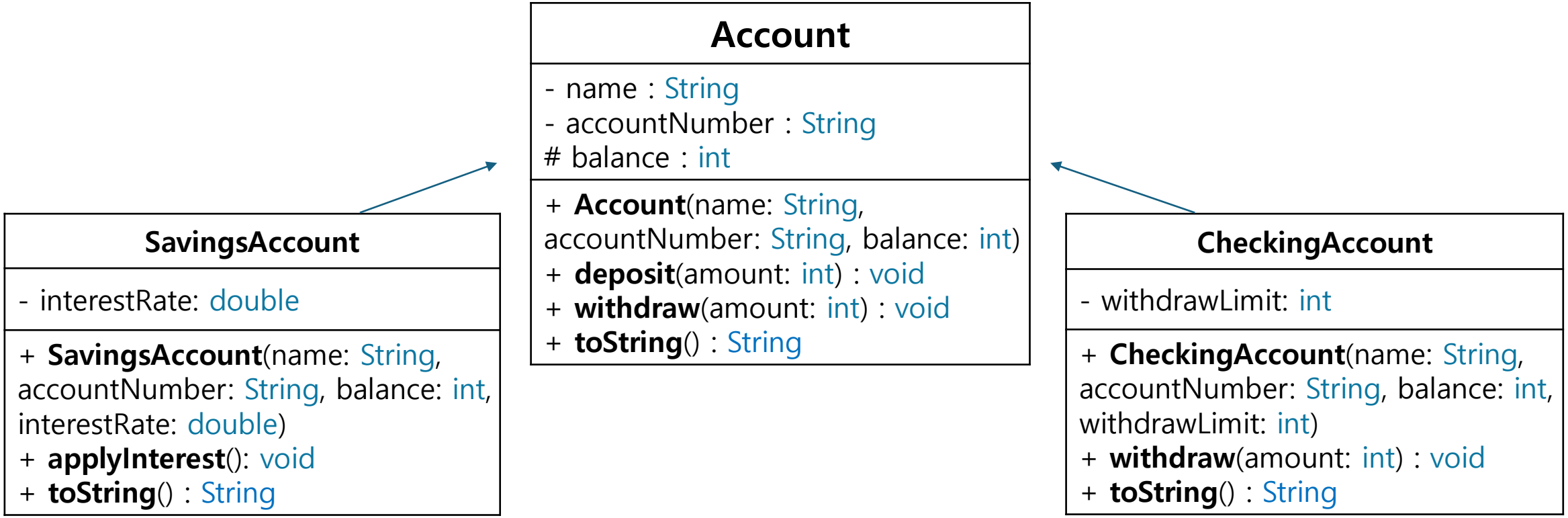
- e-루리 Q&A 게시판 활용 (작성 후 e-루리 메시지 시 빠른 응답 가능)
- 실습 TA의 e-mail 활용
  - 강지호 : rkdwlgh01@naver.com

[자바프로그래밍2]
5주차 실습 문제

제출 기한 10.2(수) 23:59 전까지

문제 1 (40점)

- 부모 클래스 **Account**와 자식 클래스 **SavingsAccount, CheckingAccount**를 조건에 맞게 구현하고 **Exercise01**에서 테스트하세요.



문제 1 (40점)

Account
- name : String - accountNumber : String # balance : int
+ **Account**(name: String, accountNumber: String, balance: int) + **deposit**(amount: int) : void + **withdraw**(amount: int) : void + **toString**() : String

Account (부모 클래스)

- 필드는 balance를 제외하고 외부에서 접근할 수 없도록 설정
- 변수 balance는 자식 클래스에서는 접근 가능하도록 설정
- **생성자** 구현
- **입금**, **출금** 동작을 하는 메소드 각각 구현

  - 반환 타입이 void형으로 각 메소드 내에서 출력

- toString() 메소드를 **오버라이딩**하여 객체의 상태를 아래 형식으로 출력

  - "예금주: name, 계좌번호: accountNumber, 잔액: balance원"

문제 1 (40점)

SavingsAccount
- interestRate: double //이자율
+ **SavingsAccount**(name: String, accountNumber: String, balance: int, interestRate: double) + **applyInterest**(): void + **toString**() : String

SavingsAccount (자식 클래스) //저축 계좌

- 필드는 외부에서 접근할 수 없도록 설정
- **생성자** 구현
- 잔액에 이자를 적용하는 **applyInterest**() 메소드 구현
- toString() 메소드를 **오버라이딩**하여 객체의 상태를 출력

CheckingAccount
- withdrawLimit: int //출금 한도
+ **CheckingAccount**(name: String, accountNumber: String, balance: int, withdrawLimit: int) + **withdraw**(amount: int) : void + **toString**() : String

CheckingAccount (자식 클래스) //입출금 계좌

- 필드는 외부에서 접근할 수 없도록 설정
- **생성자** 구현
- withdraw() 메소드를 **오버라이딩**하여 출금 한도를 초과하지 않도록 확인
- toString() 메소드를 **오버라이딩**하여 객체의 상태를 출력

문제 1 (40점)

- 클래스 명 : **Account, SavingsAccount, CheckingAccount, Exercise01**

- Account.java
- CheckingAccount.java
- Exercise01.java
- SavingsAccount.java

```
public class Exercise01 {

    public static void main(String[] args) {

        // 저축 계좌 생성
        savingsAccount = new ("홍길동", "1001", 100000, 0.03);
        System.out.println(savingsAccount);
        savingsAccount.(50000);
        savingsAccount.;

        System.out.println();

        // 입출금 계좌 생성
         checkingAccount = new ("홍길동", "1002", 200000, 100000);
        System.out.println(checkingAccount);
        checkingAccount.(120000);
        checkingAccount.(100000);

    }

}
```

실행결과 예시 :

```
저축 계좌 [예금주: 홍길동, 계좌번호: 1001, 잔액: 100000원, 이자율: 3.0%]
50000원이 입금되었습니다. 잔액: 150000원
이자가 4500.0원 적용되었습니다. 잔액: 154500원

입출금 계좌 [예금주: 홍길동, 계좌번호: 1002, 잔액: 200000원, 출금 한도: 100000원]
출금 한도를 초과했습니다. 출금 한도: 100000원
100000원이 출금되었습니다. 잔액: 100000원
```

문제 2 (20점)

- 문제 1의 **Account** 클래스에서 equals()를 조건에 맞게 재정의하여 **Exercise02**에서 테스트하세요.
  - 기본적으로 equals() 메소드는 객체의 **주소**를 비교하기 때문에 오버라이딩 없이 객체 비교 시, 주소가 동일한 객체가 아니면 fasle가 반환됨
  - 따라서, equals() 메소드를 **오버라이딩**하여 객체의 **계좌번호**가 동일하면 true를 반환

- 클래스 명 : Exercise02

문제 2 (20점)

실행결과 예시 :

```
3  public class Exercise02 {
4
5      public static void main(String[] args) {
6
7          Account a1 = new Account("홍길동", "20240001", 10000);
8          Account a2 = new Account("홍길동", "20240001", 50000);
9          SavingsAccount a3 = new SavingsAccount("홍길동", "20240001", 100000, 0.03);
10         Account a4 = new Account("홍길동", "20240002", 10000);
11
12
13         if(a1.equals(a2))
14             System.out.println("a1과 a2는 동일한 계좌입니다.");
15         else
16             System.out.println("a1과 a2는 동일한 계좌가 아닙니다.");
17
18         if(a1.equals(a3))
19             System.out.println("a1과 a3은 동일한 계좌입니다.");
20         else
21             System.out.println("a1과 a3은 동일한 계좌가 아닙니다.");
22
23         if(a1.equals(a4))
24             System.out.println("a1과 a4은 동일한 계좌입니다.");
25         else
26             System.out.println("a1괴 a4는 동일한 계좌가 아닙니다.");
27
28     }
29
30 }
31
```

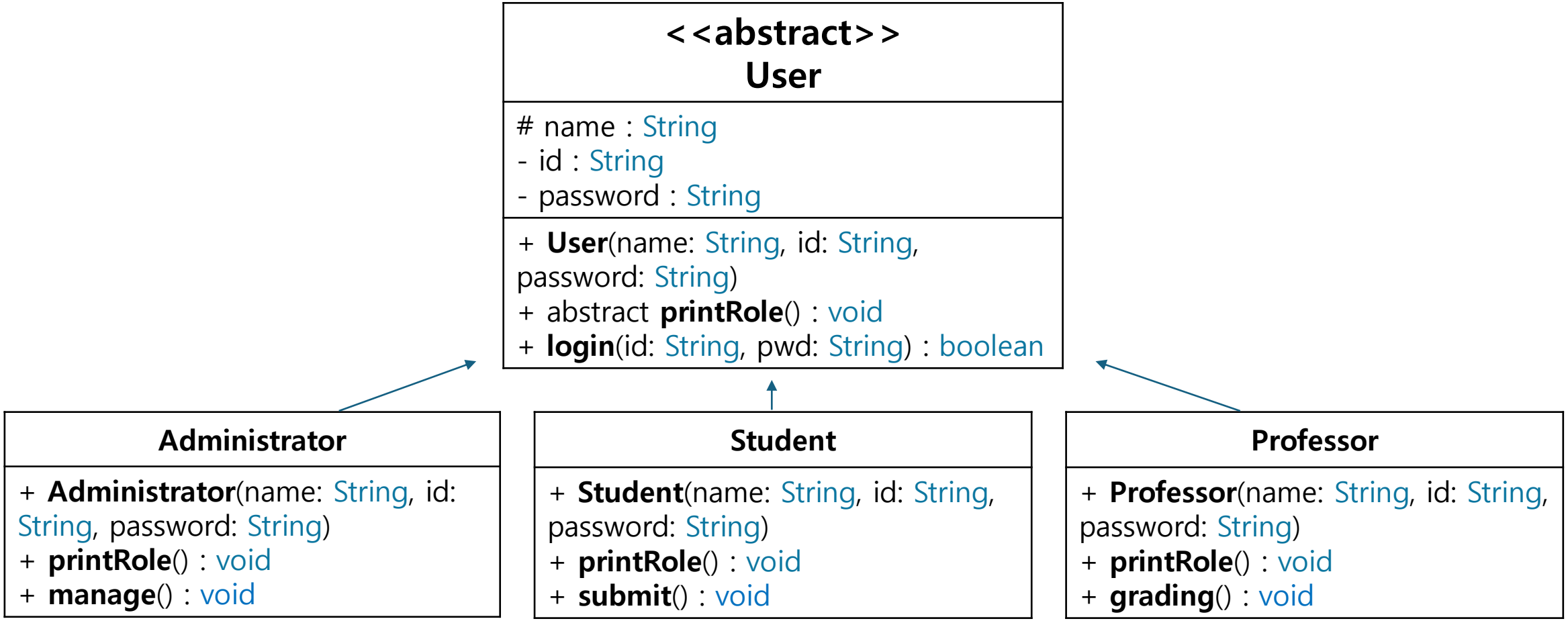
Console Problems Javadoc Declaration

<terminated> Exercise02 (4) [Java Application] C:₩Program Files₩Eclipse Foundation₩jdk-11.0.12.7-hotspot₩bin₩jav

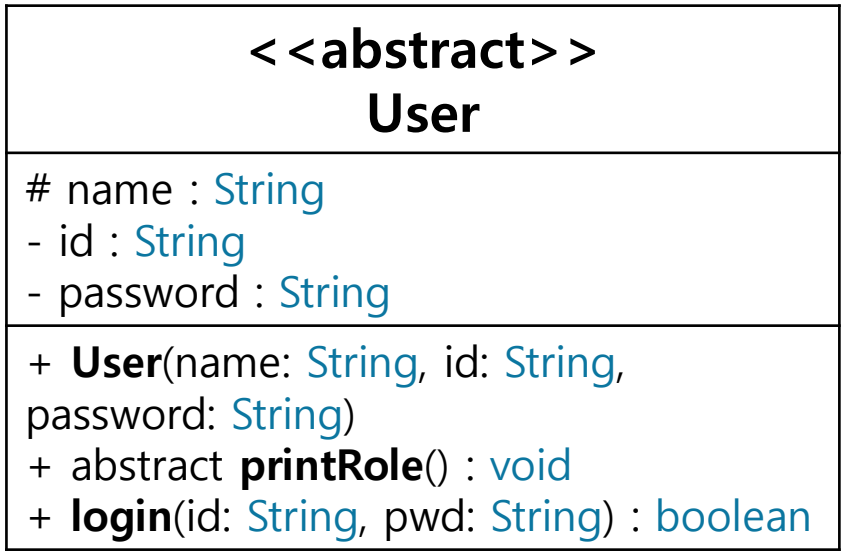
```
a1과 a2는 동일한 계좌입니다.
a1과 a3은 동일한 계좌입니다.
a1괴 a4는 동일한 계좌가 아닙니다.
```

문제 3 (40점)

- 추상 클래스 **User**와 자식 클래스 **Administrator, Student, Professor**를 조건에 맞게 구현하고 **Exercise03**에서 테스트하세요.



문제 3 (40점)



User (추상 클래스)

- 필드는 name을 제외하고 외부에서 접근할 수 없도록 설정
- 변수 name은 자식 클래스에서는 접근 가능하도록 설정
- **생성자** 구현
- abstract printRole()는 추상 메소드로 객체의 역할을 출력
- login() 메소드는 로그인 성공 - true, 로그인 실패 – false 반환

문제 3 (40점)

Administrator
+ **Administrator**(name: String, id: String, password: String) + **printRole**() : void + **manage**() : void

Administrator (자식 클래스) //관리자

- **생성자** 구현
- **printRole**() 메소드 구현 — ' name '님은 관리자입니다.
- manage() 메소드 구현 — ' name '님이 시스템을 관리합니다.

Student
+ **Student**(name: String, id: String, password: String) + **printRole**() : void + **submit**() : void

Student (자식 클래스) //학생

- **생성자** 구현
- **printRole**() 메소드 구현 — ' name '님은 학생입니다.
- sumit() 메소드 구현 — ' name '님이 과제를 제출합니다.

Professor
+ **Professor**(name: String, id: String, password: String) + **printRole**() : void + **grading**() : void

Professor (자식 클래스) //교수

- **생성자** 구현
- **printRole**() 메소드 구현 — ' name '님은 교수입니다.
- grading() 메소드 구현 — ' name '님이 채점을 시작합니다.

문제 3 (40점)

- 아래와 같이 **Exercise03** 구현
  - Administrator, Student, Professor 객체를 **상향형변환**으로 User 배열에 생성하여 저장
  - 로그인할 아이디와 비밀번호를 입력 받아 User 배열 탐색, 로그인 성공 시 변수에 로그인한 객체 저장(변수의 초기값은 null)
  - 로그인된 사용자가 어떤 타입인지 확인 후 **하향형변환**하여 고유 메소드 호출

```
import java.util.Scanner;

public class Exercise03 {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        // 상향형변환으로 User 배열에 다양한 사용자 객체 저장
        User[] users = new User[3];
        users[0] = new                ("강백호", "admin01", "1234");
        users[1] = new               ("서태웅", "student01", "0000");
        users[2] = new                ("송태섭", "professor01", "1111");

        System.out.print("아이디를 입력하세요: ");
        String id = input.nextLine();
        System.out.print("비밀번호를 입력하세요: ");
        String pwd = input.nextLine();

        User loggedIn = null;
```

```
        if (loggedIn != null) {
            loggedIn.printRole();
            //로그인된 사용자가 어떤 타입인지 확인 후 하향형변환하여 고유 동작 실행
                
        }
        else {
            System.out.println("로그인 실패: 잘못된 아이디 또는 비밀번호입니다.");
        }

        input.close();
    }
}
```

문제 3 (40점)

- 클래스 명 : **User, Exercise03**

실행결과 예시 :

```
아이디를 입력하세요: admin01
비밀번호를 입력하세요: 1234
로그인 성공 : '강백호'님 로그인
'강백호'님은 관리자입니다.
'강백호'님이 시스템을 관리합니다.
```

```
아이디를 입력하세요: student01
비밀번호를 입력하세요: 0000
로그인 성공 : '서태웅'님 로그인
'서태웅'님은 학생입니다.
'서태웅'님이 과제를 제출합니다.
```

```
아이디를 입력하세요: professor01
비밀번호를 입력하세요: 1111
로그인 성공 : '송태섭'님 로그인
'송태섭'님은 교수입니다.
'송태섭'님이 채점을 시작합니다.
```

```
아이디를 입력하세요: user01
비밀번호를 입력하세요: 1111
로그인 실패: 잘못된 아이디 또는 비밀번호입니다.
```