**12161581 박진성**

**<1> 소스코드 요구사항 및 설계 주안점**

이번 과제의 소스코드의 요구사항은 크게 두가지였다.

Updown 게임의 알고리즘을 정확히 구현할것.

그리고 최소 2번의 예외처리를 구현할것.

우선, updown 게임 자체의 구현은, Math.random 으로 난수정답을 설정해놓은 뒤

While 문과 if문을 통하여 정답 비교를 수행하는 알고리즘으로 구현하였다.

예외처리를 할 수 있는 부분은 총 3부분 이었다.

첫번째, 게임 결과 파일을 만드는 경우, IOException 이 발생할수 있다.

두번째, 사용자에게 정답을 입력받을 때, nextInt 로 입력받기 때문에,

InputMismatchException 이 발생할 수 있다.

마지막으로, 게임을 다시 실행할 지 묻는 부분에서, 문자를 입력받기 때문에,

또다시 InputMismatchException 이 발생할 수 있다.

따라서 이를 소스코드에 알맞게 구현하였다,

**<2> 소스코드 전체**

**package** Up.Down;

**import** java.util.InputMismatchException;

**import** java.util.Scanner;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** updown {

**public** **static** **void** makeFile(**int** count) **throws** IOException

// 몇번만에 맞췄는지 파일로 저장.

{ // IOException 예외를 throws 해줬다

FileWriter fo = **new** FileWriter("log.txt");

fo.write(count + "번에 맞춤!");

fo.close();

}

**public** **static** **void** main(String[] args) {

**int** n;

**int** count = 0;

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("학번:12161581 이름:박진성");

**while**(**true**) {

**int** rand = (**int**)(Math.*random*()\*100+1);

**char** c = 0;

**try** {

**while**(**true**) {

System.***out***.print("input>");

n = scanner.nextInt();

// InputMismatchException 일어날수 있는 부분

**if**(n > rand) {

System.***out***.println("Down");

count++;

**continue**;

}

**else** **if**(n < rand) {

System.***out***.println("Up");

count++;

**continue**;

}

**else** {

count++;

System.***out***.println("정답!! " + count + "번만에 맞춤!");

**break**;

}

}

*makeFile*(count);

// IOException 일어날수 있는 부분

}

**catch**(InputMismatchException e) {

// 이중 catch 문

System.***out***.println("InputMismatchException 발생");

}

**catch**(IOException e) {

System.***out***.println("IOException 발생");

}

**finally** {

**try** {

// try\_catch\_finally 구문의 중첩

System.***out***.println("다시하시겠습니까(y/n)>");

c = scanner.next().charAt(0);

// InputMismatchException 일어날수 있는 부분

}

**catch**(InputMismatchException e2) {

System.***out***.println("InputMismatchException 발생");

}

**finally** {

**if**(c == 'y') {

count = 0;

**continue**;

}

**if**(c == 'n') {

**break**;

}

}

}

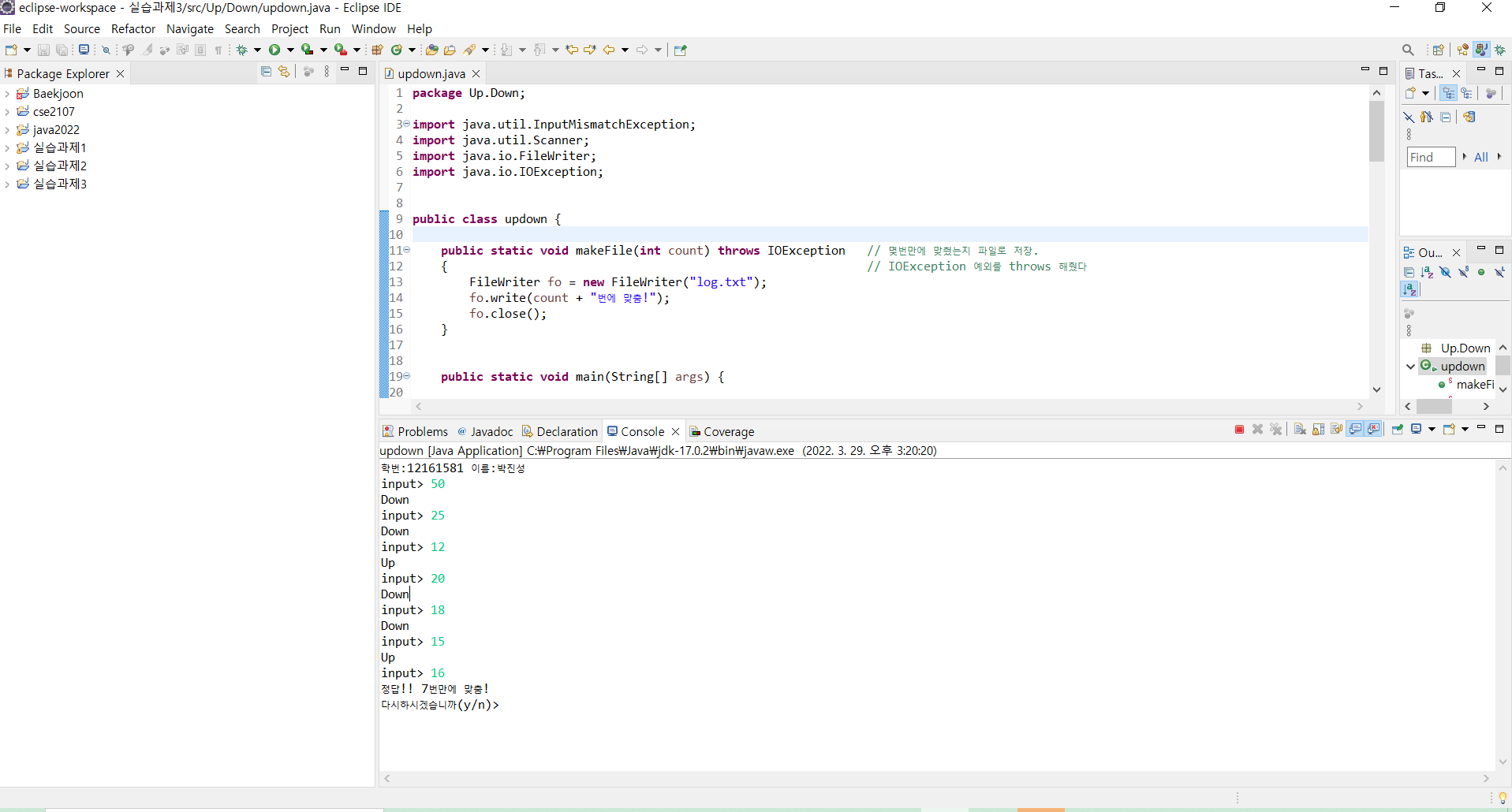
}

scanner.close();

}

}

**<3> 실행 결과**

****

<실행화면>

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

<디버깅 화면>

**<4> 결론 및 배운점**

우선, 자바에서의 예외처리를 직접 구현함으로써,

여러 예외처리의 종류와 함께 구현법도 습득하였다.

그리고 일부러 이중 catch 문과, try-catch-finally 구문의 중첩을 이용하여 사용하여

그 사용법을 정확히 익혔다.

예외처리를 보다 명시적으로 하고싶어서,

파일 만드는 부분을 메소드로 구현하여 throws 처리 해 주었다.

메소드 내부의 구문또한, 예외의 전파로 인해서 예외처리가 가능함도 확인할 수 있었다.