# 객체지향프로그래밍 – 11주차 실습 활동지(2020년 11월 11일)

성명: _____ 학과: _____ 학번: _____

※ 본 실습활동지에 대한 보고서나 코드를 작성함에 있어 교재나 강의노트를 제외한 다른 자료로부터 일부 또는 전체를 복사하였습니까? 예( ) 아니오( )

**제출방식:** 별도의 답안지 파일에 각 문제에 대하여 작성한 답안이나 작성한 코드 및 그 실행 결과를 복사해 넣는다. 답안지 문서명은 "report_주차_학번" (예, report_11_201811111)로 만든다. 답안지는 pdf형식으로 과제게시판(Bb)에 차주 월요일 23시59분까지 제출한다.

**I. Objectives**

1. Enum type에 대해 활용할 수 있다.
2. Exception의 용도를 이해한다.
3. Checked Exception과 unchecked exception의 차이를 이해한다.
4. Exception의 처리구조와 종료모델을 이해한다.

**II. Exercises (15점)**

1. **첨부한 Card 관련 class들을 실행해 본 후 Card의 suit를 String type에서 다음과 같은 Suit(enum type)으로 변경한 후 Card 및 CardDeck class를 적절하게 변경하시오. 단, CardTest.java는 변경해서는 안되며 결과가 동일하게 나와야 한다. 힌트) Card와 CardDeck class의 suit 변수의 type과 관련된 부분을 최소한으로 변경할 것 (4점)**

```
// Suit.java
public enum Suit { SPADE, DIAMOND, HEART, CLUB }
```

2. **다음 코드(첨부파일: DivideByZeroNoExceptionHandling.java)를 실행해보시오(입력데이터 100 7). 입력데이터를 다음과 같이 입력한 후 실행 결과가 어떻게 나오는지 설명하시오. (실행결과 및 설명) [1점]**

가) 100 0

나) 100 hello

3. **다음 코드(첨부파일: DivideByZeroWithExceptionHandling.java)를 실행하여 다음 각 입력을 제공한 후 수행 결과를 어떻게 나오는지 확인한 후 3번 문제와 비교하여 설명하시오. (실행결과 및 설명) [1점]**

가) 100 0
100 7

나) 100 hello
100 7

4. **3번의 numerator, denominator 변수 및 quotient 메소드의 return type을 모두 double type으로 변경한 후 수행해보시오. (4점)**

가) 어떤 현상이 생기는지 왜 생기는지 설명하시오.

100.0 0.0
100.0 7.0

나) ArithmeticException의 subclass로 DivideByZeroException을 정의하여 0.0으로 나누게 될 때 DivideByZeroException을 직접 throw하고 catch하도록 수정하시오.

5. **3번의 DivideByZeroWithException.Java 에서,** try block 바로 아래 다음과 같은 코드를 삽입하면 어떤 현상이 생기는지 보이고 왜 그런지 설명하시오. **(설명만 포함) [1점]**

```
catch(RuntimeException re)
{
    System.err.printf("%nException: %s%n", re);
}
```

6. **다음 프로그램을 컴파일시키면 오류가 발생한다. 이유를 설명하시오. 정상적으로 동작하기 위해 필요한 exception 처리를 아래 두 가지 방식으로 제공하시오. (오류 및 이유 설명, 가), 나)조건에 따라 작성한 정상적인 코드와 각각의 실행화면) [4점]**

```
import java.io.*;

class FileInputTest {

  public static FileInputStream f1(String fileName)
  {

    FileInputStream fis = new FileInputStream(fileName);


    System.out.println("f1: File input stream created");
    return fis;
  }


 public static void main(String args[])
  {
    FileInputStream fis1 = null;
    String fileName = "foo.bar";

   System.out.println("main: Starting " + FileInputTest.class.getName()
                      + " with file name = " + fileName);

   fis1 = f1(fileName);


   System.out.println("main: " + FileInputTest.class.getName() + " ended");
  }
}
```

가) Exception을 발생을 직접 처리하지 않는다. 즉, Java 시스템이 프로그램을 강제 종료시킨다.

나) Exception의 발생을 직접 처리한다. 처리방식은 Exception이 발생했다는 사실을 출력한 후 프로그램을 종료한다.