객체지향프로그래밍 - 11주차 실습 답안지

성명: 조우연 학과: 수학과 학번: 201921195

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

1. 첨부한 Card관련 class들을 실행해 본 후 Card의 suit를 String type에서 Suit(enum type)으로 변경 후 Card 및 CardDeck class를 적절하게 변경하시오.

- 변경 전 실행결과

```
After initializing...
ace of spade : 2 of spade : 3 of spade : 4 of spade :
5 of spade : 6 of spade : 7 of spade : 8 of spade :
9 of spade : 10 of spade : jack of spade : queen of spade :
king of spade : ace of diamond : 2 of diamond : 3 of diamond :
4 of diamond : 5 of diamond : 6 of diamond : 7 of diamond :
8 of diamond : 9 of diamond : 10 of diamond : jack of diamond :
queen of diamond : king of diamond : ace of heart : 2 of heart :
3 of heart : 4 of heart : 5 of heart : 6 of heart :
7 of heart : 8 of heart : 9 of heart : 10 of heart :
jack of heart : queen of heart : king of heart : ace of club :
2 of club : 3 of club : 4 of club : 5 of club :
6 of club : 7 of club : 8 of club : 9 of club :
10 of club : jack of club : queen of club : king of club :

After Shuffling...
8 of spade : 10 of diamond : ace of spade : 3 of spade :
3 of heart : ace of diamond : 8 of diamond : 5 of heart :
2 of diamond : 6 of heart : ace of heart : 5 of diamond :
7 of spade : 9 of spade : queen of diamond : 4 of heart :
8 of heart : 9 of heart : king of diamond : 5 of spade :
jack of diamond : 10 of heart : 10 of spade : jack of heart :
queen of heart : 4 of club : 4 of spade : king of heart :
8 of club : 6 of spade : jack of spade : 7 of diamond :
queen of spade : ace of club : 4 of diamond : 9 of club :
2 of spade : 6 of club : king of spade : 7 of club :
3 of diamond : 2 of club : jack of club : king of club :
7 of heart : 5 of club : 10 of club : 9 of diamond :
queen of club : 6 of diamond : 2 of heart : 3 of club :

After Sorting...
2 of club : 2 of heart : 2 of diamond : 2 of spade :
3 of club : 3 of heart : 3 of diamond : 3 of spade :
4 of club : 4 of heart : 4 of diamond : 4 of spade :
5 of club : 5 of heart : 5 of diamond : 5 of spade :
6 of club : 6 of heart : 6 of diamond : 6 of spade :
7 of club : 7 of heart : 7 of diamond : 7 of spade :
8 of club : 8 of heart : 8 of diamond : 8 of spade :
9 of club : 9 of heart : 9 of diamond : 9 of spade :
10 of club : 10 of heart : 10 of diamond : 10 of spade :
jack of club : jack of heart : jack of diamond : jack of spade :
queen of club : queen of heart : queen of diamond : queen of spade :
king of club : king of heart : king of diamond : king of spade :
ace of club : ace of heart : ace of diamond : ace of spade :
```

- Suit(enum type)으로 변경 후 실행결과

```
After initializing...
ace of SPADE : 2 of SPADE : 3 of SPADE : 4 of SPADE :
5 of SPADE : 6 of SPADE : 7 of SPADE : 8 of SPADE :
9 of SPADE : 10 of SPADE : jack of SPADE : queen of SPADE :
king of SPADE : ace of DIAMOND : 2 of DIAMOND : 3 of DIAMOND :
4 of DIAMOND : 5 of DIAMOND : 6 of DIAMOND : 7 of DIAMOND :
8 of DIAMOND : 9 of DIAMOND : 10 of DIAMOND : jack of DIAMOND :
queen of DIAMOND : king of DIAMOND : ace of HEART : 2 of HEART :
3 of HEART : 4 of HEART : 5 of HEART : 6 of HEART :
7 of HEART : 8 of HEART : 9 of HEART : 10 of HEART :
jack of HEART : queen of HEART : king of HEART : ace of CLUB :
2 of CLUB : 3 of CLUB : 4 of CLUB : 5 of CLUB :
6 of CLUB : 7 of CLUB : 8 of CLUB : 9 of CLUB :
10 of CLUB : jack of CLUB : queen of CLUB : king of CLUB :

After Shuffling...
2 of SPADE : 5 of SPADE : 8 of DIAMOND : 6 of DIAMOND :
10 of SPADE : 4 of HEART : 5 of DIAMOND : 7 of HEART :
jack of DIAMOND : jack of SPADE : 9 of HEART : 10 of CLUB :
queen of SPADE : 3 of CLUB : 7 of SPADE : 5 of HEART :
8 of CLUB : queen of CLUB : 9 of DIAMOND : 9 of SPADE :
2 of DIAMOND : 5 of CLUB : 10 of HEART : king of DIAMOND :
king of HEART : 4 of DIAMOND : jack of HEART : king of SPADE :
ace of SPADE : king of CLUB : 4 of SPADE : 6 of CLUB :
8 of HEART : 3 of SPADE : 8 of SPADE : ace of DIAMOND :
queen of DIAMOND : 2 of CLUB : 6 of HEART : 9 of CLUB :
6 of SPADE : 4 of CLUB : ace of CLUB : 7 of CLUB :
7 of DIAMOND : ace of HEART : queen of HEART : 2 of HEART :
10 of DIAMOND : 3 of DIAMOND : jack of CLUB : 3 of HEART :

After Sorting...
2 of CLUB : 2 of HEART : 2 of DIAMOND : 2 of SPADE :
3 of CLUB : 3 of HEART : 3 of DIAMOND : 3 of SPADE :
4 of CLUB : 4 of HEART : 4 of DIAMOND : 4 of SPADE :
5 of CLUB : 5 of HEART : 5 of DIAMOND : 5 of SPADE :
6 of CLUB : 6 of HEART : 6 of DIAMOND : 6 of SPADE :
7 of CLUB : 7 of HEART : 7 of DIAMOND : 7 of SPADE :
8 of CLUB : 8 of HEART : 8 of DIAMOND : 8 of SPADE :
9 of CLUB : 9 of HEART : 9 of DIAMOND : 9 of SPADE :
10 of CLUB : 10 of HEART : 10 of DIAMOND : 10 of SPADE :
jack of CLUB : jack of HEART : jack of DIAMOND : jack of SPADE :
queen of CLUB : queen of HEART : queen of DIAMOND : queen of SPADE :
king of CLUB : king of HEART : king of DIAMOND : king of SPADE :
ace of CLUB : ace of HEART : ace of DIAMOND : ace of SPADE :
```

- 수정된 코드

```
public class Card implements Comparable<Card>
{
    private int rank;
    private Suit suit;
    static final Suit[] cardSuits = { Suit.SPADE, Suit.DIAMOND, Suit.HEART, Suit.CLUB };

    public Card(int r, Suit s)
    {
        rank = r;
        suit = s;
    }
    public int getRank()
    {
       return rank;
    }
    public Suit getSuit()
    {
       return suit;
    }
```

```
public class CardDeck
{
    private static final int N = 52;
       private Card[] deck;
    private int top=0;
    private Suit suit;
    static final Suit[] cardSuits = { Suit.SPADE, Suit.DIAMOND, Suit.HEART, Suit.CLUB };
```

-> 첨부된 코드 이외의 코드는 모두 동일하다.

2. DivideByZeroNoExceptionHandling.java를 실행해보시오(입력데이터 100 7)

(가) 입력데이터 100 0

- 실행결과

```
Please enter an integer numerator: 100
Please enter an integer denominator: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lab_B_11_201921195.DivideByZeroNoExceptionHandling.quotient(DivideByZeroNoExceptionHandling.java:12)
        at lab_B_11_201921195.DivideByZeroNoExceptionHandling.main(DivideByZeroNoExceptionHandling.java:24)
```

- 이유

0으로 숫자를 나누면 quotient method의 "numerator / denominator"에서 ArithmeticException이 발생한다. ArithmeticException은 RuntimeException, 즉 unchecked exception이므로 컴파일타임에서 exception에 대한 핸들링을 따로 확인하지 않는다. 따라서 match되는 catch가 존재하면 해당 catch를 실행하지만, match되는 catch가 존재하지 않으므로, Exception발생을 직접 처리하지 않고, java시스템이 프로그램을 강제 종료시킨다.

(나) 입력데이터 100 hello

- 실행결과

```
Please enter an integer numerator: 100
Please enter an integer denominator: hello
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at lab_B_11_201921195.DivideByZeroNoExceptionHandling.main(DivideByZeroNoExceptionHandling.java:22)
```

- 이유

int가 아닌 String 타입으로 숫자를 나누면 InputMismatchException이 발생한다. InputMismatchException은 RuntimeException, 즉 unchecked exception이므로 컴파일타임에서 exception에 대한 핸들링을 따로 확인하지 않는다. 따라서 match되는 catch가 존재하면 해당 catch를 실행하지만, match되는 catch가 존재하지 않으므로, Exception발생을 직접 처리하지 않고, java시스템이 프로그램을 강제 종료시킨다.

3. DivideByZeroWithExceptionHandling.java를 실행하여 다음 각 입력을 제공한 후 수행결과를 어떻게 나오는지 확인하고, 3번 문제와 비교하여 설명하시오.

(가) 입력데이터 100 0, 100 7

- 실행결과

```
Please enter an integer numerator: 100
Please enter an integer denominator: 0

Exception: java.lang.ArithmeticException: / by zero
Zero is an invalid denominator. Please try again.

Please enter an integer numerator: 100
Please enter an integer denominator: 7

Result: 100 / 7 = 14
```

- 이유

3번과 같이 0으로 숫자를 나누면 quotient method의 "numerator / denominator"에서 Arithmetic Exception이 발생한다. 하지만 3번과 달리 match되는 catch가 존재하므로 해당 catch를 실행하여 exception에 대한 핸들링을 직접해준다. 또한 자바 시스템이 프로그램을 강제종료 시키는 것이 아니기 때문에 다시 올바른 값을 입력받아 프로그램이 정상적으로 작동할 수 있도록 핸들링되었다. 100/7이 14가 나오는 것은 int형들의 나눗셈은 int형 결과를 내므로, 100에서 14로 나누었을 때의 몫만 계산되고, 이하 소수점은 버린다.

(나) 입력데이터 100 hello, 100 7

- 실행결과

```
Please enter an integer numerator: 100
Please enter an integer denominator: hello

Exception: java.util.InputMismatchException
You must enter integers. Please try again.

Please enter an integer numerator: 100
Please enter an integer denominator: 7

Result: 100 / 7 = 14
```

- 이유

int가 아닌 String 타입으로 숫자를 나누면 InputMismatchException이 발생한다. 하지만 3번과 달리 match되는 catch가 존재하므로 해당 catch를 실행하여 exception에 대한 핸들링을 직접해준다. 또한 자바 시스템이 프로그램을 강제종료 시키는 것이 아니기 때문에 다시 올바른 값을 입력받아 프로그램이 정상적으로 작동할 수 있도록 핸들링되었다. 100/7이 14가 나오는 것은 위와 같은 이유이다.

4. 3번의 numerator, denominator 변수 및 queotient 메소드의 return type을 모두 double type으로 변경한 후 수행해보시오.

(가) 어떤 현상이 생기는지 왜 생기는지 설명하시오.

- 실행결과

```
Please enter an integer numerator: 100.0
Please enter an integer denominator: 0.0

Result: 100.000000 / 0.000000 = Infinity
```

계산결과로 Infinity가 출력된다.

- 이유

java에서 double은 IEEE 754표준을 구현한다. 이 표준은 0으로 나누면 "무한"이라는 값을 반환하도록 되어있기 때문에 double을 0으로 나누면 Infinity를 표시한다.

- 실행결과

```
Please enter an integer numerator: 100.0
Please enter an integer denominator: 7.0

Result: 100.000000 / 7.000000 = 14.285714
```

- 이유

int와 int를 나누었을 때와 달리, double과 double을 나누었을 때에는 계산값이 double형으로 계산되므로, 몫만 계산되는 것이 아니라 소수점까지 계산이 된다. 또한 소수점의 자리수를 임의로 설정해주지 않으면 기본적으로 소수점 6자리까지 출력된다.

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

- 수정한 코드

(catch는 기존의 catch (ArithmeticException arithmeticException) 그대로 사용)

```
public class DivideByZeroException extends ArithmeticException
{
    public DivideByZeroException() {}
    public DivideByZeroException(String gripe)
    {
        super(gripe);
    }

}
  public static double quotient(double numerator, double denominator) throws ArithmeticException
  {
      if(denominator==0) throw new DivideByZeroException();
      return numerator / denominator; // possible division by zero
  }
```

- 수행결과

```
Please enter an integer numerator: 100.0
Please enter an integer denominator: 0.0

Exception: lab_B_11_201921195.DivideByZeroException
Zero is an invalid denominator. Please try again.

Please enter an integer numerator: 100.0
Please enter an integer denominator: 7.0

Result: 100.000000 / 7.000000 = 14.285714
```

5. 3번의 DivideByZeroWithException.java에서 try block 바로 아래 실습지에 주어진 코드를 삽입하면 어떤 현상이 생기는지 보이고 왜 그런지 설명하시오.

- 실행결과 -> 컴파일 오류 발생

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
	Unreachable catch block for InputMismatchException. It is already handled by the catch block for RuntimeException
	Unreachable catch block for ArithmeticException. It is already handled by the catch block for RuntimeException

	at lab_B_11_201921195.DivideByZeroWithExceptionHandling.main(DivideByZeroWithExceptionHandling.java:40)
```

- 이유

ArithmeticException과 InputMismatchException이 RuntimeException의 subclass이다. catch에서는 더 specific한 exception들을 먼저 실행하고, 더 general한 것을 나중에 실행해야한다. 하지만 more general한 catch(RuntimeException re)를 먼저 위치했으므로 컴파일 에러를 발생시킨다.

6. 실습지에 주어진 프로그램을 컴파일시키면 오류가 발생한다. 이유를 설명하시오. 정상적으로 동작하기 위해 필요한 exception 처리를 아래 두 가지 방식으로 제공하시오.

- 실행결과 -> 컴파일 오류 발생

```
main: Starting lab_B_11_201921195.FileInputTest with file name= foo.barException in thread "main"
java.lang.Error: Unresolved compilation problem:
        Unhandled exception type FileNotFoundException

        at lab_B_11_201921195.FileInputTest.f1(FileInputTest.java:8)
        at lab_B_11_201921195.FileInputTest.main(FileInputTest.java:18)
```

- 이유

FileInputStream fis = new FileInputStream(fileName)은 "IOException"이 발생할 가능성이 있는 코드이다. IOException은 checked exception이므로 method가 실행되기 위해 필요한 exception이 갖춰져 있는지를 컴파일타임에서 확인한다. 하지만 실습지에 주어진 코드에는 try-catch혹은 throws declaration이 갖춰져 있지 않으므로 컴파일 오류를 발생한다.

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

- 코드

```
public class FileInputTest {

    public static FileInputStream f1(String fileName) throws IOException
    {
        FileInputStream fis = new FileInputStream(fileName);

        System.out.println("f1: File input stream created");
        return fis;
    }
    public static void main(String[] args) throws IOException {
        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");
    }
}
```

- 실행결과

```
main: Starting lab_B_11_201921195.FileInputTest with file name= foo.bar
Exception in thread "main" java.io.FileNotFoundException: foo.bar (지정된 파일을 찾을 수 없습니다)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
        at lab_B_11_201921195.FileInputTest.f1(FileInputTest.java:8)
        at lab_B_11_201921195.FileInputTest.main(FileInputTest.java:19)
```

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

- 코드

```
public class FileInputTest {

    public static FileInputStream f1(String fileName) throws IOException
    {
        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);
        try {
            fis1 = f1(fileName);
        }
        catch(IOException Ex){
            System.err.printf("%nException: %s%n", Ex);
            return;
        }

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

- 실행결과

```
main: Starting lab_B_11_201921195.FileInputTest with file name= foo.bar

Exception: java.io.FileNotFoundException: foo.bar (지정된 파일을 찾을 수 없습니다)
```