문제풀이 3

[8-1] 예외처리의 정의와 목적에 대해서 설명하시오.

⇒ 발생할 수 있는 예외를 코드 상 정의하여 처리하는 것.

⇒ 프로그램이 예외(Exception) 에 의해 중단되지 않고 계속 실행될 수 있도록 하기 위함이다.

[정의] 프로그램 실행 시 발생할 수 있는 예외의 발생에 대비한 코드를 작성하는 것

[목적] 프로그램의 비정상 종료를 막고, 정상적인 실행 상태를 유지하는 것

[8-2] 다음은 실행도중 예외가 발생하여 화면에 출력된 내용이다. 이에 대한 설명 중 옳지 않은 것은?

```
java.lang.ArithmeticException : / by zero
  at ExceptionEx18.method2 (ExceptionEx18.java:12)
  at ExceptionEx18.method1 (ExceptionEx18.java:8)
  at ExceptionEx18.main (Exception18.java:4)
```

1. 위의 내용으로 예외가 발생했을 당시 호출스택에 존재했던 메서드를 알 수 있다.
2. 예외가 발생한 위치는 method2 메서드이며, ExceptionEx18.java 파일의 12번째 줄이다.
3. 발생한 예외는 ArithmeticException 이며, 0으로 나누어서 예외가 발생했다.
4. ~~method2 메서드가 method1 메서드를 호출하였고~~, 그 위치는 ExceptionEx18.java 파일의 8번째 줄이다.

[8-3] 다음 중 오버라이딩이 잘못된 것은? (모두 고르시오)

⇒ **오버라이딩을 할 때, 조상 클래스의 메서드보다 많은 수의 예외를 선언할 수 없다.**

⇒ 답 : d , e

```
void add (int a, int b)
  throws InvalidNumberException, NotANumberException {}

class NumberException extends Exception {}
class InvalidNumberException extends NUmberException {}
class NotANumberException extends NumberException {}
```

```
// a
void add(int a, int b) throws InvalidNumberException, NotANumberException {}
// b
void add(int a, int b) throws InvalidNumberException {}
// c
void add(int a, int b) throws NotANumberException {}
// d
void add(int a, int b) throws Exception {}
// e
void add(int a, int b) throws NumberException {}
```

[8-4] 다음과 같은 메서드가 있을 때, 예외를 잘못 처리한 것은? (모두 고르시오)

⇒ c : Exception 예외의 최고조상이므로, 가장 마지막에 선언해야한다.

```
void method ()
  throws InvalidNumberException, NotANumberException {}

class NumberException extends RuntimeException {}
class InvalidNumberException extends NUmberException {}
class NotANumberException extends NumberException {}
```

```
// a
try {method();} catch(Exception e) {}
// b
try {method();} catch(NumberException e) {} catch(Exception e) {}
// c
try {method();} catch(Exception e) {} catch(NumberException e) {}
// d
try {method();} catch(InvalidNumberException e) {} catch(NotANumberException e) {}
// e
try {method();} catch(NumberException e) {}
// f
try {method();} catch(RuntimeException e) {}
```

[8-5] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

```
class Excercise8_5 {
  static void method(boolean b) {
    try {
      System.out.println(1);
      if (b) throw new ArithmeticException();
      System.out.println(2);
    } catch (RuntimeException r) {
        System.out.println(3);
        return;
      } catch (Exception e) {
```

```
        System.out.println(4);
        return
      } finally {
        System.out.println(5);
      }
    System.out.println(6);
  }
  public static void main(String args[]) {
    method(true);
    method(false);
  }
}
```

⇒ **1 3 5 1 2 5 6**

- **ArithemticException 은 RuntimeException 의 자손**

[8-6] 아래의 코드가 수행되었을 때의 실행결과를 적으시오

```
class Exercise8_6 {
  public static void main(String[] args) {
    try {
      method1();
    } catch (Exception e) {
      System.out.pritnln(5);
    }
  }
  static void method1() {
    try {
      method2();
      System.out.println(1);
  } catch (ArithmeticException e) {
      System.out.println(2);
  } finally {
      System.out.println(3);
  }
    System.out.println(4);
}
  static void method2() {
    throw new NullPointerException();
}
```

⇒ **3 5**

[8-7] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

```
public class Excercise8_7 {
    static void method(boolean b) {
        try {
            System.out.println(1);
            if (b) System.exit(0);
```

```
            System.out.println(2);
        } catch (RuntimeException r) {
            System.out.println(3);
            return;
        } catch (Exception e) {
            System.out.println(4);
            return;
        } finally {
            System.out.println(5);
        }

        System.out.println(6);
    }
    public static void main(String args[]) {
        method(true);
        method(false);
    }
}
```

⇒ 1

⇒ System.exit(0) 이 수행되어 프로그램이 즉시 종료된다. 이 경우, finally 블럭이 수행되지 않는다.

[8-8] 다음은 1~100 사이의 숫자를 맞추는 게임을 실행하던 도중에 숫자가 아닌 영문자를 넣어서 발생한 예외이다. 예외처리를 해서 숫자가 아닌 값을 입력했을 때는 다시 입력을 받도록 보완하라.

```
import java.util.*;


public class Exercise8_8 {
    public static void main(String[] args)
    {
        int answer = (int)(Math.random() * 100 ) + 1;
        int input = 0;
        int count = 0;

        do {
            count++;
            System.out.print("1 과 100 사이의 값을 입력하세요 : ");


            try {
                input = new Scanner(System.in).nextInt();

                if (answer > input) {
                    System.out.println("더 큰 수를 입력하세요.");
                } else if (answer < input) {
                    System.out.println("더 작은 수를 입력해주세요.");
                } else {
                    System.out.println("맞췄습니다. ");
                    System.out.println("시도횟수는 " + count + "번입니다.");
```

```
                    break;
                }

            } catch (InputMismatchException e) {
                System.out.println("유효하지 않은 값입니다. 다시 값을 입력해주세요.");
            }
        } while(true);
    }
}
```

[8-9] 다음과 같은 조건의 예외 클래스를 작성하고 테스트하시오.

- 클래스명 : UnsupportedFunctionException
- 조상 클래스명 : RuntimeException
- 멤버 변수
  - 이름 : ERR_CODE
  - 저장값 : 에러코드
  - 타입 : int
  - 기본값 : 100
  - 제어자 : final private
- 메서드
  1. 메서드명 : getErrorCode

     기능 : 에러코드(ERR_CODE)를 반환한다.

     반환타입 : int

     매개변수 : 없음

     제어자 : Public
  2. 메서드명 : getMessage

     기능 : 메세지의 내용을 반환한다. (Exception 클래스의 getMessage() 를 오버라이딩)

     반환타입 : String

     매개변수 : 없음

     제어자 : public

```
public class Excercise8_9 {
    public static void main(String args[]) throws Exception {
```

```
        throw new UnsupportedFunctionException("지원하지 않는 기능입니다.", 100);
    }
}


class UnsupportedFunctionException extends RuntimeException {
    final private int ERR_CODE;

    UnsupportedFunctionException(String msg, int error) {
        super(msg);
        this.ERR_CODE = error;
    }

    UnsupportedFunctionException(String msg) {
        super(msg);
        this.ERR_CODE = 100;
    }


    public int getErrorCode() {
        return ERR_CODE;
    }

    public String getMessage() {
        return "[" + ERR_CODE + "]" + super.getMessage();
    }
}
```

[8-10] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

```
public class Exercise8_10 {
    public static void main(String args[]) {
        try {
            method1();
            System.out.println(6);
        } catch (Exception e) {
            System.out.println(7);
        }
    }

    static void method1() throws Exception {
        try {
            method2();
            System.out.println(1);
        } catch(NullPointerException e) {
            System.out.println(2);
            throw e;
        } catch(Exception e) {
            System.out.println(3);
        } finally {
            System.out.println(4);
        }
        System.out.println(5);
    }
```

```
    static void method2() {
        throw new NullPointerException();
    }
}
```

⇒ 2 4 7