
JDK8로 배우는 자바 프로그래밍
어서와
Java는
처음이지!
천인국 지음
False
True
INFINITY BOOKS

TEST BANK

CHAPTER 14

1. 오류와 발생되는 예외를 올바르게 짝지으시오.

- int[] list; list[0] = 0;
- 자바 가상 기계가 클래스를 찾을 수 없는 경우
- 파일을 읽던 프로그램이 파일의 끝에 도달한 경우
- 파일의 끝에 도달했는데도 파일을 읽으려고 시도

- 에러(error)
- 체크 예외(checked exception)
- 컴파일 오류
- 예외가 발생하지 않는다

2. ArrayIndexOutOfBoundsException 예외를 발생시키는 프로그램을 작성하고 try/catch 블록을 이용하여 처리하여 보라.

```
try {
    _____;
}
catch (_____) {
    _____; // 콘솔에 오류 메시지를 출력한다.
}
```

3. 다음과 같은 코드는 유효한 코드인가? 어떤 경우에 유용하게 사용되는가?

```
try {
    ...
} finally {
    ...
}
```

4. 다음과 같은 예외 처리기로 잡을 수 있는 예외는 어떤 종류인가? 그리고 이런 종류의 예외 처리기가 바람직하지 않은 이유는?

```
catch (Exception e) {
    ...
}
```

5. 다음 코드에서 잘못된 부분을 지적하시오.

```
try {
    ...
} catch (Exception e) {
    ...
} catch (ArithmeticException a) {
    ...
}
```

6. 다음 프로그램의 출력을 쓰시오.

```
try {
```

```
        int n = Integer.parseInt("abc");
        System.out.println("try");
    }
    catch (NumberFormatException e){
        System.out.println("숫자 형식 오류");
    }
    finally {
        System.out.println("finally");
    }
```

7. 다음 프로그램의 출력을 쓰시오.

```
    try {
         int[] array=new int[-10];
         System.out.println("try");
    }
    catch (NumberFormatException e){
         System.out.println("숫자 형식 오류");
    }
    catch (NegativeArraySizeException e){
         System.out.println("배열 크기 음수 오류");
    }
    catch (Exception e){
         System.out.println("오류");
    }
```

8. 다음 프로그램의 출력을 쓰시오.

```
public class Test
{
    public static void throwit ()
    {
        System.out.print("A ");
        throw new RuntimeException();
    }
    public static void main(String [] args)
    {
        try
        {
            System.out.print("B ");
            throwit();
        }
        catch (Exception re )
        {
            System.out.print("C ");
        }
        finally
```

```
        {
            System.out.print("D ");
        }
        System.out.println("E ");
    }
}
```

9. 다음 프로그램의 출력을 쓰시오.

```
public class Test
{
    public static void main(String [] args)
    {
        try
        {
            someMethod();
            System.out.print("A");
        }
        catch (Exception ex)
        {
            System.out.print("B");
        }
        finally
        {
            System.out.print("C");
        }
        System.out.print("D");
    }
    public static void someMethod() {}
}
```

10. 다음 프로그램을 컴파일이 되도록 올바르게 수정하시오.

```
try
{
    int x = 0;
    int y = 5 / x;
}
catch (Exception e)
{
    System.out.println("Exception");
}
catch (ArithmeticException ae)
{
    System.out.println(" Arithmetic Exception");
}
```

```
System.out.println("finished");
```

11. 다음 프로그램의 출력은?

```
public class Test
{
    public static void main(String args[])
    {
        try
        {
            System.out.print("Hello world ");
        }
        finally
        {
            System.out.println("Finally executing ");
        }
    }
}
```

12. 다음과 같은 문장에 의하여 발생하는 예외는 무엇인가?

(1) int[] anArray = new int[3];
System.out.println(anArray[3]);

(2) String[] strs = new String[3];
System.out.println(strs[0].length());

(3) Integer.parseInt("abc");

(4) Object o = new Object();
Integer i = (Integer)o;

13. 다음 프로그램을 컴파일하여 보자.

```
public class Test {
        public static void main(String[] args) {
                sub();
        }
        public static void sub() {
                int[] array = new int[10];
                int i = array[10];
        }
}
```

(1) 위의 프로그램은 컴파일 시에 오류가 발생한다. 어떤 오류가 발생하는가?

(2) try/catch 블록을 사용하여서 예외를 처리하여 보라.

(3) throws 선언을 이용하여 예외를 처리하여 보라.

14. 다음은 사용자 예외를 정의하고 사용하는 간단한 예이다. checkNegative() 메소드는 음수가 전달되면 사용자 예외를 발생한다. 빈칸을 채우고 컴파일하여 실행하여 보자.

```
class MyException extends ____________  {
    public MyException(String message) { super(message); }
}
public class MyExceptionTest {
    public static void checkNegative(int number) throws MyException {
        if (number < 0) {
            __________(new MyException("음수는 안됩니다."));
        }
        System.out.println("다행히 음수가 아니군요");
    }
    public static void main(String[] args) {
        try {
            checkNegative(1);
            checkNegative(-1);
        } catch (MyException ex) {    ex.printStackTrace();    }
    }
}
```

15. 숫자를 저장하고 있는 배열을 받아서 지정된 숫자를 찾는 searchArray() 정적 메소드를 작성하라. 만약 배열 안에 원하는 숫자가 없으면 NotFoundExecption 예외를 발생한다. NotFoundExecption 클래스는 사용자 정의 예외로서 Exception 클래스를 상속받아서 정의하라. searchArray() 메소드에서 발생되는 예외를 try/catch를 이용하여 처리하여 보라.

16. 은행 예금을 나타내는 클래스 BankAccount를 작성하라. BankAccount는 잔액(balance)을 필드로 가지며, 입금을 나타내는 deposit() 메소드와 출금을 나타내는 withdraw() 메소드를 가진다. withdraw()에서 인출 금액이 잔액보다 크면 NegativeBalanceException을 발생한다. BankAccount클래스를 테스트하는 BankAccountTest 클래스를 작성하고 발생되는 예외를 try/catch를 이용하여 처리하여 보라.