

**Q1** 다음 예외 발생 코드와 예외의 종류를 연결하시오.

<code>int num = Integer.parseInt("10!");</code>	①	①	a	<code>ArrayIndexOutOfBoundsException</code>
<code>String a = null;</code>	②	②	b	<code>ClassNotFoundException</code>
<code>System.out.println(a.charAt(2));</code>	③	③	c	<code>NumberFormatException</code>
<code>System.out.println(3/0);</code>	④	④	d	<code>InterruptedException</code>
<code>new FileInputStream("text.txt");</code>	⑤	⑤	e	<code>FileNotFoundException</code>
<code>Thread.sleep(1000);</code>	⑥	⑥	f	<code>NullPointerException</code>
<code>Class.forName("java.lang.Object");</code>	⑦	⑦	g	<code>ArithmeticException</code>
<code>int[] a = {1, 2, 3};</code>				
<code>System.out.println(a[3]);</code>				

**Q2** 다음 코드의 try{} 구문과 catch(){} 구문에는 공통적인 코드가 포함돼 있다. finally{} 블록을 사용해 중복을 제거한 코드를 작성하시오.

```

try {
    int a = 3;
    System.out.println(5 / a);
    System.out.println("출력 내용 1");
    System.out.println("출력 내용 2");
    System.out.println("출력 내용 3");
} catch (ArithmeticException e) {
    System.out.println("예외 발생");
    System.out.println("출력 내용 1");
    System.out.println("출력 내용 2");
    System.out.println("출력 내용 3");
}

```

```

try {
    int a = 3;
    System.out.println(5/a);
} catch (ArithmeticException e) {
    System.out.println("예외발생");
} finally {
    System.out.println("출력내용 1");
    System.out.println("출력내용 2");
    System.out.println("출력내용 3");
}

```

**Q3** 다음은 2개의 try-catch-finally 구문으로 만든 예외 처리 코드다. 다중 catch 구문을 이용해 코드를 1개의 try-catch-finally 구문으로 수정하시오.

```
try {
    int [] array = {1, 2, 3};
    int index = 4;
    System.out.println(array[index]);
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("배열값 읽기 실패");
} finally {
    System.out.println("처리 완료");
}

try {
    A aa = new A();
    B bb = (B)aa;
} catch (ClassCastException e) {
    System.out.println("클래스 다운캐스팅 실패");
} finally {
    System.out.println("처리 완료");
}
```

```
try {
    int [] array = {1, 2, 3};
    int index = 4;
    System.out.println(array[index]);
    A aa = new A();
    B bb = (B)aa;
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("배열값 읽기 실패");
} catch (ClassCastException e) {
    System.out.println("클래스 다운캐스팅 실패");
} finally {
    System.out.println("처리완료");
}
```

**Q4** 다음 예외 처리 구문은 오류를 포함하고 있다. 오류가 발생한 이유와 그 해결책을 쓰시오.

```
try {
    int[] array = new int[] {1, 2, 3};
    System.out.println(array[3]);
} catch (Exception e) {
    System.out.println("다른 예외가 발생했습니다.");
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("배열 인덱스의 사용 범위를 벗어났습니다");
}
```

오류가 발생한 이유	해결책
항상 Exception에 걸리기 때문에 아래쪽 catch 구문은 unreachable code가 됨	Exception catch 블록과 ArrayIndexOutOfBoundsException 블록의 순서를 바꿈

**Q5** 다음은 try-with-resource 구문을 사용해 자동으로 리소스를 해제할 수 있도록 한 코드다. 코드의 빈칸을 완성하시오.

```
class A implements AutoCloseable {  
    String res = "리소스 할당";  
    @Override  
    public void close() throws Exception {  
        res = null;  
        System.out.println("리소스 자동 해제");  
    }  
}  
  
public static void main(String[] args) {  
    try (A b = new A()) {  
        // ...  
    } catch (Exception e) {  
        // ...  
    }  
}
```

실행 결과

리소스 자동 해제

**Q6** 클래스 A 내부에는 abc() 메서드와 bcd() 메서드가 있으며, bcd() 메서드는 예외 처리 구문을 포함하고 있다.

```
class A {  
    void abc() {  
        bcd();  
    }  
    void bcd() {  
        try {  
            Thread.sleep(1000);  
            Class.forName("java.lang.Object");  
        } catch (InterruptedException | ClassNotFoundException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

이때 bcd()가 예외를 직접 처리하지 않고 전가할 때의 코드를 완성하시오.

```
class A {  
    void abc() {  
  
        try {  
            bcd();  
        } catch (ClassNotFoundException | InterruptedException e) {  
            e.printStackTrace();  
        }  
    }  
    void bcd() throws InterruptedException, ClassNotFoundException {  
        Thread.sleep(1000);  
        Class.forName("java.lang.Object");  
    }  
}
```

**Q7** 클래스 A는 학점이 3.0 미만일 때 사용자 정의 일반 예외(ScoreException)를 발생시키는 abc() 메서드를 포함하고 있다.

```
class ScoreException extends Exception {  
    public ScoreException() {  
    }  
    ScoreException(String s) {  
        super(s);  
    }  
}  
class A {  
    void abc(double score) throws ScoreException {  
        if(score >= 3.0) {  
            System.out.println("장학금 대상자입니다.");  
        } else {  
            throw new ScoreException("학점 미달입니다");  
        }  
    }  
}
```

이때 다음 코드의 실행 결과를 쓰시오.

```
public static void main(String[] args) {  
    A a = new A();  
    try {  
        a.abc(3.8);  
        a.abc(2.5);  
    } catch (ScoreException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

실행 결과

장학금 대상자입니다.  
학점 미달입니다