제 9 장 예외(실습)

□ 개념 확인 학습

1. 오류와 발생되는 예외를 올바르게 짝지으세요.

• int[] list; list[0] = 0;	• 에러(error)
• 자바 가상 기계가 클래스를 찾을 수 없는 경우	• 체크 예외(checked exception)
• 파일을 읽던 프로그램이 파일의 끝에 도달한 경우	• 컴파일 오류
• 파일의 끝에 도달했는데도 파일을 읽으려고 시도	• 예외가 발생하지 않는다

2. 다음 프로그램의 "~~~" 부분에서 ArrayIndexOutOfBoundsException 예외를 발생시키는 프로그램을 작성하고 try/catch 블록을 이용하여 "배열의 인덱스를 확인해서 사용하세요"라는 문자열을 출력하도록 처리해 보세요.

```
public class ArrayIndexOutOfBoundsExceptionExample {
    public static void main(String[] args) {
        int[] array = new int[2];
        ~~~
    }
}
```

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

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

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

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

5. 다음 코드에서 잘못된 부분을 지적하세요.

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

6. 다음 프로그램의 출력은?

```
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 출력은?

```
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 출력은?

```
public class Test  {
    public static void throwit() throws RuntimeException {
        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 출력은?

```
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 프로그램의 출력은?

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

11. 다음과 같은 문장에 의하여 발생하는 예외는 무엇인가요?

```
(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;
```

12. 다음 코드가 실행되었을 때 출력 결과는 무엇입니까?

```
public class TryCatchFinallyExample {
   public static void main(String[] args) {
        String[] strArray = { "10", "2a" };
        int value = 0;
        for(int i=0; i<=2; i++) {
             try {
                  value = Integer.parseInt(strArray[i]);
             } catch(ArrayIndexOutOfBoundsException e) {
                  System.out.println("인덱스를 초과했음");
             } catch(NumberFormatException e) {
                  System.out.println("숫자로 변환할 수 없음");
             } finally {
                  System.out.println(value);
             }
        }
   }
}
```

□ 적용 확인 학습 & 응용 프로그래밍

13. 다음 프로그램을 컴파일 하세요.

```
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 발생하나요?

(2) try/catch 블록을 사용하여서 예외를 처리해 보세요.

(3) throws 선언을 이용하여 예외를 처리해 보세요.

14. 다음 프로그램을 컴파일이 되도록 올바르게 수정하세요.

```
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");
```

15. 다음은 사용자 예외를 정의하고 사용하는 간단한 예이다. checkNegative() 메소드는 음수가 전달되면 사용자 예외를 발생한다. 빈칸을 채워 컴파일하고 실행하세요.

```
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();  }
  }
}
```

16. 로그인 기능을 LoginExample 클래스의 login() 메소드에서 구현하려고 합니다. 존재하지 않은 ID를 입력했을 경우 NotExistIDException을 발생시키고, 잘못된 패스워드를 입력했을 경우 WrongPasswordException을 발생시키려고 합니다. 제시된 클래스의 구조를 보고 적절하게 변경 구현하여 프로그램을 테스트 하세요.

```
public class NotExistIDException {
  //~~~~~
}
public class WrongPasswordException {
  //~~~~
}

public class LoginExample
{
  public static void main(String[] args) {
      try { login("white", 12345); }
      catch(Exception e) { System.out.println(e.getMessage()); }

      try { login("blue", 54321); }
      catch(Exception e) { System.out.println(e.getMessage()); }
  }
  public static void login(String id, int password) {
      //id가 blue가 아니라면 NotExistIDException 발생시킴
      //~~~~~

      //password가 12345가 아니라면 WrongPasswordException 발생시킴
      //~~~~
  }
}
```

17. 필요한 클래스를 구현하고 프로그램을 테스트 하세요.

- 은행 예금을 나타내는 클래스 BankAccount는 잔액(private int balance)을 필드로 가지며, 입금을 나타내는 deposit() 메소드와 출금을 나타내는 withdraw() 메소드를 가진다.

- deposit() 메소드는 매개변수로 전달된 금액(amount)을 잔액에 추가한 후 추가된 잔액을 반환한다.

- withdraw() 메소드는 전달된 인출 금액(amount)이 현재의 잔액(balance)보다 작거나 같다면 현재의 잔액에서 인출 금액을 빼고 잔액을 반환 한다. 그러나 인출 금액(amount)이 현재의 잔액(balance)보다 크면 NegativeBalanceException("잔고가 음수입니다.")을 발생한다. NegativeBalanceException이 발생하면 현재의 잔액은 변경 없이 그대로 리턴 한다.

- BankAccountTest 클래스에서 BankAccount 객체를 생성하여 deposit(), withdraw()를 호출하여 다음 그림처럼 수행되도록 한다.