ＪａｖａプログラミングⅡ

９回目　例外処理　課題

**確認○×問題**

次の各文は正しいか誤っているか答えなさい。

(1)例外とは、プログラムの実行時に発生し、本来の処理の流れを妨げるエラーである
(2)例外は、Throwable クラスまたはそのサブクラスのオブジェクトにより表現される
(3)例外が発生した場合、本来の処理は中断される
(4)例外が発生した後、例外に対する適切な処理が行われ、プログラムは常に強制終了する
(5)例外処理は try 文を用いて記述する
(6)1 つの try 文で複数の例外に対する処理を記述することができる
(7)独自の例外クラスを作成することはできない
(8)例外を送出するにはキーワード throws を使用する

**■難易度★☆☆**

**課題１ 次のコードを実行すると、配列要素を確保して配列変数に代入する部分(7 行目)「ary = new int[-5]; 」で例外 NegativeArraySizeException （非検査例外クラス）が発生します。この例外は、負のサイズを持った配列要素を作成しようとした場合に発生します。try 文を用いてこの例外に対する例外処理を記述しなさい。下の実行例では例外処理でメッセージを出力しています。**

〔NegativeArraySizeException 例外を発生するコード〕

```
1:  class Assignment9_1
2:  {
3:     public static void main(String[] args)
4:     {
5:         int[] ary;
6:         // 例外 NegativeArraySizeException が発生
7:         ary=new int[-5];
8:     }
9:  }
```

〔例外処理前の実行結果〕

```
Exception in thread "main" java.lang.NegativeArraySizeException
        at Assignment9_1.main(Assignment9_1.java:7)
```

〔例外処理後の実行例〕

```
配列要素数の指定は負です
```

■難易度★★☆

**課題 2** 下に示しているコードはキーボードから整数を一つ読込むコードです。

① **Integer.paeseInt()**メソッドは文字列が表す整数値を **int** 型に変換します。もし、整数値以外が入力されると **NumberFormatException**（非検査例外クラス）という例外を送出します。整数値以外が入力された場合、この例外をキャッチしてメッセージ「整数値ではありません」を表示するように例外処理を加えなさい。

（補足）

**readLine()**メソッドは **IOException**（検査例外クラス）という例外を送出する可能性があります。この例外が発生した場合はこれまで通りに呼び出し元であるメインメソッドに例外処理を任せてしまいましょう。

```
public static void main(String[] args) throws IOException
{
    …
}
```

〔キーボードから整数を一つ読み込むコード〕

```
import java.io.*;
class Assignment9_2
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        int num=0;
        String str;

        System.out.println("整数を入力してください。");
        str=br.readLine();
        num=Integer.parseInt(str);
        System.out.println("入力された整数は"+num+"です。");
    }
}
```

② 実行例のように整数が入力されるまで繰り返すコードに変更しなさい。

〔実行例〕

```
整数を入力してください。
a
整数値ではありません。
もう一度入力してください。
r
整数値ではありません。
もう一度入力してください。
2
入力された整数は 2 です。
```

■難易度★★☆

**課題 3** 次は除算を管理するクラス Div の宣言です。メソッド calc()は整数 i と j を受け取り i / j を戻り値とします。ここで j がゼロであった場合には除算はできませんので、例外を送出するようにします。例外クラス Exception を拡張してこの例外を表すクラス（クラス名は各自で）を宣言しなさい。次に、j がゼロであった場合にはこの例外を送出するようにクラス Div の宣言を変更しなさい。

〔除算を管理するクラス Div〕

```
class Div
{
    public static double calc(int i, int j)
    {
        double ans;
        ans=(double)i/j;
        return ans;
    }
}
```

■難易度★★★

**課題 4** 課題 2 で作成した例外処理をもつ整数値のキーボード入力により 2 つの整数を入力して、課題 3 で作成した例外処理をもつクラスメソッド double calc(int,int)を用いて除算を計算するコードを main()メソッドに記述しなさい。ゼロでの除算を表す例外が発生しない場合は除算の結果を表示し、例外が発生した場合はこれをキャッチして無限大「∞」を表示するようにします。

〔実行例 1〕

```
整数 i÷整数 j を計算します。
整数値 i を入力してください。
4r
整数値ではありません。
もう一度入力してください。
4
整数値 j を入力してください。
2
4 / 2 = 2.0
```

〔実行例 2〕

```
整数 i÷整数 j を計算します。
整数値 i を入力してください。
4
整数値 j を入力してください。
0
4 / 0 = ∞
```

■難易度★★☆

**課題 5** 次のコードはキーボード入力の例です。この中の **readLine()**メソッドはコードの下に示すように、クラス **BufferedReader** の中で宣言されています。この宣言より、**realLine()**メソッドは **IOException**（検査例外クラス）例外を送出する可能性があることが分かります。これまでは、

```
    public static void main(String[] args) throws IOException
    {
        …
    }
```

として、**main()**メソッド内には例外処理を記述せず **main()**メソッドの呼び出し元に例外処理を任せていました。

以下のコードにおいて、

① 例外 **IOException** を **main()**メソッド内で処理するように例外処理を加えなさい

② **main()**メソッドに宣言してある **throws IOException** を消して、コンパイルできることを確認しなさい

〔キーボード入力の例〕

```
import java.io.*;

class Assignment9_5
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        String name;

        System.out.println("名前を入力してください");
        name = br.readLine();
        System.out.println("入力された名前は"+name+"です");
    }
}
```

〔readLine()メソッドの宣言〕

```
class BufferedReader{
        :
    public String readLine() throws IOException{
        :
    }
        :
}
```

機　能　1 行のテキストを読み込みます
戻り値　行の内容を含む文字列、ただし行の終端文字は含めない。
**例　外　IOException - 入出力エラーが発生した場合**