2023 年 12 月 14 日（木）実施

例外処理

Java 言語では，作成したプログラムを実行する際に，記述した処理が想定しない事態によって実行できなくなる場合を**例外**と呼び，その例外への対処，即ち**例外処理**が求められる。

例外処理を行うための try 文の一般形は次のようになる。

```
try {
   例外を発生させる可能性のある処理
} catch(例外のクラス名 1 変数 1 ) {
        例外に対処する処理 1
} catch(例外のクラス名 2 変数 2 ) {
        例外に対処する処理 2
            ・・・
} finally {
        try 文の最後に必ず実行される処理
}
```

ここで，catch 節と finally 節とは何れかを必ず記述する。catch 節は必要に応じて，例外を場合分けして複数記述することが出来る。finally 節には例外発生時に中断した処理の後処理等を記述する。なお，例外クラスの最も基本的なクラスは Exception なので，catch 節の引数に Exception を記述した場合には，全ての例外クラスに対応出来る。

例題 1（例外を発生させてみる－1）

次のプログラムは，標準入力装置から 2 個の整数を入力して，和，差，積，商，余りを求めて標準出力装置に表示するものである。これを入力，ビルドして，実行せよ。なお，実行は次の様に 3 回行う。また，その際に発生した例外に対するメッセージを確認する。

[1 回目の実行]　0 以外の整数を入力する。 ⇒ 通常，例外は発生しない。

[2 回目の実行]　1 つ目の整数を入力すべきところに整数以外の文字を入力してみる。

[3 回目の実行]　2 つ目の整数として 0 を入力してみる。

ここで，クラス名は Sample9_1，ソースファイル名は Sample9_1.java とする。

```
import java.util.InputMismatchException;
import java.util.Scanner;


public class Sample9_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
```

```
        final int NUM = 2;
        int[] x = {1, 1};

        for (int i=0; i<NUM; i++) {
            x[i] = inx(i);
        }

        System.out.printf("%d + %d => %d%n", x[0],x[1],sum(x[0],x[1]));
        System.out.printf("%d - %d => %d%n", x[0],x[1],diff(x[0],x[1]));
        System.out.printf("%d * %d => %d%n", x[0],x[1],prod(x[0],x[1]));
        System.out.printf("%d / %d => %d%n", x[0],x[1],quot(x[0],x[1]));
        System.out.printf("%d %% %d => %d%n", x[0],x[1],rem(x[0],x[1]));

    }

    private static int inx(int j){
        Scanner sc = new Scanner(System.in);
        int num = 0;

        try {
            System.out.print((j+1) +"つ目の整数を入力してください：");
            num = sc.nextInt();
        } catch (InputMismatchException err) {
            System.out.println("例外をキャッチ: "+err);
        }

        return num;
    }

    private static int sum(int a, int b)
    {
        return a+b;
    }

    private static int diff(int a, int b)
    {
        return a-b;
    }

    private static int prod(int a, int b)
    {
        return a*b;
    }

    private static int quot(int a, int b)
    {
        return a/b;
    }

    private static int rem(int a, int b)
    {
        return a%b;
    }

}
```

【解説】

1. try 文の中でデータをキーボードから受け取り，変数 num に代入する場合，データ入力時に例外が発生すると，num に値が入らなくなるので，try 文の前に num の初期値を設定しておく必要がある。
2. InputMismatchException は，キーボードから入力したデータの型が入力のメソッドと一致しない場合に発生する例外をキャッチするためのクラスである。

例題２（**例外を発生させてみる－2**）

バグ(プログラムの誤り)を含む次のプログラムを入力，ビルドして，実行せよ。ここで，クラス名は Sample9_2，ソースファイル名は Sample9_2.java とする。

```
import java.util.Scanner;


public class Sample9_2 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 5;
        int sum = 0;
        int[] x = {0, 0, 0, 0, 0};

        for (int i=0; i<NUM; i++) {
            try {
                x[i] = Integer.parseInt(inx(i));
            } catch (NumberFormatException exnf) {
                System.out.println("整数を入力してください");
                System.out.println("");
                i--;
            }
        }

        try {
            for (int i=0; i<=NUM; i++) {
                sum += x[i];
            }
        } catch (Exception e) {
            System.out.println("例外 "+e+" をキャッチ");
        } finally {
            System.out.println("合計は"+sum+"です。");
        }
    }

    private static String inx(int j){
        Scanner sc = new Scanner(System.in);
        String str;

        System.out.print((j+1) +"つ目の整数を入力してください：");
        str = sc.next();

        return str;
```

```
    }

}
```

【解説】

1. このプログラムの inx メソッドでは標準入力装置から入力されたデータは，文字列として変数に格納され，呼び出し側に返される。
2. NumberFormatException は，文字列を数値型に変換しようとした時，文字列の形式が正しくない場合に発生する例外をキャッチするためのクラスである。
3. finally 節は必ず実行されるため，例外が発生する直前の sum の値が表示される。
4. Integer.parseInt は，**プリミティブ型** int の値を**オブジェクト**にラップする，Integer クラスのメソッドである。parseInt メソッドは，文字列の引数を符号付き 10 進数の整数型として構文解析する。文字列にある文字は 1 番目の文字以外は全て，10 進数でなければならない。1 番目の文字は，負の値を表すためのマイナス記号の ASCII 文字 '-' (\u002d) とすることが可能である。以上の結果，生成された整数値が返される。クラスの階層は，次の通りである。

   java.lang.Object → java.lang.Number → java.lang.Integer

＊ プリミティブ型はオブジェクトではなく，フィールドやメソッドを持たない。そこで，それぞれのプリミティブ型に対応するラッパークラスが用意されており，Integer クラスは，そのうちの一つである。

＊＊ parseInt メソッドに対して，符号付き 10 進数以外の文字列を引き渡した場合には，例外が発生する。

演習

次のプログラムリストの空欄 1)　　～3)　　に適切な語句を埋めたプログラムを作成し，ビルドして実行せよ。ここで，プログラムのクラス名は Ex9，ソースプログラム名は Ex9.java とする。但し，このプログラムにはバグがあるので，挙動のおかしな部分を try 文で囲み(桃色の四角い枠内)，例外を発生させて検査しようとしている。

従って，実行は 2 回行い，先ず正の整数を 5 個入力して，どのような結果が得られるかを確認した上で，次に 5 個の整数として全て 0 を入力して，例外が発生することを確認する。また，その際に発生した例外に対するメッセージを確認する。

```
import java.util.Scanner;


public class Ex9 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 5;
        int sum = 0;
        int[] x = {0, 0, 0, 0, 0};
```

```
        double ave = 0.0;

        for (int i=0; i<NUM; i++) {
            try {
                x[i] = Integer.parseInt(inx(i));
            } catch (NumberFormatException exnf) {
                System.out.println("整数を入力してください");
                System.out.println("");
                i--;
            }
        }

        for (int i=0; i<NUM; i++) {
            sum += x[i];
        }

        try {
                ave = NUM / sum;
        } 1)       (Exception 2)       ) {
                System.out.println("例外 "+ex+" をキャッチ");
                ex.printStackTrace();
        } 3)       {
                System.out.println("");
                System.out.println("合計は"+sum+"です。");
                System.out.println("平均は"+ave+"です。");
        }
    }

    private static String inx(int j){
        Scanner sc = new Scanner(System.in);
        String str;

        System.out.print((j+1) +"つ目の整数を入力してください：");
        str = sc.next();

        return str;
    }
}
```

【解説】例外のインスタンスに対して printStackTrace メソッドを用いると，どのクラスの何行目でどのような例外が発生させられたかという，**スタックトレース**と呼ばれる情報が表示される。

提出物：

1) 例題 1，例題 2 及び演習のプログラムのコンソールへの出力結果をコピーして貼り付けたテキストファイル res9.txt をメールに添付する。
2) 演習で空欄に適切な語句を埋めたソースプログラムのファイル Ex9.java をメールに添付する。
3) 第 9 回の理解度確認用の質問ファイル Prog1_Questions_9th.txt に解答を記入して，メールに

添付する。

＊ メールの件名は『プログラミング１　第９回課題』（鍵括弧は要らない）とする。