Introduction to Java

Special Care Set for Java Begineers/ Ex 11

・Notice!
The motivation of this page is to lead regular exercises.
We prepare many useful information in this page as hints of exercises, but this page has no answer.
If you have question, please don't hesitate to ask and discuss your teacher and teaching assistants.
このページは、通常の演習問題への導きとして作られています。
多くの有用な情報をヒントとして準備しておりますが、ここに回答が隠されているとかはありません。
もし、演習に対して質問等ございましたら、遠慮なく教員・TAに質問・議論をしてください。

Example Problem: 例外処理とエラー
Exception Process and Error

プログラムを作成し動作させていると、皆さんも様々なバグや仕様に遭遇するでしょう。
特に、自分のプログラム自体にはさしたる間違いがないのにもかかわらず、外的要因（入出力データや内部のデータ受け渡し、メモリやVirtual Machine等の不具合等）によって途中まで上手く動いていたのにエラー終了してしまい、さてどこで変なエラーが起きてしまったのかと、もう一度動作させ直して時間が経つのを待つ…などと言ったことを経験した人は少なくはないのではないかと思います。

実際、業務アプリケーションを作成している時には、プログラミング仕様だけではなく、業務的な仕様、データ仕様、ユーザーのインタラクション等、無数にある操作の『組み合わせ』全てについて良く動く（停止せずに動作する）必要があります。そんな時に、『想定外』なことを定義して、その部分には特殊な動作をさせるという処理を『例外処理』と呼んでいます。
また、JavaのVirtual Machineが停止するような致命的な動作不良を『エラー』と呼んでいます。

これら、例外処理とエラーは、Javaではデバッグモードで動作させる場合にはソースコードの文字単位で異常が起こった所を逐一取り出し、“処理を停止させて”異常が起こった部分を管理者に提示できますが、『例外処理』に関してはより具体的な情報・背景を表示させたり、また、異常が起こっても致命的な『エラー』以外の異常の場合はそのまま動作を継続させることが出来るように、自分で『例外処理』のプロセスを定義できるようになっています。

今回はこの例外処理の一般的な記述方法について勉強していきましょう。

1. 代表的な『例外処理』や『エラー』
Typica l "Exception" and "Error"

“２ちゃんねる”等の掲示板で良く見かける『ぬるぽ』『ｶﾞｯ』というキーワード、これの発祥はJavaにおける『Null Pointer Exception』とそのバグを『叩く』と言う所から来ています。
Javaではおおよそほとんどのプログラマが遭遇するだろう幾つかの代表的な例外処理があります。

ClassNotFoundException
　使用するクラスが無い場合、またはクラスの依存関係が解消されていない場合に発生。
　Javaで動作させようとするクラスが無い場合に良く出現（ファイル名とクラス名が違う場合に頻発）
IOException
　入出力対象となるデバイスやファイルが無い場合や、その入出力を行おうとしたオブジェクトが無い場合に発生。
FileNotFoundException
　IOExceptionのサブクラス。対象となるファイルが見当たらない場合に発生。
NoSuchMethodException
　クラス内に使用するメソッドが無い場合に発生。

Javaアプリケーションとして動作させようとしたクラス内にMainメソッドが無い場合に良く出現。

RuntimeException
実行時に出現する種々の例外のための上位クラス。

ArithmeticException
ゼロ除算等、算術上の例外が発生した時に発生。

BufferOverflowException
リスト等を使用した際に確保されるバッファを超過した場合に発生。

IllegalArgumentException
メソッド使用時の引数内に使用不可能な（異常な）引数の型が検知された場合に発生。

IndexOutOfBoundsException
初期化時に確保した配列長に対して、インデックスがそれを超過して配列にアクセスしようとした場合に発生。

NullPointerException
オブジェクト等が初期化されていない等、配列型やオブジェクト型等のインスタンスに実体が無いのに使用しようとした場合に発生。

NegativeArraySizeException
負のサイズの配列が宣言された場合に発生。

また、同様に、プログラマが遭遇しやすいエラーもあります。

InternalError
Java Virtual Machine内部の異常による強制終了が起こる場合に発生。

OutOfMemoryError
Java Virtual Machineで確保しているHeap領域を超過してメモリ使用を行った場合に発生。

StackOverflowError
メソッド呼び出しが多重になり、メソッドのI/Oを管理するStack領域を超過してメソッドを呼び出した場合に発生。

エラーに関しては、遭遇したら強制終了させられる（程重篤な状態）ので、停止した場合のログを取得して、エラーの原因を取り除く必要があります。
例外処理については、遭遇しても強制終了する必要のないものが多く、ログを取得して処理を継続する方法を取るケースが多いです。次の節ではその方法をサンプルを通して見てみましょう。

2. Try-Catch-Finally構文による例外処理
Exception Process using Try-Catch-Finally Clause

次のソースコードは、外部からファイルを呼び出し、そのファイルからスペース区切りになっている2つの小数を使って、その分数を計算として出すものです。

```
import java.io.*;
public class ExceptionSCC {
        public static void main(String[] args) {
                BufferedReader br = new BufferedReader(new FileReader(args[0]));
                String line;
                while((line = br.readLine()) != null){
                        String[] parts = line.split(" ");
                        int i = Integer.valueOf(parts[0]) / Integer.valueOf(parts[1]);
                        System.out.println(i);
                }
        }
}
```

残念ながら、このソースコードはコンパイルエラーを引き起こします。これは、

- FileReaderによってファイルを読み込む場合には必ず "FileNotFoundException"を例外としてThrowしなければいけない

- BufferedReaderクラス内のメソッド readLine() を使用する際には必ず "IOException" を例外としてThrowしなければいけない

というルールが存在するからです。

また、もし読み込めたとしても、

- String[] parts = line.split(" "); によって分割されたStringは、もしかするとスペースが無く、要素が1個しかない（parts[1]のアクセスが出来ない）可能性がある。
- Double.valueOf(parts[0]), Double.valueOf(parts[1]) に小数とみられる数値が含まれていない可能性がある　＝　非数（NaN）の計算になる。
- Double.valueOf(parts[1]) が0である可能性がある = ゼロ除算になる。

となる可能性があり、このソースコードは仮にコンパイルできたとしても、非常に危ない、途中で停止する可能性のあるコードと言えます。

これらをまとめてその中身を例外処理し、条件に応じてその処理を分けることが出来るようにすると、次のようなソースコードとなります。

```
import java.io.*;
public class ExceptionSCC {
        public static void main(String[] args) {
                int expnum = 0;
                try{
                        BufferedReader br = new BufferedReader(new FileReader(args[0]));
                        String line;
                        while((line = br.readLine()) != null){
                                try{
                                        String[] parts = line.split(" ");
                                        int i = Integer.valueOf(parts[0]) / Integer.valueOf(parts[1]);
                                        System.out.println(i);
                                }
                                catch(ArrayIndexOutOfBoundsException aibe){ // 配列インデックス超過
                                        System.err.println("Array Index Out of Bounds Exception :" + aibe);
                                        expnum++;
                                }
                                catch(IllegalArgumentException iae){ // 引数に合わない型の使用
                                        System.err.println("Illegal Argument Exception :" + iae);
                                        expnum++;
                                }
                                catch(ArithmeticException ae){ // ゼロ除算等
                                        System.err.println("Arithmetic Exception :" + ae);
                                        expnum++;
                                }
                                catch(Exception e){ // その他もろもろ
                                        System.err.println("Others :" + e);
                                        expnum++;
                                }
                        }
                }
                catch(FileNotFoundException fe){ // ファイルが無い場合
                        System.err.println("File Not Found Exception :" + fe);
                        expnum++;
                }
                catch(IOException ioe){ // 入力エラー
                        System.err.println("IO Exception :" + ioe);
                        expnum++;
                }
                catch(Exception e){ // その他
                        System.err.println("Others :" + e);
                        expnum++;
                }
                finally{ // 全ての例外処理の回数の出力
                        System.err.println( "Done. Num of Exceptions: " + expnum);
                }
        }
}
```

各catchの中に、対応するExceptionが含まれていることに注意してください。
また、このソースコード中で起こる例外処理のカウントを入れています。
試に、[demoSCC12.txt]を入力として与えると次のような結果が出力されます。

```
92
2
1523
21615
75827
2051
Illegal Argument Exception :java.lang.NumberFormatException: For input string: "ffdf"
0
249
1
Arithmetic Exception :java.lang.ArithmeticException: / by zero
Array Index Out of Bounds Exception :java.lang.ArrayIndexOutOfBoundsException: 1
Illegal Argument Exception :java.lang.NumberFormatException: For input string: "123125.345"
28081
986596
Done. Num of Exceptions: 4
```