オブジェクト指向プログラミング　第 8回
例外処理と標準入力

担当：高橋、佐藤聖也

1　例外処理

1.1　try-catch 文

プログラムを実行すると，**例外 (exception)** が発生することがあります．例えば，次のプログラムを実行してみましょう．

DivSample01.java

```
public class DivSample01 {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;
        int c = a/b;
        System.out.println("c = "+c);
    }
}
```

この **DivSample01.java** は，ゼロによる割り算（ゼロ除算）を行っているため，次のようなエラーメッセージが出力されます．

```
Exception in thread "main" java.lang.ArithmeticException:  / by zero
at Lec07.DivSample01.main(DivSample01.java:6)
```

`ArithmeticException` とは「算術に関する例外」という意味ですが，ゼロ除算は値が定義されないので，このようなメッセージが表示されます[1]．

Table 1.1　エラー処理のためのクラス（一部）

例外クラス	内容
ArithmeticException	算術計算における例外処理
ArrayIndexOutOfBoundsException	配列の添字が範囲外になっている場合の処理
StringIndexOutOfBoundsException	文字列中に存在しない位置の文字を取りだそうとした場合の処理
NegativeArraySizeException	負の大きさを持つ入れるを生成しようとした時の処理
NullPointerException	プログラムが null を使おうとしたときの処理
AccessControlException	アクセスが拒否された場合の処理
IOException	ファイル入出力に関する例外処理．サブクラスを多数持つ
EOFException	入力途中で EOF となった場合の処理
FileNotFoundException	ファイルが見つからない場合の処理
UnknownHostException	取得しようとした IP アドレスがない場合の処理
ClassNotFoundException	実行しようとしたクラスがないときの処理

ゼロ除算のような算術上の例外が発生すると，ArithmeticException というクラスのインスタンスが生成され，その時点でプログラムが強制的に終了し，JVM のエラーメッセージ表示プログラム

[1] Java の場合，このようなエラーメッセージが返されるのは，整数型のゼロ除算であり，実数型のゼロ除算では Infinity が値として返ってきます（確かめてみて下さい）．

にエラー情報を格納したインスタンスを渡します[2]．エラーの種類には様々なものがあり，Table 1.1 のようにそれぞれに応じたクラスが用意されています．

さて，**DivSample01.java** のゼロ除算で発生するエラーを回避するために，差し当たり次のような if 文を使ったプログラムを書くことができるでしょう．

DivSample02.java

```
public class DivSample02 {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;
        if(b != 0){
            int c = a/b;
            System.out.println("c = "+c);
        }else{
            System.out.println("We cannot calculate a/b!");
        }
    }
}
```

これを実行すると「`We cannot calculate a/b!`」というメッセージが表示されます．しかし，Java プログラミングでは **DivSample02.java** のような if 文を使った処理はせず，次の **ExceptionSample01.java** にあるような，**try-catch 文**とよばれるものを使います．

ExceptionSample01.java

```
public class ExceptionSample01 {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;
        try{
            int c = a/b;
            System.out.println("c = "+c);
        }catch(ArithmeticException e){
            System.out.println("ArithmeticException generated!");
        }
        System.out.println("It was finally done.");
    }
}
```

ここでは，try ブロック内で発生しスローされた例外オブジェクト（この場合，ArithmeticException クラスのインスタンス）が catch 文でキャッチされ，catch ブロック内で書かれた処理が行われます[3]．

[2] これを「**例外をスローする (throw, 投げる)**」といいます．また，渡されるインスタンスを**例外オブジェクト (exception object)** といいます．

[3] **DivSample01.java** では，発生した例外オブジェクトはプログラム自身の中には行き場がなく，JVM のエラーメッセージ表示プログラムにスローされ，プログラムは強制終了されていましたが，**ExceptionSample01.java** では，発生した例外オブジェクトが catch 文でキャッチされ処理されるため，プログラムは強制終了されません．

1.2 try-catch-finally 文

try ブロックおよび catch ブロックの後ろに，さらに finally ブロックとよばれる部分を付け加えることができます．try-catch 文の中でどのような処理になろうと，finally ブロックは必ず実行されます．

ExceptionSample02.java

```
public class ExceptionSample02 {
    public static void main(String[] args) {
        try{
            int a = 10;
            int b = 1;
            int c = a/b;
            System.out.println("c = "+c);

            //ArithmeticException 以外の例外発生
            String str = "abc";
            char ch = str.charAt(10);
            System.out.println("ch = "+ch);

        }catch(ArithmeticException e){
            System.out.println("ArithmeticException generated!");
        }finally{
            //どんな例外が出ても，ここは実行される
            System.out.println("It was done(1)");
        }
        //想定外の例外が出るとここは実行されない
        System.out.println("It was done(2)");
    }
}
```

ExceptionSample02.java を実行すると，以下のような結果が表示されます．

```
c = 10
It was done(1)
Exception in thread "main" java.lang.StringIndexOutOfBoundsException:  String
index out of range:  10
at java.lang.String.charAt(String.java:686)
at Lec07.ExceptionSample02.main(ExceptionSample02.java:12)
```

try ブロックで発生したエラーが catch 文でキャッチできなかった場合，finally ブロックが実行された後，プログラムは強制終了します．

1.3　複数の例外のキャッチ

ExceptionSample03.java

```
public class ExceptionSample03 {
    public static void main(String[] args) {
        try{
            //ArithmeticException を発生させる
            int a = 10;
            int b = 0;
            //int c = a/b; //(A1)
            //System.out.println("c = "+c); //(A1)

            //IndexOutOfBoundsException を発生させる
            String str1 = "abc";
            //char ch = str1.charAt(10); //(B1)
            //System.out.println("ch = "+ch); //(B1)

            //NullPointerException を発生させる
            String str2 = null;
            //int k = str1.compareTo(str2); //(C1)
            //System.out.println("k = "+k); //(C1)

        }catch(ArithmeticException e){
            //(A2)
            System.out.println("ArithmeticException generated!");
        }catch(IndexOutOfBoundsException e){
            //(B2)
            System.out.println("IndexOutOfBoundsException generated!");
        }catch(NullPointerException e){
            //(C2)
            System.out.println("NullPointerException generated!");
        }finally{
            System.out.println("It was done(1)");
        }
        //想定外の例外が出るとここは実行されない
        System.out.println("It was done(2)");
    }
}
```

ExceptionSample03.java では，(A1)，(B1)，(C1) のいずれかの行のコメントアウト//を外して実行すれば，それぞれ (A2)，(B2)，(C2) の catch ブロックが実行されます．

charAt メソッドは，例えば str1.charAt(10) としたとき，str1 で参照される文字列の先頭の文字を 0 番目として 10 番目の位置にある（つまり 11 個目の）一文字を取り出してきて char 型として返すメソッドです．**ExceptionSample03.java** では，str1 は 3 文字の文字列を指しているので，str1.charAt(10) とした場合，IndexOutOfBoundsException クラスの例外オブジェクトがスローされます．

compareTo メソッドは，例えば str1.compareTo(str2) としたとき，str1 と str2 が参照する文字列を比較して，同じであれば 0，異なっていれば-1 を，int 型で返すメソッドです．**ExceptionSample03.java** では，str2 がであるため，NullPointerException クラスの例外オブジェクトがスローされます．

2　標準入力

2.1　InputStreamReader クラス

Java プログラムにおいて，標準出力は System.out.println(); を用いて行われましたが，標準入力はどのように行えばよいでしょうか．最初に，**InputSample.java** を例として考えてみます．

InputSample.java

```
import java.io.*;//InputStreamReader クラスが含まれる

//このクラスはコンパイルエラーが発生します
public class InputSample {
    public static void main(String[] args){
        //バイトストリームを文字ストリームに変換するため
        InputStreamReader isr = new InputStreamReader(System.in);
        System.out.print("Prease input a charactor: ");

        //標準入力からバイトデータを読み込み単一の文字を返す
        int c = isr.read();
        System.out.println("Your input charactor is : "+(char)c);
    }
}
```

最初に，InputStreamReader クラスを使用するため，これが含まれる java.io パッケージをインポートします．

標準出力ストリームは，System クラスのクラスフィールドである out を使って System.out と書くことができ，標準出力は System.out.println(); のように行いました．同様に標準入力ストリームは，System クラスのクラスフィールドである in を使って System.in と書くことができます．

InputStreamReader クラスは，標準入力からのバイトストリーム（byte 型データの列）を文字ストリーム（文字列）に変換するためのメソッドが用意されたクラスです．また，InputStreamReader クラスのインスタンスメソッドである read メソッドは，標準入力からバイトデータを読み込み単一の文字の Unicode を返します．

さて，このファイルをコマンドプロンプトからコンパイルしようとすると，次のようなコンパイルエラーが起きます．

```
C:¥CompExB¥07>javac InputSample.java
InputSample.java:10:  例外 java.io.IOException は報告されません。
スローするにはキャッチまたは、スロー宣言をしなければなりません。
    int c = isr.read();
                  ^
エラー 1 個
```

Ecripse では，コンパイルと実行がワンクリックで実行されますが，次のようなエラーが発生します．

```
Exception in thread "main" java.lang.Error:  Unresolved compilation problem:
    処理されない例外の型 IOException
at Lec07.InputSample.main(InputSample.java:12)
```

キーボードからデータを入力する場合，IOException という例外が発生する可能性があるため，必ず try ブロックを使う必要があるので，このようなコンパイルエラーが発生します．

練習問題 1

InputSample.java に try-catch 文を加筆しコンパイルエラーの起こらないプログラム **InputSample01_00rd000.java** を作成せよ．ただし，**00rd000** は各自の学籍番号とする．

2.2 BufferedReader クラスを用いた入力の最適化

InputSample.java では

```
InputStreamReader isr = new InputStreamReader(System.in);
```

によって，標準入力からのバイトストリーム（byte 型データの列）を文字ストリーム（文字列）に変換することを学びました．しかし，**この文だけでは変換の効率が悪く，通常は用いません．**どうするかというと，

```
BufferedReader br = new BufferedReader(isr);
```

のように，BufferedReader クラスを用いて文字，配列，行をバッファリングすることにより，文字型入力ストリーム isr からテキストを効率良く読み込むようにします．

上記の 2 行は

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

のように 1 行にまとめることができるため，通常，この文を使って **InputSample02.java** のように標準入力を構成します．

InputSample02.java

```
import java.io.*;//InputStreamReader クラスを含む
public class InputSample02 {
    public static void main(String[] args) {
        try{
            //InputStreamReader isr = new InputStreamReader(System.in);
            //BufferedReader br = new BufferedReader(isr);
            //上の 2 行は下のようにまとめられる
            BufferedReader br
            = new BufferedReader(new InputStreamReader(System.in));

            System.out.print("Please input some strings: ");
            String inputData = br.readLine();
            System.out.println("Your input strings are   : "+inputData);
        }catch(IOException e){
            System.out.println("IOException!!");
        }
    }
}
```

この中の

```
String inputdata = br.readLine();
```

では，バッファリングされたストリーム（の 1 行分）を文字列として inputData で参照できるようにしています.

2.3 Scanner クラス

InputStreamReader クラスと BufferedReader クラスを使った入力よりも豊富なメソッドを使えるクラスとして，Java SE 5 以降で導入された Scanner クラスがあります．Scanner クラスを使い，一行の標準入力を読み込んで同じ文字列を標準出力するプログラムは，次のように書くことができます.

ScannerSample01.java

```
import java.util.*;//Scaner クラスを含む
public class ScannerSample01 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Prease input some strings: ");
        String inputData = sc.nextLine();
        System.out.println("Your input strings are   : "+inputData);
    }
}
```

課題

1. 練習問題 1 で作成した **InputSample01_00rd000.java** を提出しなさい.
2. ある文字列が標準入力されたとき，次のように出力するプログラム **Kadai02_00rd000.java**を作成せよ．ただし，**00rd000** は各自の学籍番号とする．String クラスの，charAt メソッド，length メソッドなどを使うとよい.
   - 'a' ならば'x'，それ以外の文字ならば'o' に置き換える.
   - 空白は空白のままとする.
   - "ab Adaf" という文字列であれば，"xo ooxo"を出力する.

提出の際の注意点：
- コンパイルすると実行できる形式で提出すること
- 学籍番号のフォルダに提出ファイルを全て入れ、そのフォルダをzipファイルにして提出すること。

提出先：https://tdu.app.box.com/f/3d13ce1298654463a236ec9ec4436bbe
提出期限：7/5 23:59