発展プログラミング演習II

8.2 例外

オブジェクト指向の基礎

- クラス
- インタフェース
- メソッド
- フィールド
- 継承
- アクセス制御
- 多態性
- 例外

教科書8章参照

例外 exception

- Java言語でのエラー処理の方法の一つ

エラー処理は重要

- プログラムが異常動作、異常終了すると……
  - 使い物にならない、ユーザの不満
  - システム全体のセキュリティの問題

- 滅多に呼ばれないのに必要
- エラー処理のコーディング量は全体の半分以上という説も

エラーを起こしうる例

```
int num[20];

void getNum(int index) {
    return num[index];
}
```

- 配列の添字の範囲を超える可能性がある

例外を使わないエラー処理(1/2)

- 不具合が起きないように事前にチェックする
- 例：
  - 配列のインデックスは有効範囲内か？
  - 0で割ろうとしてないか？

```
int intArray[20];

if (index < 0 || index >= 20) {
  printf(“index out of bounds\n”);
}
.....
```

例外を使わないエラー処理(2/2)

- ある関数内で不都合が生じたことを、返値で関数の呼び出し元に知らせる
  - 例
    - 成功したらtrue, 失敗したらfalseを返す
    - 特別な値を返す　例：null, -1
- グローバル変数に処理結果をセットする
  - 例：succeed = false;

例外を使わないエラー処理の問題

- エラーチェック漏れ
- 返値の意味をどうやって知るか

- エラーが起こるとプログラムが変な動作をしたり、異常終了してしまいかねない

例外を使ったエラー処理

- 異常終了を未然に防ぐ
- プログラム中のエラー処理部分を明確にできる
- エラー処理を呼び出し元に強要できる

例外処理の基本構造

- エラーが発生した場合、例外を投げる
  - エラーの起きたメソッドは、呼び出し元に対して例外クラスオブジェクトを渡す
- 例外を投げられたら受け止める
  - 例外を投げる可能性があるメソッドを呼ぶ際には、投げられた場合の処理も書いておく

try文

```
try {
  ..... // 通常の処理
} catch (例外クラス名 引数名) {
  ..... // 例外発生時の処理
}
```

tryの中に例外があれば、catchに行く

- tryブロックで例外を投げうるメソッドを呼ぶ
- 例外が起こったらcatchブロックに飛ぶ
  - catchブロックは例外クラスごとに複数書ける

finallyブロック

```
try {
  ..... // 通常の処理
} catch (例外クラス名 引数名) {
  ..... // 例外発生時の処理
} finally {
  ..... // いずれにしても最後に実行する処理
}
```

- 例外が投げられても必ず実行されるブロック
- 後片付けの処理を書くことが多い
  - ファイル処理の際のcloseなど

try文について補足

- 例外が発生した場合、例外を投げたメソッド以降のtryブロック中の処理は行われない
- 空のcatch節を書くのは避けよう
- 複数の例外クラスの処理を分ける場合はcatch節を複数書く。例外クラスの型は先頭から順にチェックされる。

例題8.5

- Example0802.javaを実行して
  例外処理を確認せよ
  - 整数以外を入力してみる

どのメソッドが例外を投げる？

- メソッドの宣言部で throws が指定してある
  - 例：
    public static int Integer.parseInt(String s)
    throws NumberFormatException
- throwsが指定してあるメソッドを呼ぶ時は、次のいずれかをしておく必要がある
  - try-catchを書く
  - 受けた例外を呼び出し元に丸投げする

代表的な例外のクラス(1/3)

- java.lang.Throwable
  - すべてのエラーと例外のスーパークラス
- java.lang.Error
  - キャッチすべきでない重大な問題
  - throws節を宣言する必要がない
- java.lang.Exception
  - 通常のアプリケーションでキャッチされうる例外の上位クラス

代表的な例外のクラス(2/3)

- java.lang.RuntimeException
  - java.lang.Exceptionのサブクラスのひとつ
  - コンパイル時にチェックされない例外
  - 例外処理を書かなくてもよい
  - java.lang.NullPointerException
    java.lang.ClassCastException
    java.lang.ArrayIndexOutOfBoundsException
    などがサブクラス

代表的な例外のクラス(3/3)

- java.lang.ArithmeticException
  - 算術計算での例外、例えば0割
- java.lang.SecurityException
  - セキュリティ違反
- java.io.IOException
  - 入出力での例外のスーパークラス
  - FileNotFoundExceptionなど

throws節

`throws 例外クラス名`

- 例外を投げうるメソッドで宣言する必要がある
- 投げうる例外クラスが複数ある場合は,でつないで書く
- 例外を投げるメソッドを呼び出すメソッドは、次のどちらかの対応をする必要がある
  - 同じ例外を投げる（throws節を付ける）
  - 例外を処理する（try-catchを書く）

throw文

```
throw 式;
```

- 式にはThrowableクラスオブジェクト（Throwableのサブクラスでもよい）
- 式の省略は不可
- メソッドはthrow文で終了して呼び出し元に戻る（例外をthrowする＝異常が発生したことを知らせる）

例題8.6

- Example0803.javaを実行して
  例外処理を確認せよ
  - 整数以外を入力してみる

例題8.7

- Exercise87.javaのgetFruitNameメソッドのエラー処理を次のように書き直せ
  - ArrayIndexOutOfBoundsExceptionが発生したら呼び出し元に投げる
    - プログラムの冒頭に
      import java.lang.ArrayIndexOutOfBoundException;
      と追記すること
  - mainメソッドは例外を受けたらエラーメッセージを表示する

例題8.8

- Exercise88.javaのgetFruitNameメソッドに、次の例外処理を追加せよ（NotAFruitExceptionクラスは別途提供する）
  - idが3の時、NotAFruitExceptionを投げる
  - mainメソッドは例外を受けたら
    エラーメッセージを表示する

課題8.2

- 課題8.1のプログラムを例外を使って修正せよ
  - 玉が無くなったことを表す例外クラスPachiEmptyExceptionをExceptionクラスを継承して作成する（中身は何も無くてよい）
  - Pachiクラスのplayメソッドは玉が無くなったら例外PachiEmptyExceptionを投げる
  - PlayPachiクラスのsimulateメソッドは、玉が無くなったらメッセージを表示して、繰り返しを終了する。