Programming II

第13回　例外処理・ファイル処理（第17章）

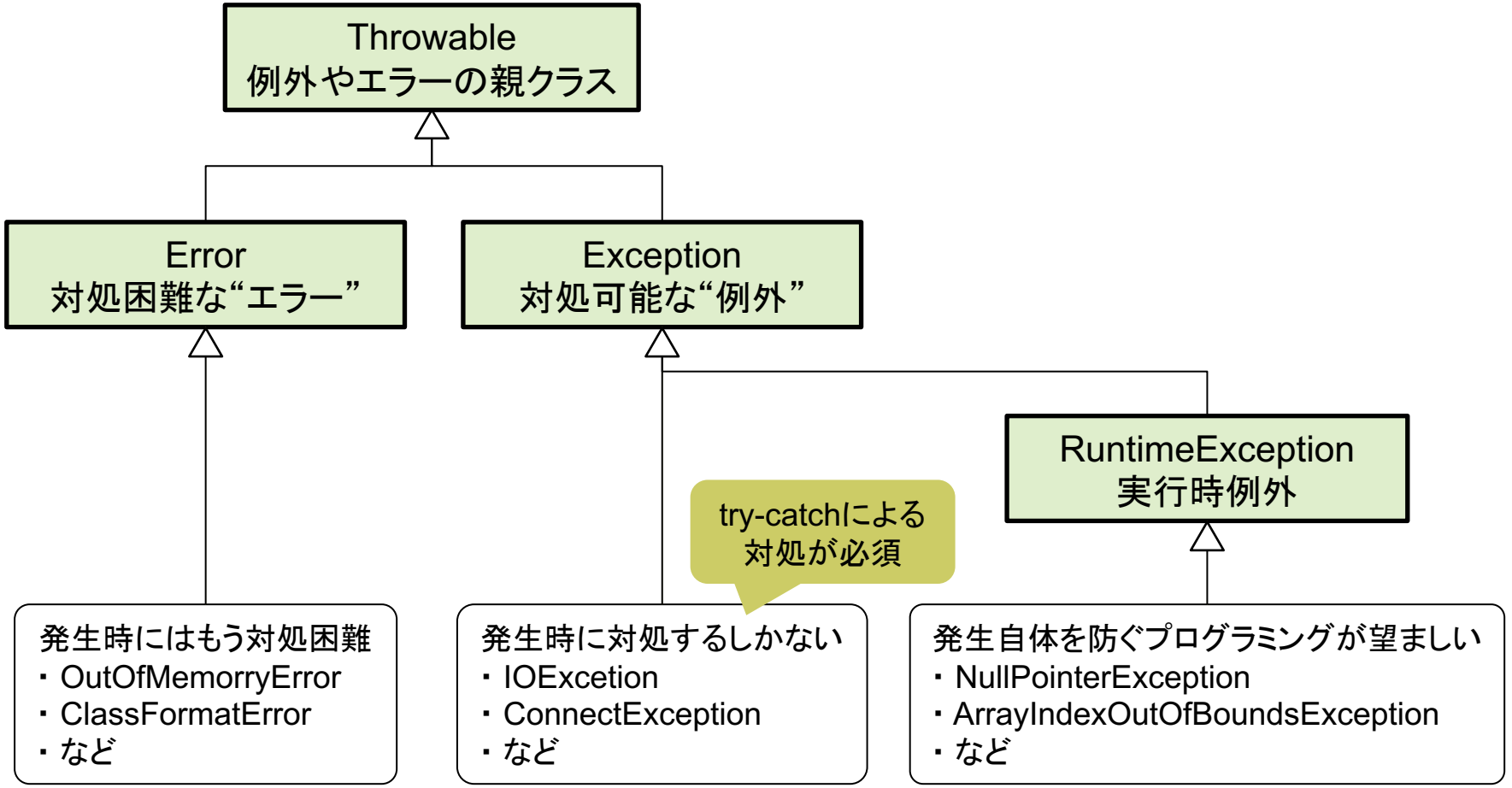
塩澤秀和

13.1 例外処理

- プログラミングにおける“例外”（p.651）
  - プログラムの実行中に発生する特別に対処すべき状況
  - 例）数値のエラー，想定外の入力，ハードウェアの異常
- どうやって例外に対処するか？（p.653）
  - 事前にあらゆる事態を想定しておき，if文等で予防する？
    ⇒ プログラムの分岐が増え，見通しが悪くなりやすい
  - 異常表す特別な値をreturnし，受け取った側が対処する？
    ⇒ 戻り値だけでは，対処方法が複雑になることがある
- Javaなどの言語の解決策（p.655）
  - “例外”を定義し，その発生と処理のための機能を提供

13.2 例外関連クラス

- 例外の種類はクラスで分類されている（p.658）



13.3 try-catch文

- 例外処理の構文（p.656,666）
  - 発生した（投げられた）例外ごとに，キャッチして対処する
  - キャッチされない例外が発生した場合は，異常終了する

```
try {
    通常の処理（例外が発生するかもしれない処理）
    ...
} catch (例外クラス名１　変数1) {
    例外1が発生した場合の処理
    ...
} catch (例外クラス名２　変数2) {
    例外2が発生した場合の処理
    ...
}
...
```

13.4 try-catch文（続き）

- catch (Exception e)
  - とりあえず，すべての例外をキャッチする書き方（p.667）
- finally { 処理 }
  - catchブロックの後の最後に書ける
  - 例外が発生しても，発生しなくても，キャッチされなくても，後始末として実行する処理を記述する（p.670）
- try (処理) {
  - カッコ付きtry文：「処理」の部分には複数の文が書ける
  - カッコ内でnewされたインスタンスは，tryブロックを抜けるときに自動的にclose()が呼ばれて後始末される（p.679）

13.5 例外の発生と伝播

- 例外を発生させる（投げる）（p.685，使用例：p.510）
  - 例外クラスのインスタンスを生成し，throwする

```
if (arg1 == null) {
    throw new IlligalArgumentException("不正な引数！");
}
```

  - 自作の例外クラスを定義して，throwすることもできる

- メソッドからの例外の送出と伝播（p.682）
  - メソッド内で発生した例外を，その場ではキャッチせずに呼び出し元に処理を委譲し，伝播させていくことができる
  - そのように，内部で発生した例外を送出するメソッドは，名前の後にthrowsをつけて，例外クラスを列挙する

```
void method1() throws IOException { … }
```

13.6 ファイル処理

- ファイル
  - 名前がつけられて（補助）記憶装置に保存（セーブ）されたデータ
  - ソフトの途中データの保存，他のソフトとのデータ交換

- ファイルの種類
  - テキスト：単純な文字データだけが記録されたファイル
  - バイナリ：画像，音楽，ワープロの文書ファイル等

- パス（path）
  - ファイルやフォルダの場所を表す記法
  - 絶対パス：C:¥フォルダ名¥フォルダ名¥・・・¥ファイル名
  - 相対パス：フォルダ名¥フォルダ名¥・・・¥ファイル名

プログラムの中では「¥」の代わりに「¥¥」

13.7 ファイルに書き出す

- java.io.FileWriterクラス（p.696）
  - ファイルにデータを書き出すためのクラス
  - 入出力の失敗では，例外 IOException が送出される
- 準備（オープン）
  - fw = new FileWriter("ファイルのパス名");
- 書き出し
  - fw.write(文字列);
  - fw.write(x + " "); ←区切りの空白を付加する方法
- 後始末
  - fw.close(); ←カッコ付きtry文を使っていれば通常不要

13.8 FileWriterの使用例

```
// ファイル冒頭に適切なimport文（例：java.io.*）が必要
final String KAI = System.lineSeparator();  // 改行文字を取得しておく
double data = 10.0;

// カッコ付きtry文を使用しないと，後始末に複雑なfinally処理が必要（p.676）
try (FileWriter fw = new FileWriter("test.txt")) {

    fw.write("テストです。" + KAI);  // 文字列と改行を書き出す
    fw.write("" + data + KAI);      // 数値は文字列に変換

    // fwはカッコ付きtry文でnewされたので，close()は自動的に呼び出される

} catch (IOException e) {
    System.out.println("入出力エラーが発生しました。");
}
```

13.9 実は，改行はややこしい…

- 改行を表す文字コード
  - 実は，標準の改行コードは環境（Windows, Mac等）によって違う
  - Windows　"¥r¥n"（CR+LF）← たいていは，"¥n"だけで問題ない
  - UNIX, Mac　"¥n"　（LF）　　← Macの古い形式では，"¥r"のみ
  - ただし，多くのソフトでは標準以外の改行コードのファイルも扱える
- print系での改行出力
  - System.out.println(文字列);　// 文字列の後に改行が出力される
  - System.out.println();　　// 改行のみが出力される
  - System.out.printf("%n");　　// printfでは%nで改行が出力できる
- 実行環境に合わせた改行の利用方法
  - String kai = System.lineSeparator();　// 改行文字を取得する方法
  - System.out.print("文字列" + kai);　　// printlnと同じ動作になる
  - fw.write(data + kai);　　// ファイル出力での改行付加

13.10 ファイルから読み込む

- java.util.Scannerクラス
  - キーボードからの入力と同じ方法で，java.io.Fileクラスで開いたファイルからデータを読み込める
- 準備
  - sc = new Scanner(new File("ファイルのパス名"));
- 読み込み
  - if (sc.hasNext()) { ... ←次のデータがあるかチェック
  - a = sc.nextInt(); など ←データの読み込み処理
- 後始末
  - sc.close(); ←カッコ付きtry文を使っていれば通常不要

13.11 Scannerの使用例

```
// ファイル冒頭に適切なimport文（例：java.util.*とjava.io.*）が必要

// カッコ付きtry文を使用しないと，後始末に複雑なfinally処理が必要（p.676）
try (Scanner sc = new Scanner(new File("test.txt"))) {

    String str = sc.nextLine();     // 文字列の読み込みと表示
    System.out.println(str);

    double x = sc.nextDouble();  // 数値の読み込みと表示
    System.out.println(x);

    // scはカッコ付きtry文でnewされたので，close()は自動的に呼び出される

} catch (Exception e) {
    System.out.println("エラーが発生しました。");
}
```

13.12 演習

- 13.8と13.11
  - mainメソッドに，13.8のコードを記述して動作を確認せよ
  - 生成されたテキストファイルtext.txtの内容を確認せよ
  - さらに，続きに13.11をコードを記述して動作を確認せよ
  - そのプログラムを，自分のクラス（int），番号（int），名前（String）をファイルに保存し，さらにそれらをファイルから読み出して表示するように修正せよ

13.13 配列にデータを読み込む

```
double[] data = new double[10];
int num = 0;

try (Scanner sc = new Scanner(new File("data.txt"))) {

    int i;
    for (i = 0; i < data.length; i++) {
        // 次のデータがあるかどうかチェックし，ない場合はループを離脱する
        if (!sc.hasNext()) break;
        data[i] = sc.nextDouble();
    }
    num = i;

} catch (Exception e) {
    System.out.println("エラーが発生しました。");
}
```

第13回 まとめ

- 例外処理
  - 例外関連クラス
- try-catch文
- 例外の発生と伝播
  - throw, throws
- ファイル処理
- ファイルに書き出す
  - FileWriterクラス
- ファイルから読み込む
  - Fileクラス
  - Scannerクラス