第8章
例外処理

例外

実行時に発生するエラーのこと

例外が発生する　　⇒　例外が**スロー**される

例外処理

例外への対処をしていない　　⇒　プログラムは**強制終了**される

例外処理により対処可能

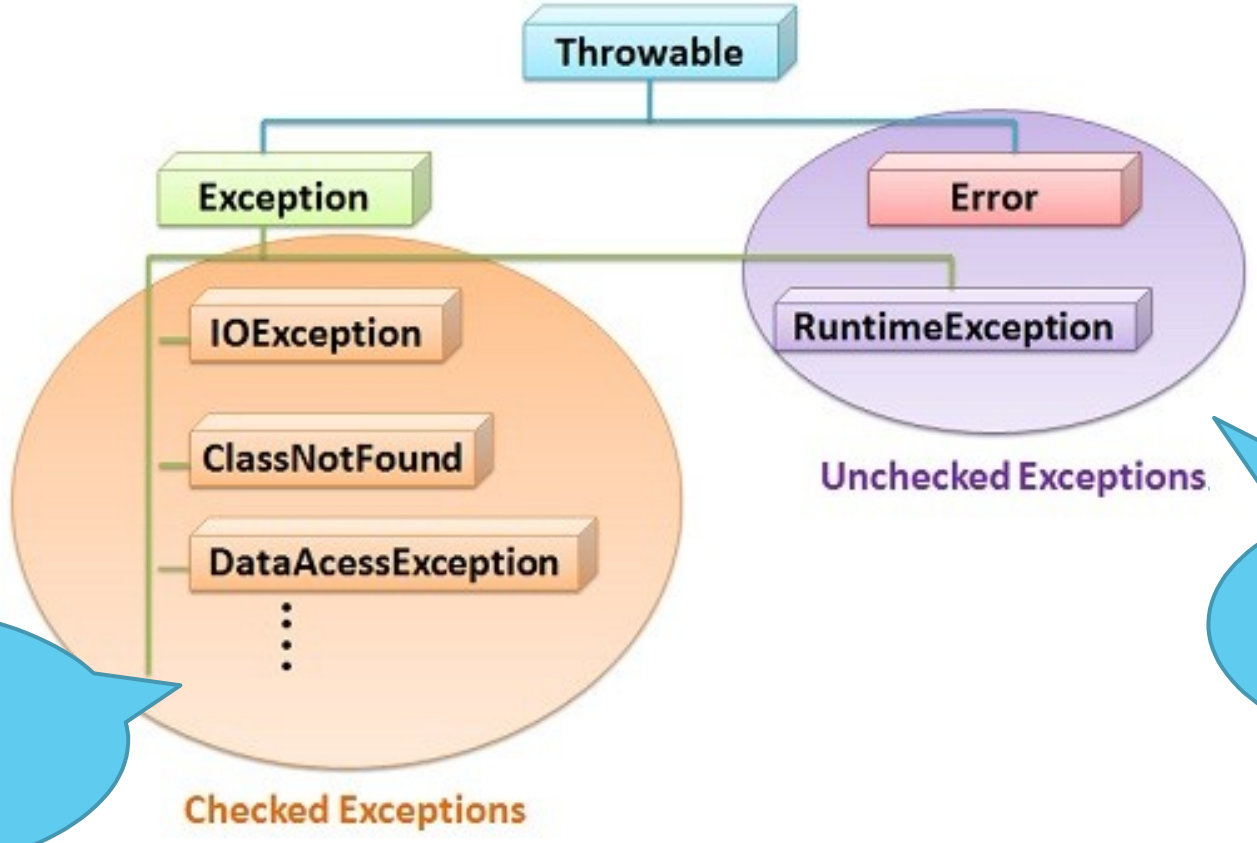
例外クラス

checked例外　Java実行環境以外の環境が原因の例外

unchecked例外　実行中のプログラムが原因の例外

プログラムの例外処理では復旧できない例外

例外クラス



例外処理は
必須

例外処理は
任意

主な例外クラス

ArrayIndexOutOfBoundsException

配列のインデックスとして存在しない要素を指定した場合に発生する

対策

正しい配列のインデックスを指定できているかチェックする

配列の設定を繰り返し処理で行っている場合、繰り返し回数などに
問題がないかチェックする

主な例外クラス

ClassCastException

あるクラスを変換できないクラスにキャストしようとした時に発生する

対策

配列などに設定した値のクラスと、値を取り出す時にキャストしようとしている型が一致しているかをチェックする

継承関係のある2つのクラスで、サブクラスにスーパークラスをキャストしようとしていないかをチェックする

主な例外クラス

ArithmeticException

「ゼロ除算」など、不正な算術処理が行われた場合に発生する

対策

ゼロ除算が行われてないかをチェックする

数値型変数で計算している場合、割る数の変数にゼロが設定されていないかチェックする

主な例外クラス

NullPointerException　　通称：ぬるぽ

nullのオブジェクトに対してアクセスしようとしたときに発生する

対策

インスタンスの生成や値の設定が正しく行われているかをチェックする

オブジェクトがnullになってしまうタイミングをデバッグで確認する

⇒その他の例外についてはテキストのP330 表8-1を参照するか、ググってください

独自例外クラス

提供されている例外クラス以外に、
プログラマが独自で例外クラスを定義することが可能

既存の例外クラスを継承する
※一般的にはExceptionクラスを継承

try-catch-finally

```
try {
    例外が発生する可能性のある処理
} catch (例外の型 引数) {
    例外が発生した場合の処理(例外が発生しなければ行われない処理)
} finally {
    例外の有無に関わらず、最後に必ず実行される処理
}
```

catchブロックは複数定義することも可能

throwsとthrow

ざっくり言えば、記述する位置が違う

throws　メソッド内で発生したExceptionを、メソッド内でcatchするのではなく、呼び出し元に投げる
⇒記述できるのはメソッドの引数の横

```
public void test() throws IOException, SQLException {

}
```

throw　メソッド内で意図的にExceptionを発行し投げる
⇒記述できるのはメソッドの内側

```
public static void main(String[] args) {
  throw new RunTimeException("意図的にExceptionを発行しました");
}
```

オーバーライドの注意点

throwsキーワードが使用されているメソッドを
オーバーライドする場合にはルールがある

文字ばかりになりそうだったので詳細はテキストP344参照…