Ｊａｖａプログラミング Ⅱ

９回目　例外処理

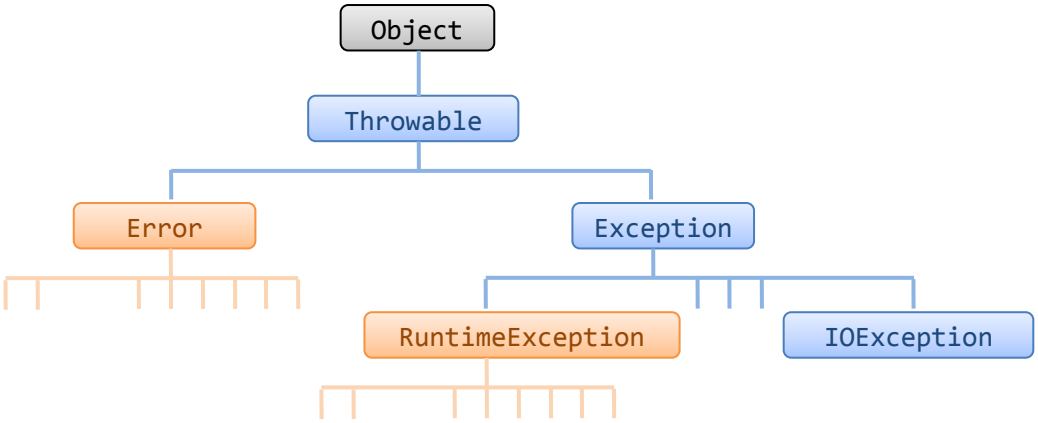
§例　　外

■例外とは　　プログラムの**実行中に発生するエラー**です

■例外クラス　　**例外を表現するクラス**です
発生した例外はこの例外クラスのオブジェクトで表現されます

具体的には、
Throwable クラス、又は**そのサブクラスのオブジェクト**
で表現されます

□例外クラスのクラス階層



□例外クラスのクラス階層と分類

Error クラス以下　→　**続行不可能な例外** Virtual Machine のエラーなど
Exception クラス以下　→　**実行時に対処する必要のある例外** I/O エラーなど
RuntimeException クラス以下　→　**予め防げる例外** 配列要素の数を超える代入など

□例外クラスの 2 つのタイプ

クラス　**検査（Checked）例外**　　コンパイラは**例外処理の約束**を検査します

クラス　**非検査（Unchecked）例外**　コンパイラは例外処理の約束を検査しません

※例外処理の約束については 3 ページ目を見てください

§例外処理

■例外処理とは　　発生した**例外に対して行われる処理**です

■宣言（try 文）

```
try{
  文
}catch(例外ｸﾗｽ型 1  e){
  文
}catch(例外ｸﾗｽ型 2  e){
  文
  :
}finally{
  文
}
```

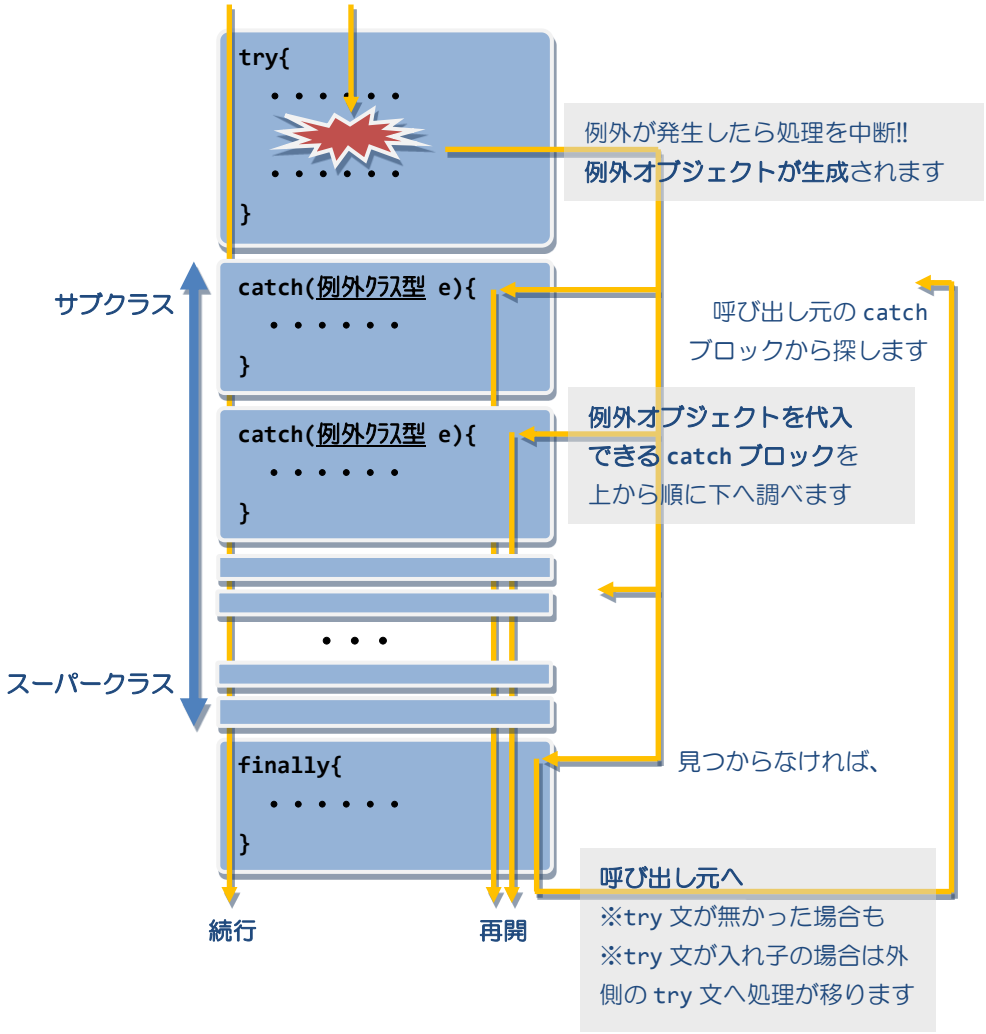
try 文の最小構成

1 つの try 文は、1 つの try ブロックと、少なくとも 1 つの catch ブロック又は finally ブロックをもちます

■機　　能

①try ブロック　　例外の発生を監視します
②catch ブロック　　例外が発生すると例外に対応する処理を実行します
③finally ブロック　try ブロックに入ったら
・例外の発生の有無に関わらず
・break や return により try ブロックを出るなど
try 文を抜ける時に必ず実行されます

■処理の流れ



§例外の送出

■例外の送出とは　**例外を発生させること**です

■手　　順

1. Exception クラスを拡張して**各自の例外クラスを宣言**します
   一般に Throwable クラスを継承していればよいです
   既存の例外クラスを用いてもよいです
2. **オブジェクトを生成**します
3. throw オブジェクトを指す変数; により**例外を発生**させます

§throws 宣言

■throws 宣言とは　メソッドが**例外を発生する可能性があることを明示**します

例外の送出を行うメソッド、例外処理を記述していない例外や
例外処理では catch していない例外を発生する可能性のあるメソッド
にこれを宣言します

■宣　　言

```
戻り値の型 メソッド名(引数リスト) throws 例外クラス1, 例外クラス2,…{
        本　体
}
```

例外クラス は例外に対応するクラス
のスーパークラスでもよいです

§例外処理の約束

〔**約束**〕あるメソッド A 内で例外を送出するメソッド B を実行する場合、
次のいずれかを行います

1. **メソッド A 内で try 文によりメソッド B の例外を処理してしまいます**
   → メソッド A の利用者は例外を気にせずにコードを実行できます
2. **メソッド A に throws 宣言をします**
   → メソッド A の使用者にこのメソッドは例外が発生することを伝えます

□**検査例外**の場合　→ コンパイラがこの**約束をチェックします**

例えば、
IOException（キーボードなどの入力に関する例外）
や Exception を拡張した独自の例外クラスなど
実行時に対処する必要のあるどうしても起きてしまう例外です

□**非検査例外**の場合　→ コンパイラはこの**約束をチェックしません**
例外への対応はプログラマに任されます

例えば、
ArrayIndexOutOfBoundsException（配列範囲超え）
ArithmeticException（ゼロでの除算）など
予めアルゴリズムの工夫で防げる例外です

§throws 宣言をもつメソッドのオーバーライドの約束

※教科書では扱っておりません ※試験範囲から外します

オーバーライドの機能を妨げないように新しいメソッドの throws 宣言を次のようにします

〔オーバーライドの約束〕

オーバーライドする側のメソッド A は、オーバーライドされる側（スーパークラス又はインタフェース）のメソッド B が送出する例外のクラスとそのサブクラスを送出できます

メソッド B を使うコードはメソッド B が送出する例外に対する例外処理のみを持つためです

□拡張の場合の例 1

```
class Super{
    void A() throws E1
}
class Sub extends Super{
    void A() throws E2
}
```

例外クラス **E2** は、
「例外クラス **E1** と同じ、又はそのサブクラス」
とします

□拡張の場合の例 2

```
class Super{
    void A( ) throws E1, E2
}
class Sub extends Super{
    void A( ) throws E3
}
```

例外クラス **E3** は、
「例外クラス **E1** または **E2** と同じ、又はそれらのサブクラス」
とします

□実装の場合の例

```
interface Inter{
    void A( ) throws E1, E2
}
class Sub implements Inter{
    void A( ) throws E3
}
```

例外クラス **E3** は、
「例外クラス **E1** または **E2** と同じ、又はそれらのサブクラス」
とします