PRG1-w2b
イベント処理 → 副作用と例外処理

脇田建

2019.10.7

例外って何？

困った状況.1

- 1 / 0

困った状況

- 1 / 0: 0での除算は定義されていない。

```
⌥⌘1                                Default
dancer.local        10/05 3:13 PM        master

scala> 1/0
java.lang.ArithmeticException: / by zero
  ... 28 elided

scala>
```

1 / 0 と類似の事例

- 計算が定義されていない/定義域を外れている
- import scala.math
  math.toIntExact(Int.MaxValue.toLong + 1)
- import math.BigDecimal
  BigDecimal(“0”).pow(1000000000)
- def divide(a: List[Int], b: List[Int])

困った状況.2

- Integer.parseInt(“壱”)

困った状況.2: Integer.parseInt(“壱”)

- “12345” → 12345 なんだから、

  ”壱”だって、ブツブツ。。。

```
scala> "12345".toInt
res2: Int = 12345

scala> "壱".toInt
java.lang.NumberFormatException: For input string: "壱"
  at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:6
5)
  at java.base/java.lang.Integer.parseInt(Integer.java:652)
  at java.base/java.lang.Integer.parseInt(Integer.java:770)
  at scala.collection.immutable.StringLike.toInt(StringLike.scala:304)
  at scala.collection.immutable.StringLike.toInt$(StringLike.scala:304)
  at scala.collection.immutable.StringOps.toInt(StringOps.scala:33)
  ... 28 elided

scala>
```

Integer.parseInt(“壱”) と類似の事例

- 想定外の入力
- 別のやりかた: “壱”.toInt
- 想定外の入力への対応
  - “3.14”.toInt
  - “3+2i”.toDouble
- シメイ ← 脇田
- 郵便番号 ← 東京都
- 必須項目 ← 空欄
- ゲーム中に想定外のキーの組み合わせが押された

困った状況.3

- Source.fromFile(“/../file.txt”)
  Linux と Mac な人の悩み

- Source.fromFile(“c:\\..\file.txt”)
  Windows な人の悩み

- Source.fromFile(“256文字以上のファイル・パス名”)

- 教訓:

Source.fromFile(“/../file.txt”)と類似の事例

- / は最上位のフォルダ、.. は親フォルダ ← そんなものは存在しない。

```
scala> import scala.io.Source
import scala.io.Source

scala> Source.fromFile("/../file.txt")
java.io.FileNotFoundException: /../file.txt (No such file or directory)
  at java.base/java.io.FileInputStream.open0(Native Method)
  at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
  at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
  at scala.io.Source$.fromFile(Source.scala:94)
  at scala.io.Source$.fromFile(Source.scala:79)
  at scala.io.Source$.fromFile(Source.scala:57)
  ... 28 elided

scala>
```

Source.fromFile(“/../file.txt”)と類似の事例

- Source.fromFile(“a.txt/b.txt”)
  a.txt はテキストファイルで、フォルダではない
- val root: Tree = …;　parent(root)
- val ある植物 = …;　get_鼻(ある植物)
- ビッグバンの5秒前に…

困った状況.4

- head([]), tail([]) – 空リストの内容
- Random.between(5, 3) – between(3, 5)ならいいけど
- max(Set.empty) – 空の集合の内容
- 教訓 – コレクションは空のこともあります

困った状況.5

- zip([1], [2, 3])
- $A : (m \times n), v \in R^k (k \neq m)$ に対して Av
- $A : (m \times n)$ matrix $(m \neq n)$に対してA^2 のような計算
- 教訓 – 形が合っているか気をつけましょう

困った状況.6

- 教務課データ.find(“ある卒業生のお名前”).呼び出し()

- 教訓 – 探しものが見つからないこともある

困った状況.7

- def f(): Int = { 0 + f() }

困った状況

- StackOverflowError – 再帰のしすぎ

```
⌥⌘1                                Default
dancer.local                  10/05 4:57 PM
(base)
 /tmp  cat a.scala
def f(): Int = { 0 + f() }
f()
(base)
 /tmp  scala a.scala |& head
Exception in thread "main" java.lang.StackOverflowError
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
        at Main$$anon$1.f(a.scala:1)
(base)
 /tmp
```

類似の状況

- val ints = Range(0, Int.MaxValue).**toList**
- OutOfMemoryError – メモリが足りません。

```
⌥⌘1                                Default
dancer.local                  10/05 5:06 PM
 /tmp  cat b.scala
val manyInts = Range(0, Int.MaxValue).toList
(base)
 /tmp  scala b.scala
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at scala.collection.mutable.ListBuffer.addAll(ListBuffer.scala:125)
        at scala.collection.mutable.ListBuffer$.from(ListBuffer.scala:38)
        at scala.collection.immutable.List$.from(List.scala:617)
        at scala.collection.IterableOnceOps.toList(IterableOnce.scala:1234)
        at scala.collection.IterableOnceOps.toList$(IterableOnce.scala:1234)
        at scala.collection.AbstractIterable.toList(Iterable.scala:921)
        at Main$$anon$1.<init>(b.scala:1)
        at Main$.main(b.scala:1)
        at Main.main(b.scala)
```

困った状況を区別して表現したものが例外

- build.sbt に
  javaOptions in run += “-Xms256M -Xmx2G” のようなの設定を追加すれば、メモリ領域を拡張できます。
  - スタックサイズ := 256MB、メモリサイズ := 2GB
- 演習ではそんな無茶な課題は出ないでしょうけど。

自分から例外を発生してみよう

例外を発生する2ステップ

- 例外を作成する
  - val e = new IllegalArgumentException(“やめて下さい”)
- 例外を投げる
  - throw e
- ふたつを一緒にしても構わない
  - throw new IllegalArgumentException(“やめて下さい”)

例外発生の効果的な例 (1/2)

- お馴染みの階乗計算 (lx04/throws.scala)
- object Factorial {
  def f(n: Int): Int = { … }
  }
- 困った状況
  - n < 0: 負の数は嫌いです
  - 計算結果が Int.MaxValue を越える: 大きな数は嫌いです

例外発生の効果的な例 (2/2)

```
def factorial(n: Int): Int = {
  if (n < 0) throw new IllegalArgumentException(“負の数は嫌！")
  if (n == 0) 1
  else {
    val v = n * factorial(n - 1)
    if (v < 0)
      throw new IllegalArgumentException(“大きな数は嫌！")

    v
  }
}
```

正常な実行例

- sbt:lx04> runMain Fact
  [info] Running (fork) Fact
  [info] factorial(0) = 1
  [info] factorial(1) = 1
  [info] factorial(2) = 2
  [info] factorial(3) = 6
  [info] factorial(4) = 24
  [info] factorial(5) = 120
  [info] factorial(6) = 720
  [info] factorial(7) = 5040
  [info] factorial(8) = 40320
  [info] factorial(9) = 362880
  [info] factorial(10) = 3628800

(-1)! の場合

- sbt:lx04> **runMain FactNegative**

```
[info] Running (fork) FactNegative
[info] --------------------------------------------------------------------
[info] 例外が発生しました。 -- java.lang.IllegalArgumentException: 負の数は嫌いです
[info] Factorial$.f(throws.scala:3)
[info] Factorial$.factorial$1(throws.scala:15)
[info] Factorial$.$anonfun$test$1(throws.scala:17)
[info] ExceptionsLab$.g$1(exceptions.scala:30)
[info] ExceptionsLab$.h$1(exceptions.scala:31)
[info] ExceptionsLab$.test(exceptions.scala:34)
[info] Factorial$.test(throws.scala:17)
[info] FactNegative$.delayedEndpoint$FactNegative$1(throws.scala:29)
[info] FactNegative$delayedInit$body.apply(throws.scala:27)
[info] scala.Function0.apply$mcV$sp(Function0.scala:34)
[info] --------------------------------------------------------------------
```

!17 の場合

- sbt:lx04> **runMain FactTooLarge**
  [info] Running (fork) FactTooLarge
  [info] ----------
  [info] **例外が発生しました。 -- java.lang.IllegalArgumentException: 大きな数は嫌いです**
  [info] Factorial$.f(throws.scala:8)
  [info] Factorial$.factorial$1(throws.scala:15)
  [info] Factorial$.$anonfun$test$1(throws.scala:17)
  [info] ExceptionsLab$.g$1(exceptions.scala:30)
  [info] ExceptionsLab$.h$1(exceptions.scala:31)
  [info] ExceptionsLab$.test(exceptions.scala:34)
  [info] Factorial$.test(throws.scala:17)
  [info] FactTooLarge$.delayedEndpoint$FactTooLarge$1(throws.scala:33)
  [info] FactTooLarge$delayedInit$body.apply(throws.scala:32)
  [info] scala.Function0.applymcVsp(Function0.scala:34)
  [info] ----------

例外を掴まえてみよう

try { … } catch { … } 構文

- try {
  よからぬことが起き得る式
  } catch {
  case e: 例外1 => { 例外1のときの処理 }
  case e: 例外2 => { 例外2のときの処理 }
  }

よからぬ事例？

- 規定の例外 – プログラムのデータとして規定されている。
- MatchError
- IllegalArgumentException
- UnsupportedOperationException
- java.lang.ArithmeticException
- java.lang.NumberFormatException
- java.io.FileNotFoundException
- java.io.IOException

try { … } catch { … } 構文の例(1/4)

-
```
def try_div(a: Int, b: Int): Int = {
  try { a / b
  } catch {
    case e: ArithmeticException => Int.MaxValue
  }
}
```

-
```
for (i <- -5 to 5) println(s"5 / $i = ${try_div(120, i)}")
```

try { … } catch { … } 構文の例(2/4)

- var number_format_errors = 0
- def print_number(num: String): Unit = {

```
  try { println(s"$num => ${Integer.parseInt(num)}")
  } catch {
    case e: NumberFormatException =>
      number_format_errors = number_format_errors + 1
  }
}
```

- for (num <- List("1", "Zero", "零", "壱")) { print_number(num) }

  println(s”入力エラー数: **$number_format_errors**")

try { … } catch { … } 構文の例(3/4)

- def try_first_line(path: String): String = {
  try { Source.fromFile(path).getLines().take(1).mkString
  } catch {
  case e: FileNotFoundException => "<<<ファイルは見つかりませんでした>>>"
  }
  }
- val filenames = List("trycatch.scala", "こんなファイルは存在しない", "throws.scala")
- for (filename <- filenames) {
  println(s"$filename => ${try_first_line(filename)}")
  }

まとめ

- 例外を作る構文: val e = **new 例外(“メッセージ”)**
- 例外を発生する構文: **throw** e
- 例外を掴まえる構文:

```
try { よからぬことが起き得る式や文
} catch {
  case e: 例外1 => { 例外1のときの処理 }
  case e: 例外2 => { 例外2のときの処理 }
}
```