

見せ算をScalaで 実装してみた

id:arthur-1 株式会社はてな

2024-02-27 Scala わいわい勉強会 #2 LT



Arthur と申します

株式会社はてな
アプリケーションエンジニア

前回のScalaわいわい勉強会でも
LTさせてもらいました！

X: @Arthur1__



Mackerel 作ってます



製品 ▾ 料金 事例 資料 サポート お知らせ

ログイン

無料トライアル

お問い合わせ

News

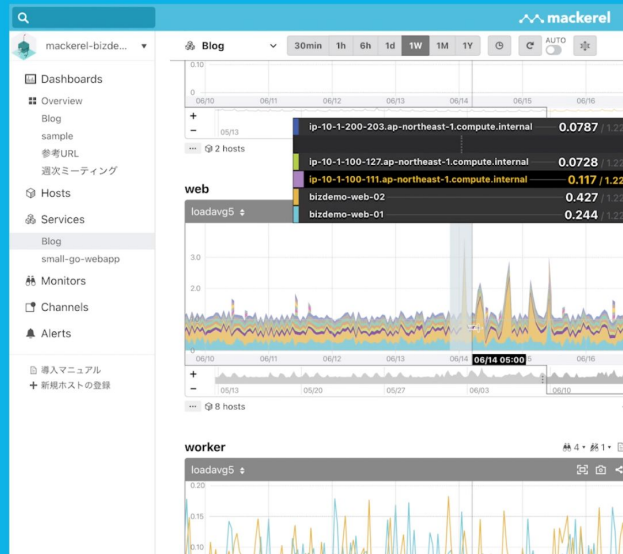
2023.09.29

【開催レポート】Mackerel Drink Up #11 Tokyoを開催しました！

監視を育てる、 「Mackerel」

クラウド時代に最適な監視モデルを
使いやすいUIで提供し、
システムの運用・監視に
チームで取り組む文化を作る
「クラウド運用の道標」となる
SaaS型サーバー監視サービス。

無料トライアルをはじめる



最近、
キャリア悩み中



あんまりエンジニアに
こだわってもいけない



お笑い芸人も
いいな～



M-1 2023
見ましたか？





<https://www.youtube.com/watch?v=QSfNwVBg3EA>



決勝の出来事

お笑いコンビ「さや香」の新山が
四則演算に加えて5則目の「**見せ算**」を提唱

“数字と数字を見せ合わせて、どう思うか？”



見せ算の基本ルール: その1

同じ数字を「見せ」すると、その「眼」は0になる

[例] 1見せ1=0

同じ服装の人が前からやってくると恥ずかしく
なってお互い立ち去る



見せ算の基本ルール: その2

違う数字を「見せ」すると、大きい方が「眼」になる

[例] 1見せ2=2

小さい者が大きい者を見ると怖くなって逃げる



見せ算の応用ルール

- 6見せ9=11
 - 似てるな〜と気になってお互い近づく
- 2見せ5=1.1
 - 同様に近寄るも、「違う！」とびっくりして携帯を落とす
- 1見せ100=83
 - 1が逃げられないと腹を括って突っ込み17人倒す



そろそろ タイトルコール



見せ算をScalaで 実装してみた

id:arthur-1 株式会社はてな

2024-02-27 Scala わいわい勉強会 #2 LT



なぜ Scala か

中置演算子を自由に作れる！

2 **mise** 5 のように書ける

- 演算子はメソッド
- 既存の型の拡張ができる



演算子はメソッド

```
case class MyInt (v: Int) {  
  def unary_- : MyInt = MyInt(-this.v)  
  def +(that: MyInt): MyInt = MyInt(this.v + that.v)  
}
```

```
-MyInt(4) // MyInt(-4)
```

```
MyInt(3) + MyInt(2) // MyInt(5)
```

<https://scastie.scala-lang.org/N10goNFbQj6eOwmPzGPDRQ>



既存の型の拡張ができる

ビルトインの型にも演算子を増やせる

```
implicit class ExtendedInt(val i: Int) {  
  def **(j: Int): Int = if (j <= 0) 1 else i ** (j - 1) * i  
}
```

```
3 ** 2 // 9
```



<https://scastie.scala-lang.org/CvFRoSDPQF6gWilg9hHCKQ>

Scala 3 からは.....

extensionキーワードでより直感的に書ける

```
extension (i: Int)  
  def **(j: Int): Int = if (j <= 0) 1 else i ** (j - 1) * i
```

```
3 ** 2 // 9
```



<https://scastie.scala-lang.org/ly6je4cNRwelh4IRbGSi6g>

見せ算の実装はこんな感じ

```
extension (i: Int)
  def mise(j: Int): BigDecimal = (i min j, i max j) match
    case (6, 9)           => 11
    case (2, 5)           => 1.1
    case (1, 100)         => 83
    case (small, big) => if (small == big) 0 else big
```



<https://scastie.scala-lang.org/LrLMjihdQMejwFhiTvHphQ>

GitHubに上げておきました

<https://github.com/Arthur1/misezan-scala>

```
test(  
  "応用: 2見せ5=1.1(お互いが生き別れの兄弟と勘違いして近寄るがよく見ると全然違うことに気付きびっくりして携帯「。」を落としてしまうため。)  
  ) {  
    2 mise 5 shouldBe 1.1  
    5 mise 2 shouldBe 1.1  
  }  
  
test("応用: 1見せ100=83(あまりにも人数差がありもう逃げても仕方ないと1が腹をくくって100に立ち向かい17人倒すため。)") {  
  1 mise 100 shouldBe 83  
  100 mise 1 shouldBe 83  
}  
}
```



余談その1

Scala 3のコードでは**Optional Braces Syntax**を使ってみたけど皆さんは好きですか？

Tour of Scalaなどのコードも{}なくなってる

<https://docs.scala-lang.org/tour/pattern-matching.html>



{ }をちゃんとつけるとこんな感じ

```
extension (i: Int) {  
  def mise(j: Int): BigDecimal = (i min j, i max j) match {  
    case (6, 9)          => 11  
    case (2, 5)          => 1.1  
    case (1, 100)        => 83  
    case (small, big) => if (small == big) 0 else big  
  }  
}
```



余談その2

見せ算を使ったアプリケーション開発を支援するため
Maven Central Repositoryにpublishしようと思った
のだが.....

Sonatype has begun the retirement process for issues.sonatype.org.

If you have been instructed to sign up for a new account here in an e-mail from central-support@sonatype.com, you can ignore this notice.

For further details, please visit https://central.sonatype.org/news/20240109_issues_sonatype_org_deprecation/.

The **MVNCENTRAL** and **OSSRH** projects are now read-only and will be accessible for a few more weeks so that you can refer to previously created issues when sending support requests to central-support@sonatype.com

If you currently publish to Maven Central via oss.sonatype.org or s01.oss.sonatype.org, this change only impacts how you request publishing support, which you can now do by emailing central-support@sonatype.com. There should be no impact to your existing permissions or publishing processes.

If you are looking for the **NEXUS** project, please see <https://github.com/sonatype/nexus-public/issues/105>.



助けてくれ！

issues.sonatype.orgが 2024年2月に廃止された
最近アカウント作ったユーザーは、OSSRHではなく
Central Portal経由でpublishするしかない

- アップロード先のendpointやAPIも変わっていきそう
- これに対応したsbt pluginの情報など大募集中です



おしまい？



ここからは 大学院の講義



型レベルプログラミング版

<https://github.com/Arthur1/misezan-scala-type>

```
test("基本: 小さい数字に大きい数字を見せ合うと大きい数字が残る。(小さい数が大きい数を怖がって逃げてしまうため。)") {  
  implicitly[Aux[_1, _2, _2]]  
  implicitly[Aux[_2, _1, _2]]  
  implicitly[Aux[_3, _0, _3]]  
  
  mise(_1, _2): _2  
  mise(_2, _1): _2  
  mise(_3, _0): _3  
  
  mise(_1, _2).toInt shouldBe 2  
  mise(_2, _1).toInt shouldBe 2  
  mise(_3, _0).toInt shouldBe 3  
}
```



実装の紹介

こちらはScala 2で実装

AUXパターン:

引数と結果に相当する型のマッピングを作るイメージ
場合分けは型制約とimplicitの解決で実現

面倒なので基本的な型はsharplessに頼ってみた
まだ基本ルールだけなので今後の自分に期待



おしまい

