Scala言語紹介

Learn Languages 2017

株式会社ドワンゴ　水島宏太

自己紹介

- Twitter ID: @kmizu
- GitHub: kmizu
- Love: Nemerle/Rust/Standard ML/Scala/...
  - 言語オタク
- 自分のプログラミング言語も作ってます
  - Klassic
- Japan Scala Association代表理事

Scalaとは

- 2003年、 Martin Odersky教授によって開発
  - スイス連邦工科大学（EPFL）
- 最新安定版： Scala 2.12
- **Sca**lable **La**guage
- オブジェクト指向と関数型の融合
- 静的型付き言語
- 全ての値がオブジェクト
- 主にJVM向けのコードを吐く

Hello, World!

```
object Main extends App {
  println("Hello, World!")
}
```

- App は main メソッドを持つ

全ての値はオブジェクト

```
1 + 2 // (1).+(2)
1 - 2 // (1).-(2)
1 * 2 // (1).*(2)
1 / 2 // (1)./(2)

"FOO" substring 1 // "Foo.substring(1)
```

var と val

- var は再代入可能

```
var i: Int = 0
println(i) // => 0
i += 1
println(i) // => 1
```

- val は再代入不可

```
var i: Int = 0
println(i)
i = i + 1 // error: reassignment to val
println(i)
```

統一的な型階層

- 全ての型は Any のサブタイプ
- Nothing は全ての型のスーパータイプ

```
val any1: Any = 1
val any2: Any = "FOO"
```

```
// ??? は Nothingを返すメソッド
// 未実装のメソッドを表すのに使える
def add(x: Int, y: Int): Int = ???
```

case classes

```
case class Person(name: String, age: Int)
...
val p = Person("Kota Mizushima", 33)
val q = p.copy(age = 34) // 一部分を更新
```

- 不変オブジェクトを作るのに便利

代数的データ型とパターンマッチ

```
sealed abstract class ParseResult[+A]
case class Success[+A](value: A, next: String)
case class Failure(message: String)
```

```
val r: ParseResult[String] = ...
r match {
  case Success(v, n) =>
  case Failur(m) =>
}
```

- 様々な用途
  - 列挙型
  - エラー処理
  - DSL
  - ...

不変コレクションとパターンマッチ

```
val list: List[Int] = List(1, 2, 3)
```

```
def sum(list: List[Int]): Int = list match {
  case x::xs => x + sum(xs)
  case Nil => 0
}
```

ローカル型推論

```
/* val list: List[Int] */
val list = List(1, 2, 3)

/* val vec: Vector[String] */
val vec = Vector("A", "B", "C")
```

第一級関数と高階関数

```
val src = List(1, 2, 3, 4, 5)

// val dbl == List(2, 4, 6, 8, 10)
val dbl = src.map(x => x * 2)

/* lt2 == List(3, 4, 5) */
val lt2 = src.filter(x => x > 2)

/* val sum = 15 */
val sum = src.foldLeft{(x, acc) => acc + x}
```

名前付き引数とデフォルト引数

```
def add(x: Int = 0, y: Int = 0): Int = x + y
```

```
add(x = 1, y = 2)
// 順番を入れ替えてもOK
add(y = 2, x = 1)
// y = 0
add(x = 1)
```

割愛した機能

- implicit parameter
  - Haskellの型クラスに対応
- 名前呼び出し
- コンパニオン（クラス｜オブジェクト）
- ...