



Scalanus

Marek Kaput, Radomir Krawczykiwicz

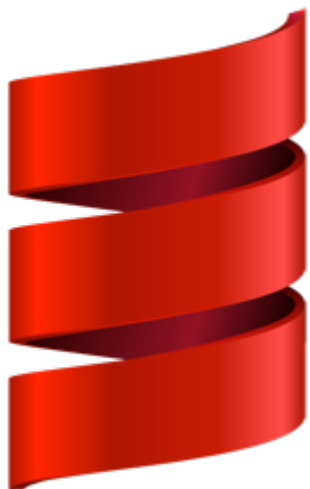
<https://github.com/mkaput/scalanus>



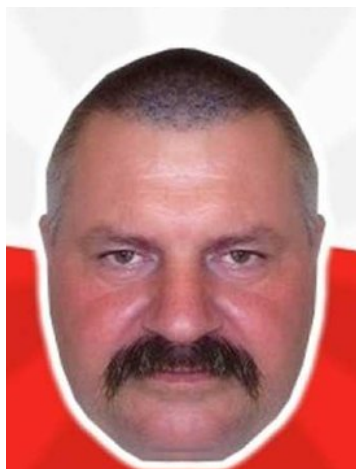
+



Znowu?



+



=




 Tak

Czyli...

- Interpretowany język programowania
- Kontynuacja języka Janus
- Dostępny jako biblioteka dla Javy/Scali lub jako oddzielny program
- Prawie da się używać!

Co nowego?

- Typy złożone – krotki, tablice i słowniki
 - Prosty pattern matching
 - Wywoływanie metod
 - Pętla for
- 

Czego dalej nie ma?

- W sumie to się nie wyrobiliśmy...
- Modułów
- Obiektowości
- Lambd
- Biblioteki standardowej

Przykład

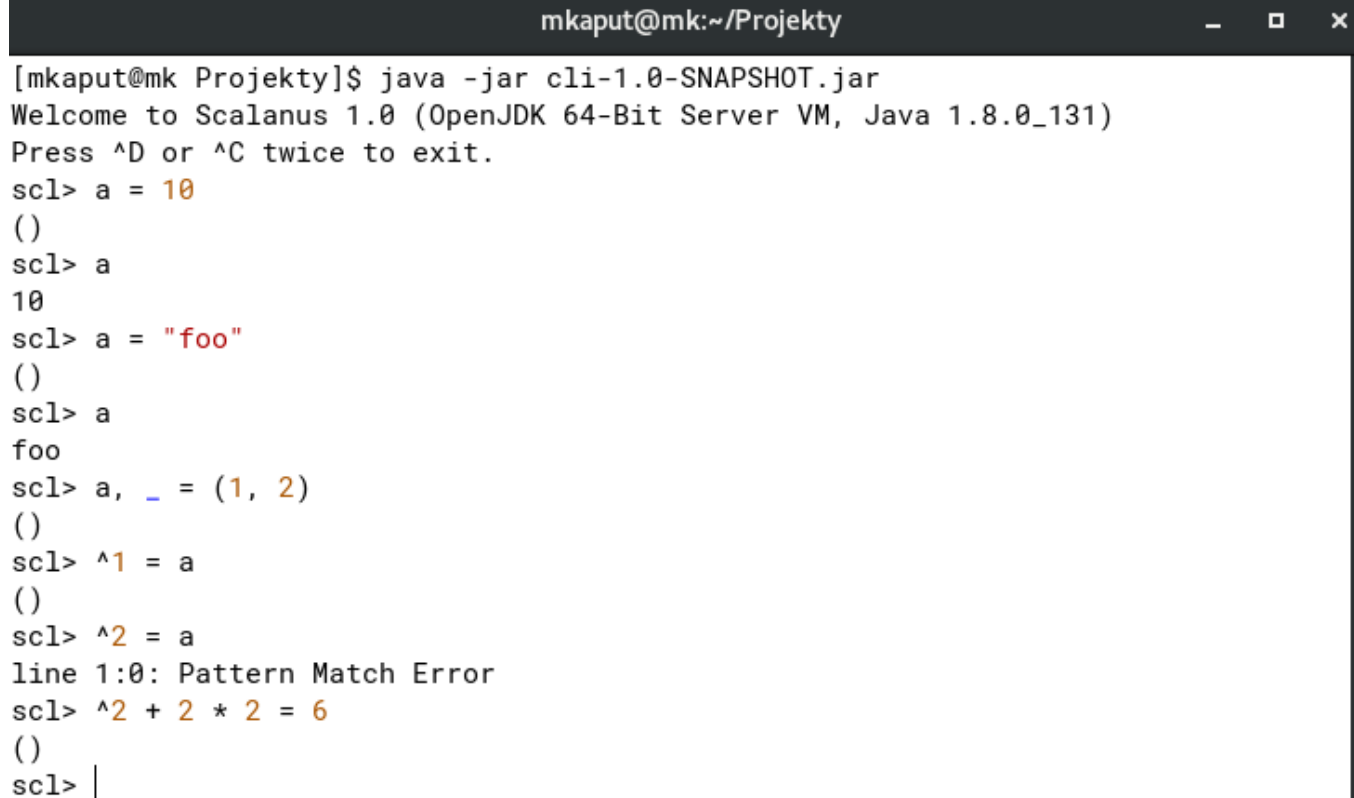
```
numbers = #{ "foo" = 1, "bar" = 2, "moo" = 3 }
```

```
numbers["moo"] = 5 // średniki są opcjonalne
```

```
fn funnyPrint(name, number) {  
    IO.println(name + " = " + number)  
}
```

```
for name, number in numbers {  
    funnyPrint(  
        // prawie wszystko jest wyrażeniem  
        if number mod 2 == 0 {  
            name.toUpperCase()  
        } else {  
            name  
        },  
        number  
    )  
}
```

Pattern matching



A terminal window titled 'mkaput@mk:~/Projekty' showing a REPL session for Scalanus 1.0. The session starts with a welcome message and instructions to press ^D or ^C to exit. The user enters several commands: 'a = 10', 'a', 'a = "foo"', 'a', 'a, _ = (1, 2)', '^1 = a', '^2 = a', and an error message 'line 1:0: Pattern Match Error'. The session ends with '^2 + 2 * 2 = 6' and a prompt line.

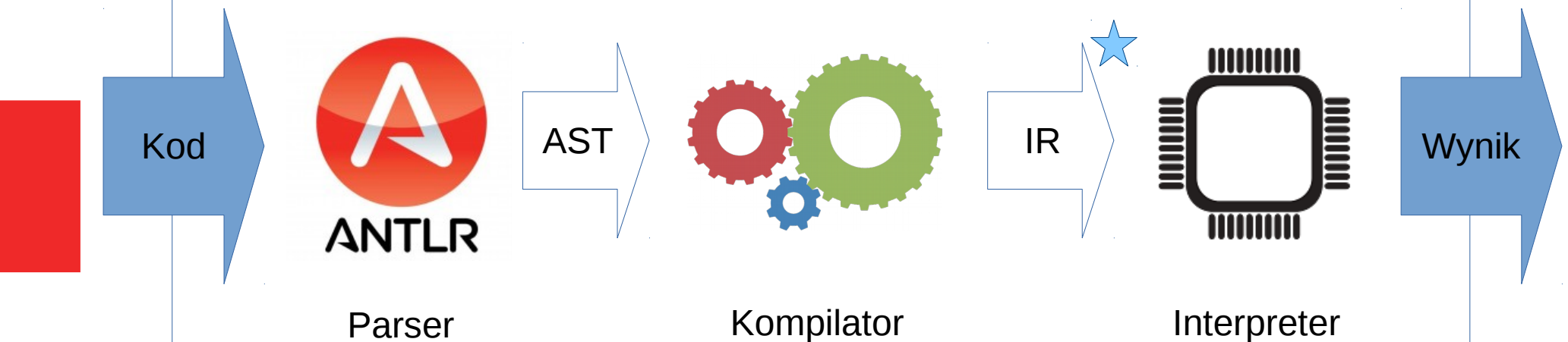
```
mkaput@mk:~/Projekty
[mkaput@mk Projekty]$ java -jar cli-1.0-SNAPSHOT.jar
Welcome to Scalanus 1.0 (OpenJDK 64-Bit Server VM, Java 1.8.0_131)
Press ^D or ^C twice to exit.
scl> a = 10
()
scl> a
10
scl> a = "foo"
()
scl> a
foo
scl> a, _ = (1, 2)
()
scl> ^1 = a
()
scl> ^2 = a
line 1:0: Pattern Match Error
scl> ^2 + 2 * 2 = 6
()
scl> |
```




Implementacja

Architektura

JSR 223: Scripting for the Java™ Platform



JSR 223

- Standard i zestaw klas bazowych do tworzenia języków skryptowych w Javie

```
val engine = new ScriptEngineManager().getEngineByName("scalanus")
```

```
engine.eval("2 + 2")
```

```
val engine = new ScriptEngineManager().getEngineByName("javascript")
```

Antlr



Download
v4.7



Dev Tools



Book



Doc



About ANTLR



Support



Bugs

What is ANTLR?



ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.

Looking for ANTLR v3?



Terence Parr is the maniac behind ANTLR and has been working on language tools since 1989. He is a professor of computer science at the University of San Francisco.

Quick Start

Samples

Samples



```
grammar Expr;
prog:  (expr NEWLINE)* ;
expr:  expr ('*' | '/')
      | expr ('+' | '-' )
      | INT
      | '(' expr ')'
      ;
NEWLINE : [\r\n]+ ;
INT      : [0-9]+ ;
```

```
$ antlr4 Expr.g4
$ javac Expr*.java
$ grun Expr prog -gui
100+2*34
^D
```



Jak to wygląda w całości?

```
// Parser
whileLoop : 'while' expr block ;
```

```
// IR
case class IrWhileExpr(cond: IrExpr, routine: IrExpr)(ctx: IrCtx) extends IrNode(ctx) with IrExpr
```

```
// Kompilator
override def visitWhileExpr(ctx: WhileExprContext): Option[IrNode] = compileM(ctx) {
  val loop = ctx.whileLoop
  for {
    cond <- accept[IrExpr](loop.expr)
    routine <- accept[IrExpr](loop.block)
  } yield IrWhileExpr(cond, routine)
}
```

```
// Interpreter
def evalWhileExpr(irWhileExpr: IrWhileExpr, context: ScalanusScriptContext, scope: Int): Unit = {
  while(evalExpr(irWhileExpr.cond, context, scope).equals(true)){
    try {
      evalExpr(irWhileExpr.routine, context, scope)
    } catch {
      case _: ScalanusBreak => return
      case _: ScalanusContinue => // do nothing
    }
  }
}
```

CLI

- Możliwość wykonywania plików jak i tryb REPL
- Scopt – parser argumentów
- JLine3 – biblioteka do rozbudowanej obsługi konsoli

Koloruje składnię!

Historia poleceń!

Można więcej!



Demo