

Praktyczne programowanie w Groovy'm

Dla kompletnie zielonych

Prowadzący

Łukasz Krauzowicz

Programista,

Miłośnik ekosystemu JVM,

Entuzjasta Otwartego Oprogramowania

lukasz.krauzowicz@ericsson.com

<https://github.com/kraluk>

<https://github.com/ericsson>



Zasady

Zasada warsztatów



Agenda

Agenda

- **Groovy**
 - Co to jest?
 - Groovy vs. Java
 - Dlaczego ten język?

Agenda (II)

- **Zaprogramujmy coś!**
 - Zmienne
 - Podstawowe interakcje
 - Operacje na tekście
 - Instrukcje warunkowe
 - Pętle
 - Funkcje
 - Obsługa plików*
 - Klasy*

Agenda (III)

- **Groovy w akcji!**
 - Własny program
 - Komunikacja ze światem!



Oprogramowanie

Oprogramowanie

- **Java Development Kit (JDK) 8**

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



- **Groovy Development Kit (GDK) 2.4**

<http://www.groovy-lang.org/download.html>



Narzędzia

Groovy Shell (**groovysh**)

```
→ /cygdrive/c/Users/lkra groovysh
Groovy Shell (2.4.7, JVM: 1.8.0_102)
Type ':help' or ':h' for help.

groovy:000> :h
:n

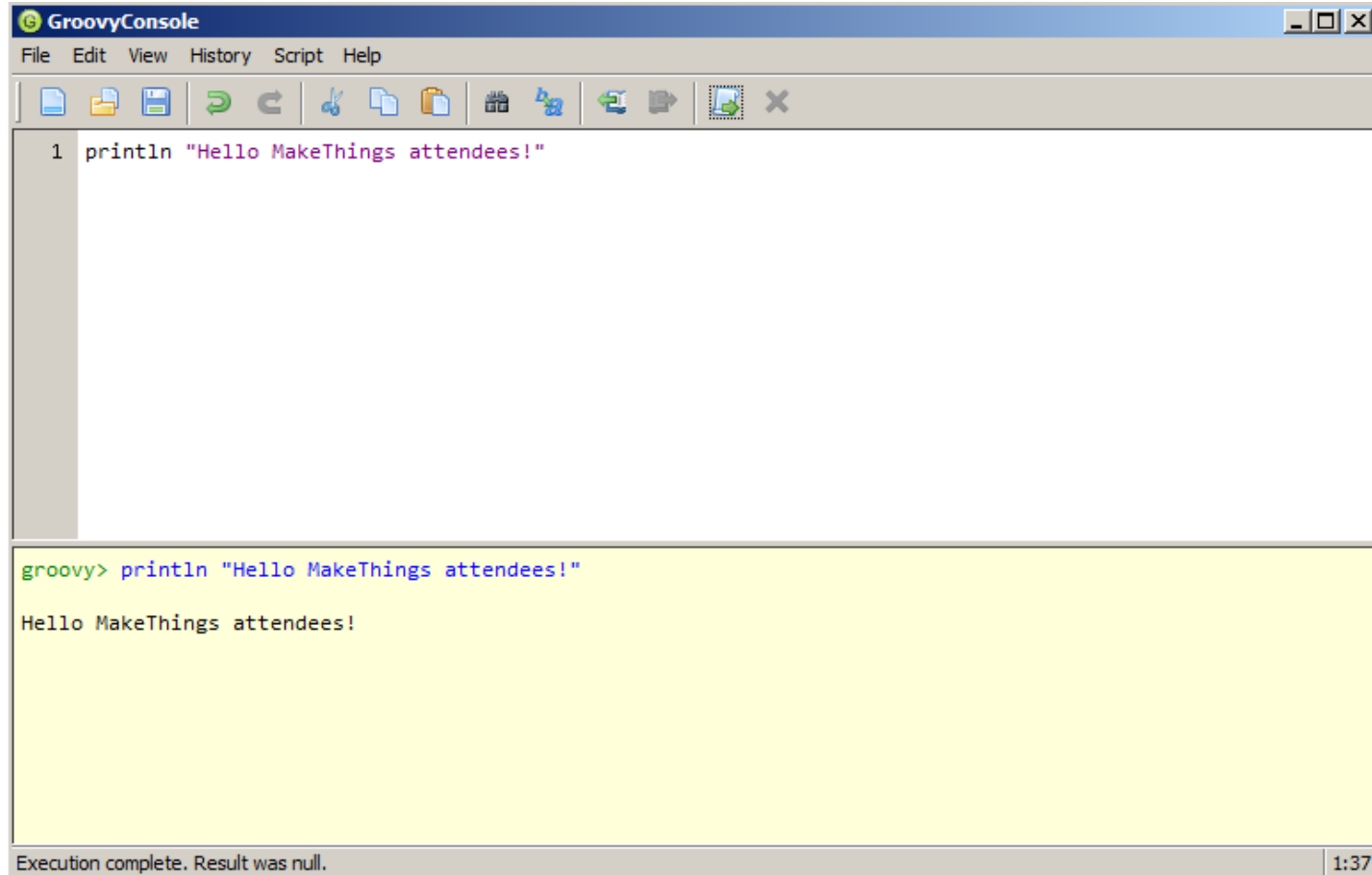
For information about Groovy, visit:
  http://groovy-lang.org

Available commands:
:help      (:h ) Display this help message
?          (:? ) Alias to: :help
:exit      (:x ) Exit the shell
:quit      (:q ) Alias to: :exit
:import    (:i ) Import a class into the namespace
:display   (:d ) Display the current buffer
:clear     (:c ) Clear the buffer and reset the prompt counter
:show      (:S ) Show variables, classes or imports
:inspect   (:n ) Inspect a variable or the last result with the GUI object browser
:purge     (:p ) Purge variables, classes, imports or preferences
:edit      (:e ) Edit the current buffer
:load      (:l ) Load a file or URL into the buffer
.          (.. ) Alias to: :load
:save      (:s ) Save the current buffer to a file
:record    (:r ) Record the current session to a file
:history   (:H ) Display, manage and recall edit-line history
:alias     (:a ) Create an alias
:set       (:= ) Set (or list) preferences
:register  (:rc) Register a new command with the shell
:doc       (:D ) Open a browser window displaying the doc for the argument

For help on a specific command type:
  :help command

groovy:000> |
```

GroovyConsole (Windows)



```
1 println "Hello MakeThings attendees!"
```

```
groovy> println "Hello MakeThings attendees!"  
Hello MakeThings attendees!
```

Execution complete. Result was null. 1:37

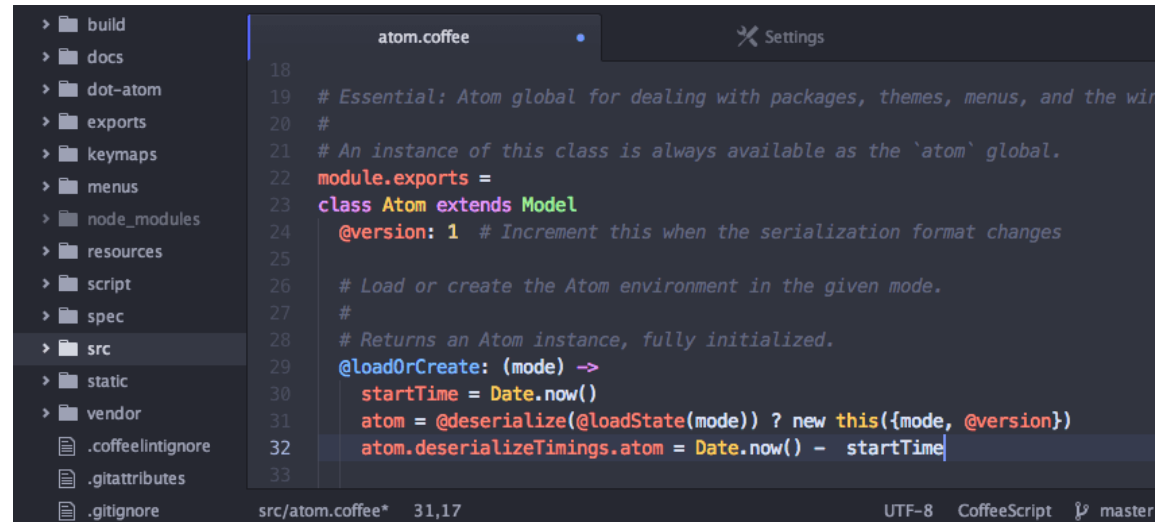
groovy

groovy program.groovy

```
→ /cygdrive/c/Users/lkra groovy hello.groovy  
Hello from the script!
```

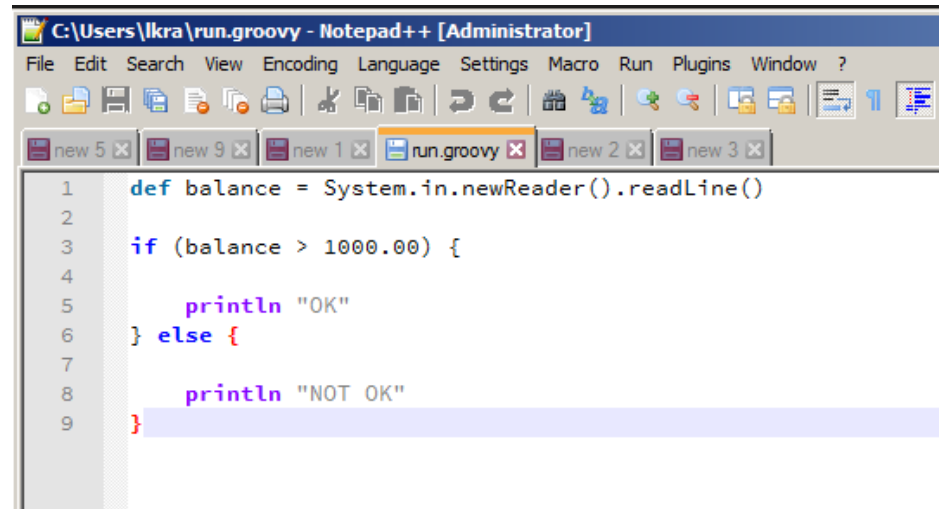
Edytor kodu

- Notatnik
- Notepad++
- Atom
- ...



The screenshot shows the Atom code editor interface. On the left is a file explorer sidebar with a tree view containing folders like 'build', 'docs', 'dot-atom', 'exports', 'keymaps', 'menus', 'node_modules', 'resources', 'script', 'spec', 'src', 'static', and 'vendor'. The 'src' folder is selected. The main editor area displays the 'atom.coffee' file with CoffeeScript code. The code includes comments and defines the 'Atom' class, which extends 'Model'. It sets a version number and implements a '@loadOrCreate' method that initializes the Atom environment by loading or creating a state and deserializing it. The status bar at the bottom indicates the file path 'src/atom.coffee', line 31, column 17, and the encoding 'UTF-8'.

```
18
19 # Essential: Atom global for dealing with packages, themes, menus, and the win
20 #
21 # An instance of this class is always available as the `atom` global.
22 module.exports =
23   class Atom extends Model
24     @version: 1 # Increment this when the serialization format changes
25
26     # Load or create the Atom environment in the given mode.
27     #
28     # Returns an Atom instance, fully initialized.
29     @loadOrCreate: (mode) ->
30       startTime = Date.now()
31       atom = @deserialize(@loadState(mode)) ? new this({mode, @version})
32       atom.deserializeTimings.atom = Date.now() - startTime
33
```



The screenshot shows the Notepad++ code editor window titled 'C:\Users\lkra\run.groovy - Notepad++ [Administrator]'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. The editor has several tabs open, with 'run.groovy' being the active one. The code is a Groovy script that reads a line from the standard input and checks if the value is greater than 1000.00. If true, it prints 'OK'; otherwise, it prints 'NOT OK'.

```
1 def balance = System.in.newReader().readLine()
2
3 if (balance > 1000.00) {
4
5     println "OK"
6 } else {
7
8     println "NOT OK"
9 }
```

Groovy

Groovy – co to jest?

- opcjonalnie typowany język dynamiczny zorientowany obiektowo
- oparty i wpisujący się w ekosystem **JVM**
- pierwsza publiczna wersja – 2003
- aktualna wersja – **2.4.7**
- w pełni współpracujący z Javą
- aktualnie rozwijany przez Apache Software Foundation



Groovy vs. Java

Program:

- Tworzy listę osób
- Sortuje listę po nazwisku i imieniu
- Pokazuje osoby starsze niż 20 lat
- Zapisuje wyniki do pliku

Java:

- LOC: 319
- Czas tworzenia: ~ 2 h
- Czas wykonania: ~ 30 ms
- Deweloper: ~ 14 lat doświadczenia

Groovy:

- LOC: 115
- Czas tworzenia: ~ 1 h
- Czas wykonania: ~ 700 ms
- Deweloper: < 1 miesiąc doświadczenia

Dlaczego Groovy?

- opcjonalnie typowany
- prostsza składnia niż np. w Javie
- prosty w użyciu (według mnie)
- lubię go 😊



Teoria w praktyce

Uruchamiamy groovys...

```
:= interpreterMode
```

Zmienne

Zmienne

```
def number = 69
```

```
def floating = 6.9
```

```
def message = „Hello!”
```

```
def elements = [„first”, „second”, „third”]
```

Zmienne (II)

```
int number = 69
```

```
double floating = 6.9
```

```
String message = „Hello!”
```

```
String[] elements = [„first”, „second”, „third”]
```


Operacje na zmiennych

```
def sum = 4 + 5
```

```
def sub = 4 - 5
```

```
def mul = 4 * 5
```

```
def div = 4 / 5
```

```
def sum = a + b
```

```
def sub = a - b
```

```
def mul = a * b
```

```
def div = a / b
```

Operacje arytmetyczne

`==`

```
def result = 5 == 5
```

`< | <=`

```
def result = 4 < 5
```

`> | >=`

```
def result = 5 >= 4
```

`!=`

```
def result = 4 != 5
```

Zadania (1)

1. Do zmiennej **a** przypisz wartość **55**
2. Do zmiennej **b** przypisz wartość **45**
3. Wykonaj dowolną operację arytmetyczną na nich i jej wynik przypisz do kolejnej zmiennej.
4. * Wykonaj podobne ćwiczenie, ale ze zmiennymi tekstowymi.

Interakcja z użytkownikiem

Wypisywanie zmiennych

```
def name = „Tadek”
```

```
println „Hello, ” + name
```

```
println „Hello, $name”
```

```
println(„Hello, ” + name)
```

A teraz napiszemy skrypty...

1. Stwórz plik **zadanie.groovy** → 2. Napisz kod → Uruchom go **groovy zadanie.groovy**

Pobieranie danych z klawiatury

`System.in.newReader().readLine()` *

*****) Niestety z GroovyConsole nie za bardzo współpracuje... ☹

Pobieranie danych z klawiatury

```
println „What is your name?”
```

```
println „Your name is ${System.in.newReader().readLine()}”
```


Pobieranie danych z klawiatury

```
println „Who are you?”
```

```
def name = System.in.newReader().readLine()
```

```
println „You are $name!”
```

Zadania (2)

1. Poproś użytkownika o hasło
2. Do zmiennej **password** przypisz podaną przez niego wartość
3. Wyświetl użytkownikowi informację o niepoprawnym hasle i wyświetl mu je 😊

```
def variable = System.in.newReader().readLine()
```

Operacje na tekście

Operacje na tekście

```
def text = „Some message”  
„Some message”
```

```
def result = text.equals(„nothing”)  
def result = „nothing”.equals(„nothing”)
```

Operacje na tekście

`boolean equals(„text”)`

`boolean equalsIgnoreCase(„text”)`

`boolean startsWith(„prefix”)`

`boolean endsWith(„postfix”)`

`String replace(„target”, „replacement”)`

`String toUpperCase()`

`String toLowerCase()`

`int length()`

<http://docs.groovy-lang.org/docs/groovy-2.4.7/html/groovy-jdk/java/lang/String.html>

Zadania (3)

1. Poproś użytkownika o wpisanie frazy „MAKETHINGS”
2. Sprawdź, czy faktycznie takie słowo zostało podane, niezależnie od wielkości liter

`System.in.newReader().readLine()`

`equalsIgnoreCase()`

Instrukcje warunkowe

Instrukcje warunkowe - **if**

```
if (condition) {  
    ...  
} else if (condition) {  
    ...  
} else {  
    ...  
}
```


Instrukcje warunkowe - **if**

```
def condition = true

if (condition == true) {
    println „TRUE”
} else {
    println „FALSE”
}
```

Instrukcje warunkowe - **if**

```
def condition = "m"

if ("m".equals(condition)) {

    println "Yay, you are $condition"
} else if ("k".equals(condition)) {

    println "Ay, you are $condition"
} else {
    println "Unknown choice :-(
}
```

Instrukcje warunkowe - **switch**

```
switch (condition) {  
  
    case "m":  
        println "Yay, you are $condition"  
        break;  
  
    case "k":  
        println "Ay, you are $condition"  
        break;  
  
    default:  
        println "Unknown choice :-( "  
  
}
```

Zadania (4)

1. Do zmiennej **balance** przypisz stan konta losowej osoby
2. Do zmiennej **amount** przypisz wartość do wypłaty z konta
3. Jeśli wypłata będzie większa niż stan konta wypisz odpowiedni komunikat, w przeciwnym wypadku pomniejsz stan konta o jej wartość i wypisz nowy stan konta.

```
if (condition) {  
    ...  
} else {  
    ...  
}
```

Petle

Petle – **for**

```
for (start_condition; end_condition; operation) {  
    ...  
}
```

Pȩtle – **for**

```
for (def i = 0; i < 5; i++) {  
    println "$i"  
}
```

Peçle – **for**

```
def array = [8, 7, 6, 5, 4]
```

```
for (i in array) {  
    println "$i"  
}
```


Peşle – **for**

```
for (i in [8, 7, 6, 5, 4]) {  
    println "$i"  
}
```

```
for (i in 8..4) {  
    println "$i"  
}
```

Perl – **while**

```
while (condition) {  
    ...  
}
```

Pętle – **while**

```
def var = 0
def stop = 9

while (var <= stop) {
    println „$var”
    var = var + 1
}
```

Zadania (5)

1. Wypisz liczby od 10 do 20 przy użyciu dowolnej pętli

```
for (start; stop; operation) {  
    println "$i"  
}
```

Funkcje

Funkcje

```
def functionName() {  
    result  
}
```

```
def functionName(param) {  
    result + param  
}
```

Funkcje

```
def sum(a, b) {  
    a + b  
}
```

```
def sum(int a, int b) {  
    a + b  
}
```

Obsługa plików

Obsługa plików*

```
def fileName = /c:\files\file.txt/  
def file = new File("$fileName")  
  
file.eachLine { line ->  
    java.util.regex.Matcher matcher = line =~ /word/  
    println matcher.count  
}  
  
file.eachLine { line ->  
    println line =~ /^.*word.*$/  
}
```

Klasy

Klasy*

```
class Person {  
  
    def firstName  
    def lastName  
  
    def getFullName() {  
        return firstName + " " + lastName  
    }  
}
```

Klasy*

```
def person = new Person(firstName: "Luke", lastName: "Skywalker")

println("Hello $person.firstName, the one from $person.lastName's")
println("Ello, " + person.getFullName())

person.lastName = "Vader"

println("Hello $person.firstName, the one from $person.lastName's")
```



Groovy w akcji!

Część 1

- Napisz program, który wczyta z klawiatury do zmiennej:
 - **Wiadomość**



Część 2

- Dodaj kolejną operację, która wczyta z klawiatury do zmiennej:
 - **Typ wiadomości** – dozwolone są litery **S** lub **M**

Część 3

- Przy użyciu instrukcji warunkowej i wartości **typ wiadomości** poproś użytkownika o:
 - dla **S** – numer telefonu
 - dla **M** – adres email

Część 4

- W pierwszych dwóch liniach swojego pliku dodaj następujący fragment kodu:

```
@Grab('org.codehaus.groovy.modules.http-builder:http-builder:0.7')  
import groovyx.net.http.RESTClient
```

Część 5

- Przed instrukcją warunkową dodaj:

```
def client = new RestClient('http://localhost:8080/')
```

- W instrukcji warunkowej:

- Dla S dodaj:

```
def response = client.get(path : "sms/$numb/$message")
```

- Dla M dodaj:

```
def response = client.get(path : "mail/$mail/$message")
```

Część 6

- Uruchom swój program!

groovy program.groovy

- Obserwuj swoją skrzynkę pocztową lub telefon 😊

DZIEKUJE ZA UWAGE



WSZYSTKIM, KTÓRZY NIE ZASNĘLI

Koniec



lukasz.krauzowicz@ericsson.com



<https://github.com/kraluk/make-things>