

AN INTRODUCTION TO GROOVY

JEFF BROWN

G2ONE - DIRECTOR

NORTH AMERICAN OPERATIONS

JEFF@G2ONE.COM

NO FLUFF JUST STUFF

- NO FLUFF JUST STUFF
- LOT OF GROOVY / GRAILS COVERAGE
- 5 TRACKS - 55 SESSIONS, BOFS, EXPERT PANELS
- LIMITED ATTENDANCE
- GREAT SPEAKERS
- AUSTIN - JULY 11-13

WHAT IS GROOVY?

- AGILE DYNAMIC LANGUAGE FOR THE JVM
- INSPIRED BY LANGUAGES SUCH AS...
 - PYTHON
 - RUBY
 - SMALLTALK

WHY GROOVY

- POWERFUL DYNAMIC LANGUAGE
- RELATIVELY EASY TO LEARN
- FAMILIAR SYNTAX FOR JAVA PROGRAMMERS
- INTEGRATES REALLY WELL WITH JAVA
- CONTAINERS, LIBRARIES, EXISTING JAVA CODE

INSTALLING GROOVY

- DOWNLOAD LATEST RELEASE
 - [HTTP://GROOVY.CODEHAUS.ORG/](http://groovy.codehaus.org/)
- EXTRACT ARCHIVE
- SET \$GROOVY_HOME
- ADD \$GROOVY_HOME/BIN TO PATH

GROOVY TOOLS

- GROOVY - INTERPRETER
- GROOVYC - COMPILER
- GROOVYSH - SHELL
- GROOVYCONSOLE - SWING CONSOLE

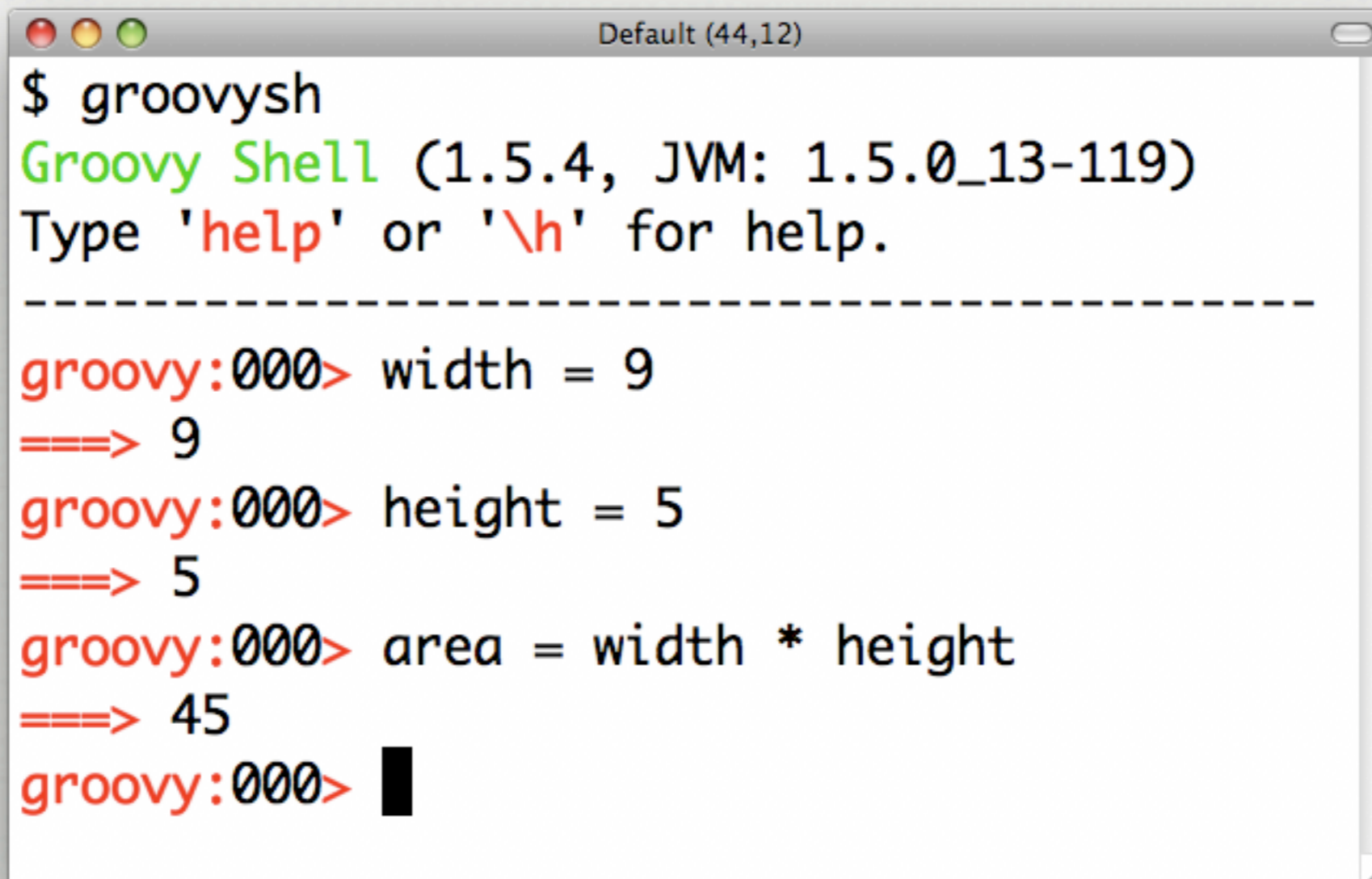
GIVE IT A SPIN

```
$ groovy -version  
Groovy Version: 1.5.4 JVM: 1.5.0_13-119
```

```
$ groovy -e "println 'Groovy Rocks.'"  
Groovy Rocks.
```

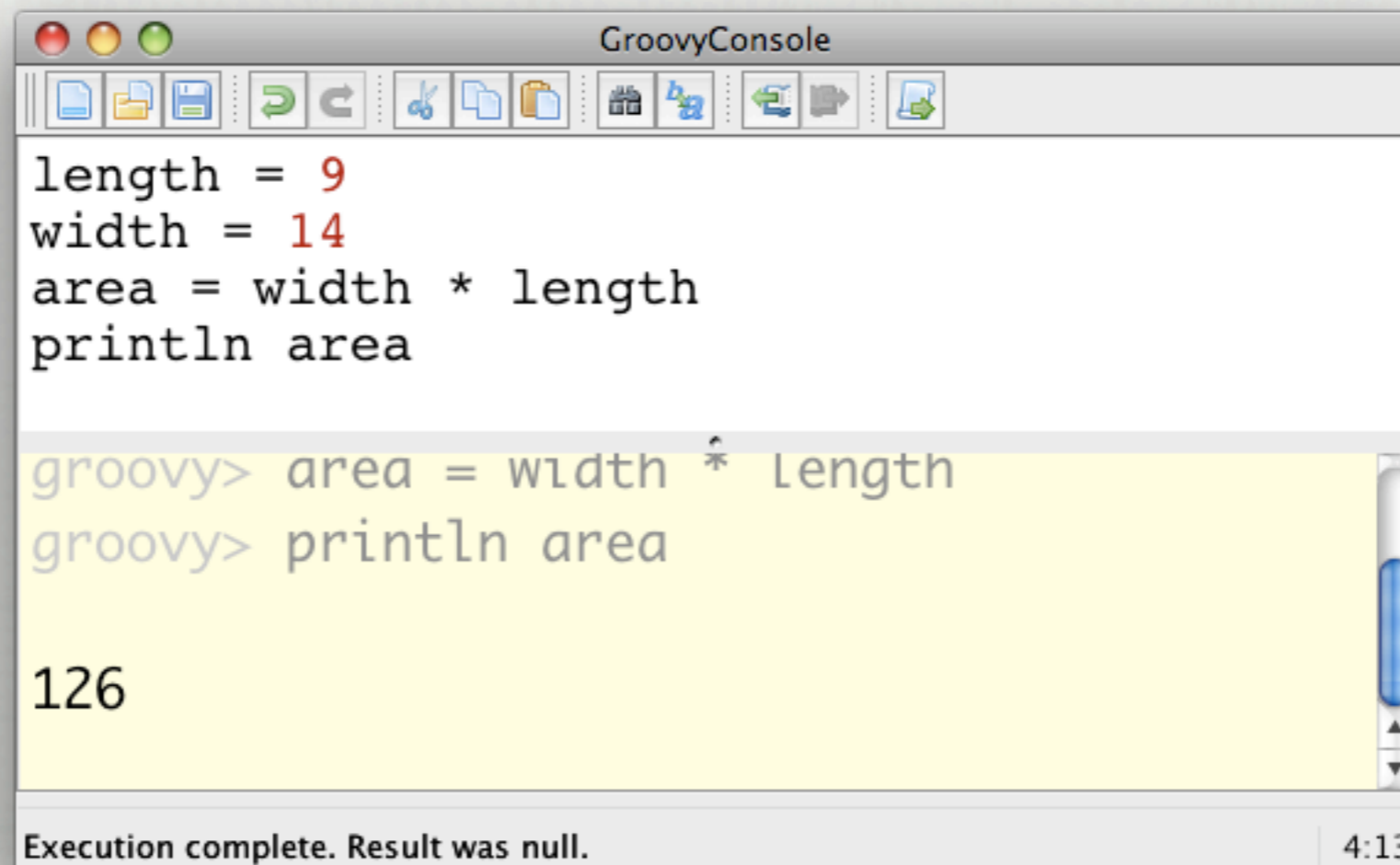
```
$ groovy -e "x=5; y=10; z=x*y; println z"  
50
```

GROOVYSH



```
Default (44,12)
$ groovysh
Groovy Shell (1.5.4, JVM: 1.5.0_13-119)
Type 'help' or '\h' for help.
-----
groovy:000> width = 9
==> 9
groovy:000> height = 5
==> 5
groovy:000> area = width * height
==> 45
groovy:000> █
```

GROOVYCONSOLE



The image shows a screenshot of the GroovyConsole application window. The window title is "GroovyConsole". The main area contains a Groovy script with the following code:

```
length = 9
width = 14
area = width * length
println area
```

Below the script, the console shows the execution of the code:

```
groovy> area = width * length
groovy> println area
```

The output of the execution is:

```
126
```

At the bottom of the window, a status bar indicates "Execution complete. Result was null." and the time "4:13".

GROOVY SCRIPTS

- SCRIPTS DO NOT REQUIRE A CLASS DEFINITION
- NO MAIN METHOD

```
// MyGroovyScript.groovy  
println 'This is an executable script!'
```

GROOVY CLASS

```
class GroovyPerson {  
  
    // dynamically typed property  
def age  
  
    // statically typed property  
    String name  
  
    def printName() {  
        println name  
    }  
  
    static void main(String[] args) {  
        def person = new GroovyPerson(age:7,  
                                       name:'Jake')  
        person.printName()  
    }  
}
```

MORE ON
GROOVY
PROPERTIES
LATER...

GROOVY STRINGS

- SINGLE QUOTED STRINGS ARE `JAVA.LANG.STRING`
- DOUBLE QUOTED STRINGS ARE "GSTRINGS"
- MAY CONTAIN EMBEDDED GROOVY CODE

```
aString = 'This is a String'  
answer = 42  
aGString = "The answer is ${answer}."
```

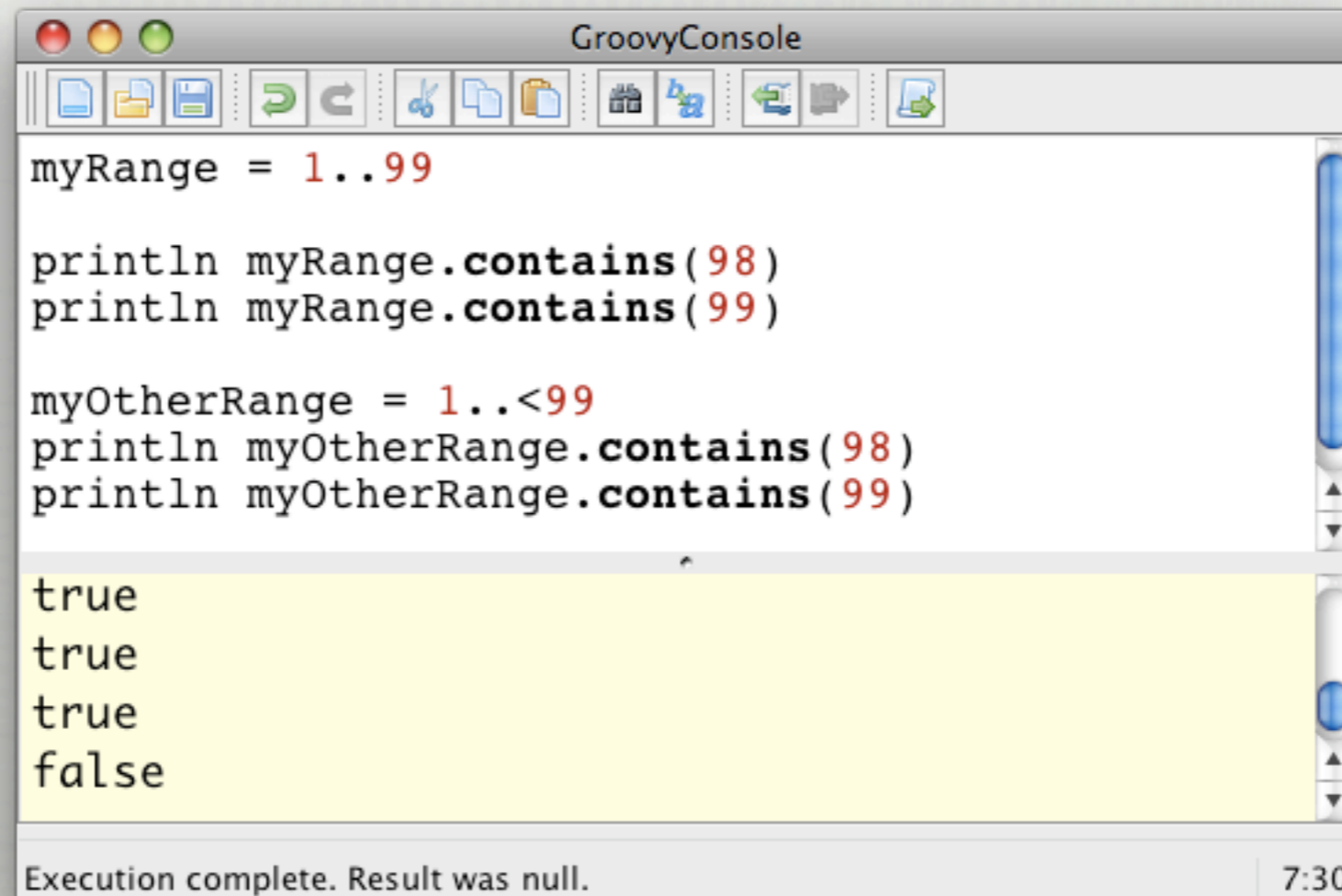
GROOVY STRINGS

- STRINGS MAY BE REFERENCED USING []

```
message = 'Groovy Is Cool'  
  
println message[0]           // G  
println message[-4]          // C  
println message[0..5]        // Groovy  
println message[-4..-1]      // Cool  
println message[-1..-4]      // looC
```

MORE ON
RANGES
COMING UP...

RANGES



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The main area contains the following code and output:

```
myRange = 1..99

println myRange.contains(98)
println myRange.contains(99)

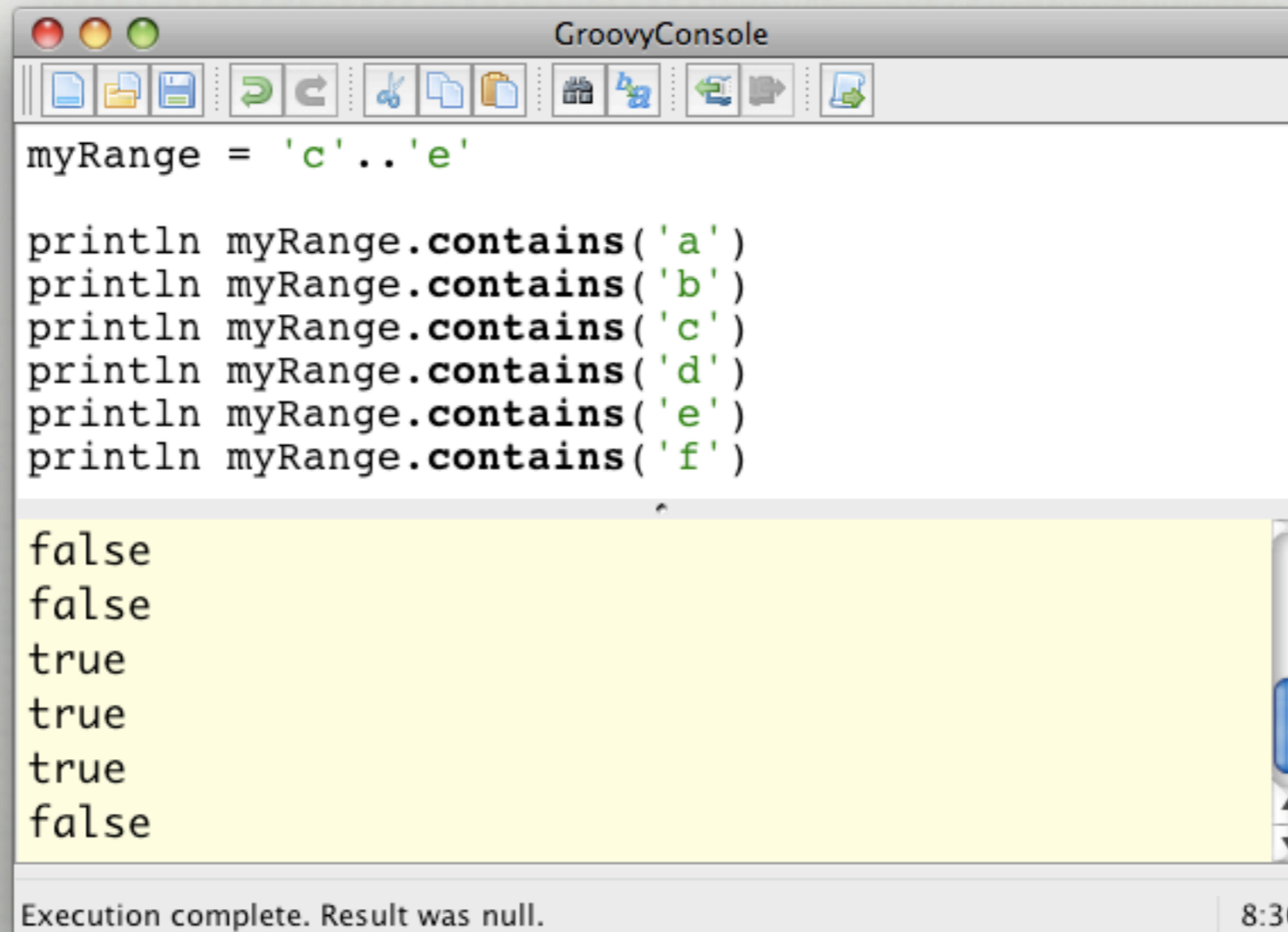
myOtherRange = 1..<99
println myOtherRange.contains(98)
println myOtherRange.contains(99)
```

The output of the code is displayed in a yellow-highlighted area:

```
true
true
true
false
```

At the bottom of the window, a status bar indicates "Execution complete. Result was null." and the time "7:30".

RANGES



The screenshot shows a GroovyConsole window with a toolbar at the top containing icons for file operations and execution. The main text area contains Groovy code defining a range and testing its contains() method for various characters. The output area below shows the results of these tests. The status bar at the bottom indicates that the execution is complete and the result was null, along with a timestamp of 8:30.

```
myRange = 'c'..'e'

println myRange.contains('a')
println myRange.contains('b')
println myRange.contains('c')
println myRange.contains('d')
println myRange.contains('e')
println myRange.contains('f')
```

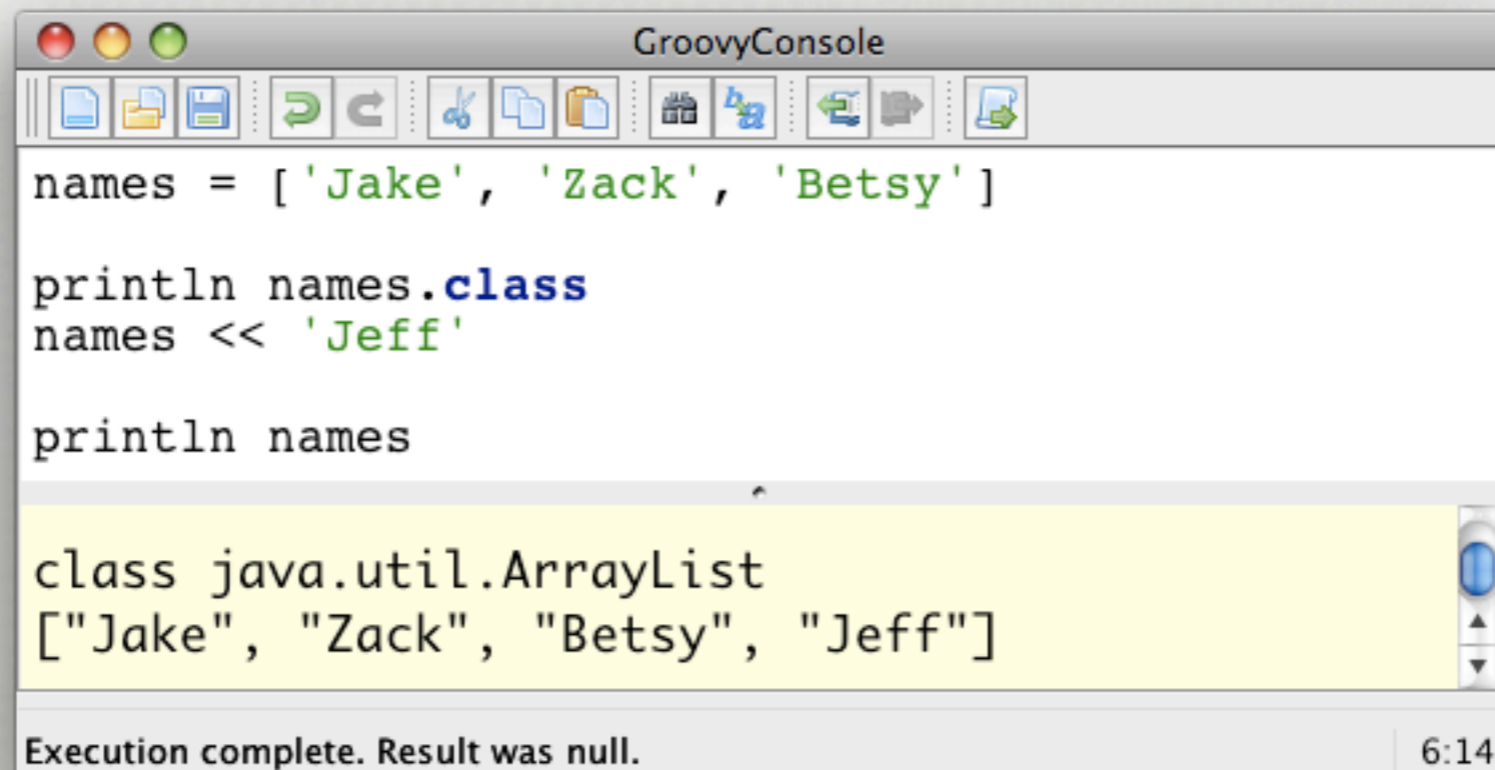
false
false
true
true
true
false

Execution complete. Result was null. 8:30

GROOVY COLLECTIONS

- GROOVY COLLECTIONS ARE STANDARD `JAVA.UTIL.COLLECTIONS`
- GROOVY ADDS MANY USEFUL METHODS TO EXISTING COLLECTIONS
- MANY COMMON TASKS ARE MUCH MORE SIMPLE IN GROOVY COMPARED TO JAVA

GROOVY LIST



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The code entered is:

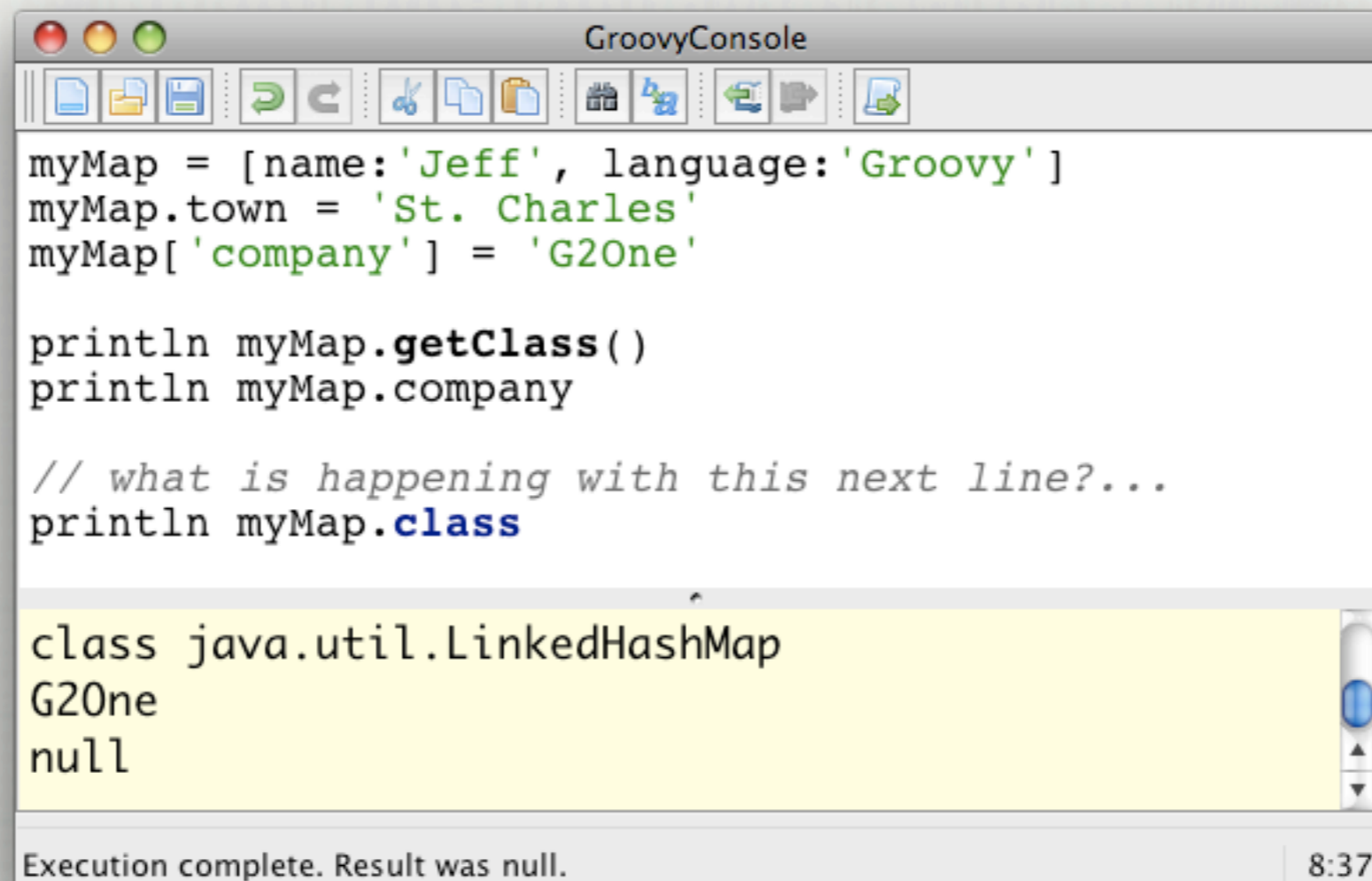
```
names = ['Jake', 'Zack', 'Betsy']  
  
println names.class  
names << 'Jeff'  
  
println names
```

The output, which is highlighted in yellow, is:

```
class java.util.ArrayList  
["Jake", "Zack", "Betsy", "Jeff"]
```

At the bottom of the window, it says "Execution complete. Result was null." and the time "6:14" is displayed in the bottom right corner.

GROOVY MAPS



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The main text area contains Groovy code. The first part defines a map and prints its class and company. The second part prints the class of the map object, which is highlighted in yellow. The status bar at the bottom indicates "Execution complete. Result was null." and the time "8:37".

```
myMap = [name:'Jeff', language:'Groovy']
myMap.town = 'St. Charles'
myMap['company'] = 'G2One'

println myMap.getClass()
println myMap.company

// what is happening with this next line?...
println myMap.class
```

```
class java.util.LinkedHashMap
G2One
null
```

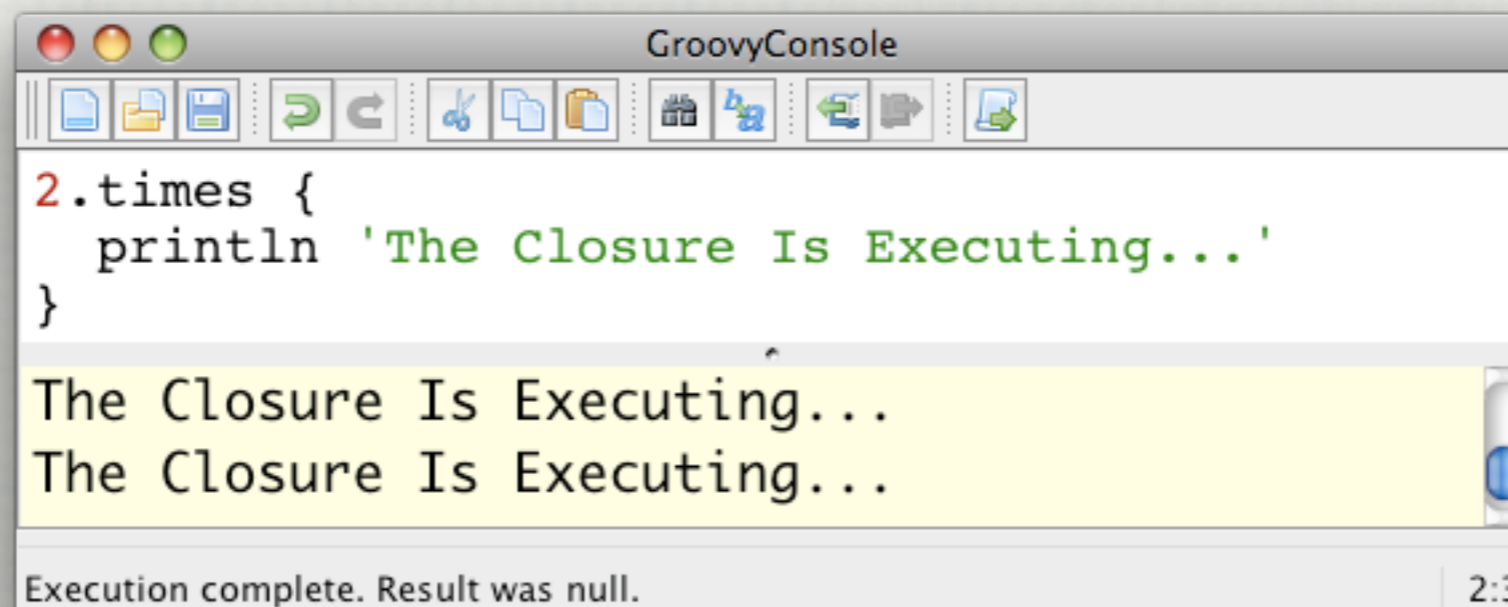
Execution complete. Result was null. 8:37

CLOSURES

- A BLOCK OF CODE
- MAY BE PASSED AS ARGUMENTS
- MAY ACCEPT PARAMETERS
- MAY RETURN A VALUE
- MUCH MORE POWERFUL THAN ANONYMOUS INNER CLASSES

CLOSURES

- GROOVY ADDS A 'TIMES' METHOD TO NUMBER
- THE 'TIMES' METHOD ACCEPTS A CLOSURE AS AN ARGUMENT

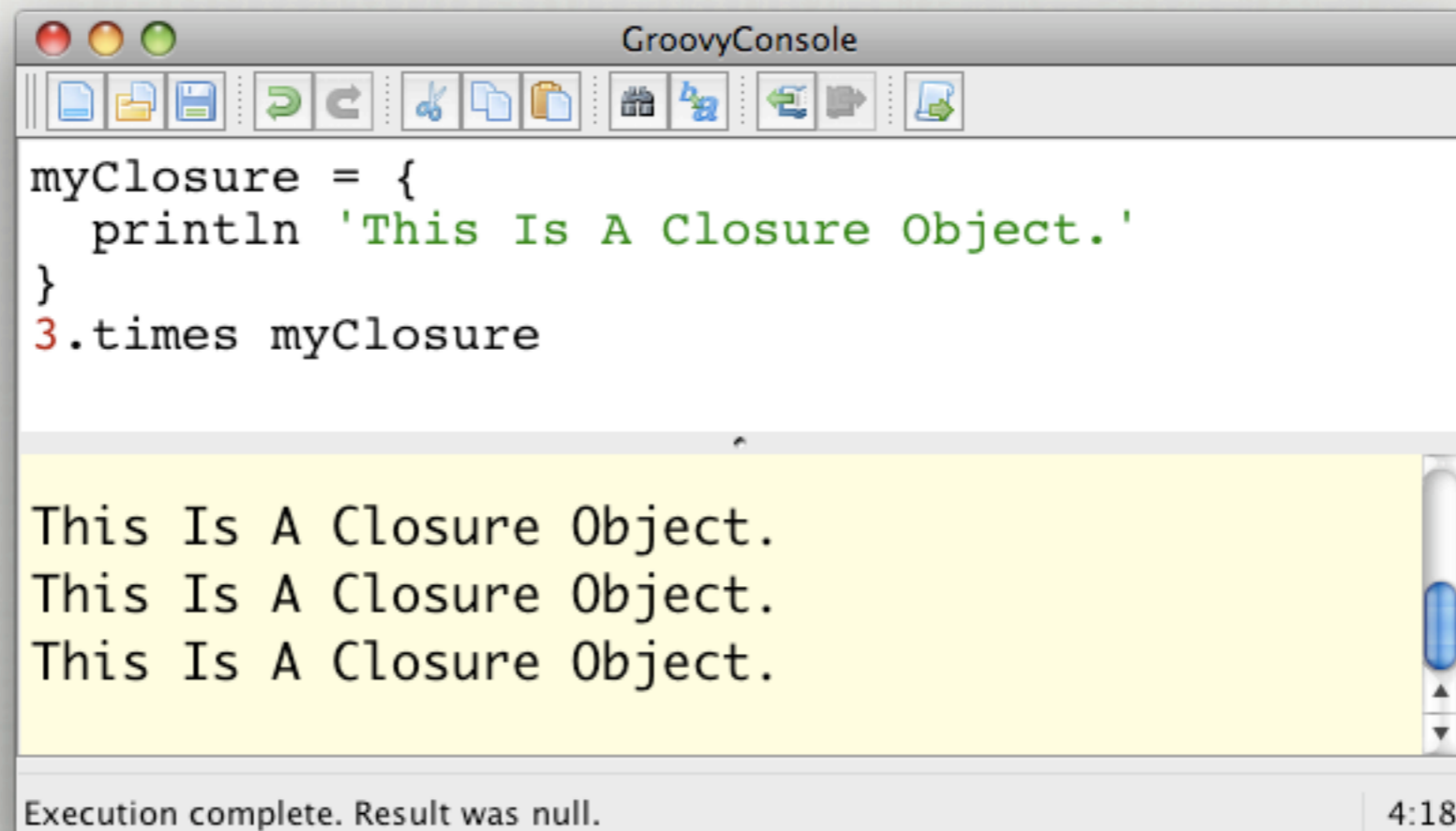


The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The code entered is:

```
2.times {  
  println 'The Closure Is Executing...'  
}
```

The output area shows two lines of text: "The Closure Is Executing..." and "The Closure Is Executing...". At the bottom, a status bar indicates "Execution complete. Result was null." and a timer shows "2:3".

CLOSURES



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The main area contains the following Groovy code:

```
myClosure = {  
    println 'This Is A Closure Object.'  
}  
3.times myClosure
```

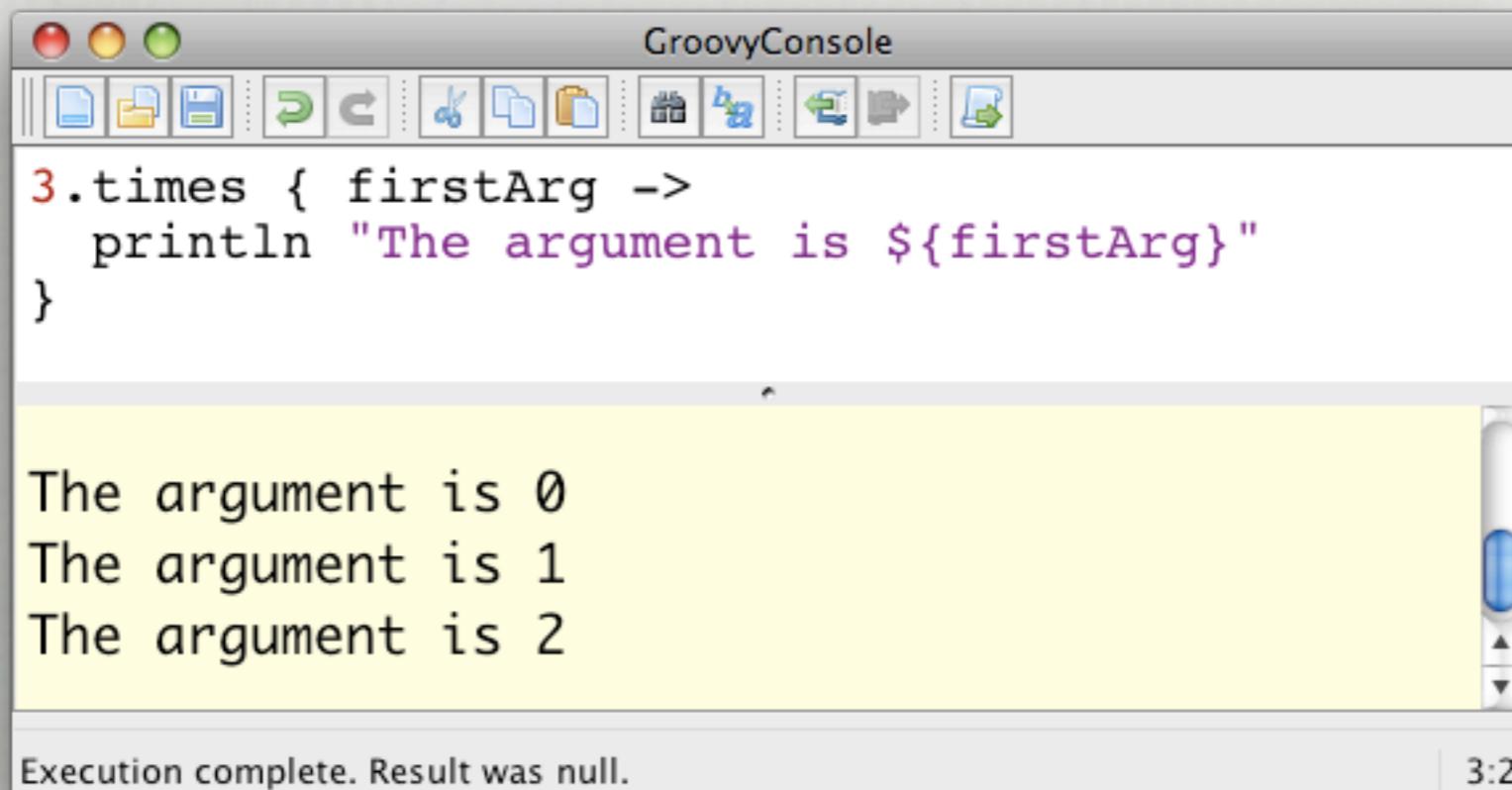
The output area, which is highlighted in yellow, shows the result of the execution:

```
This Is A Closure Object.  
This Is A Closure Object.  
This Is A Closure Object.
```

At the bottom of the window, a status bar indicates "Execution complete. Result was null." and a timer shows "4:18".

CLOSURES

- CLOSURES MAY DECLARE AN ARGUMENT LIST



```
GroovyConsole  
3.times { firstArg ->  
  println "The argument is ${firstArg}"  
}
```

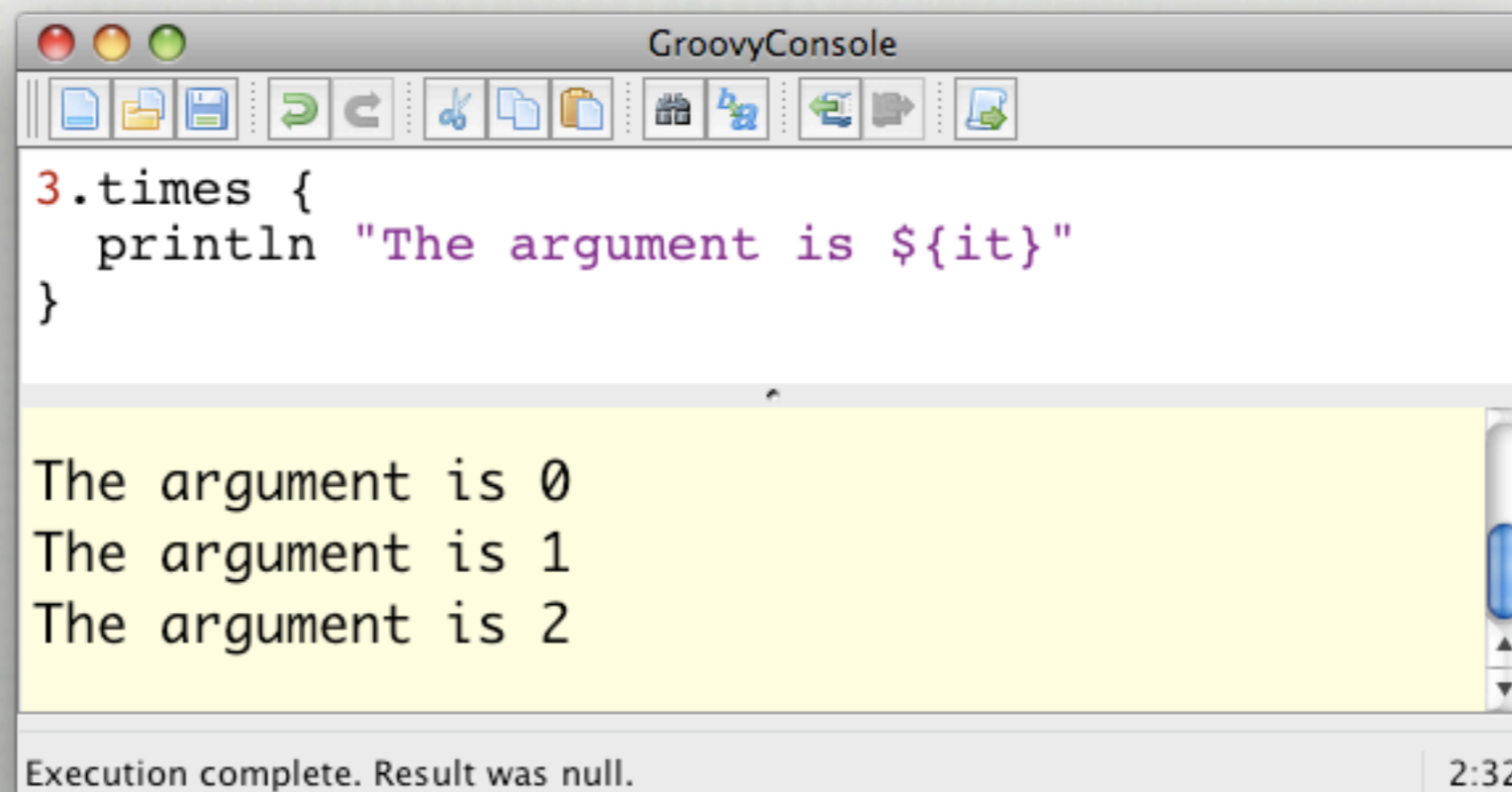
The argument is 0
The argument is 1
The argument is 2

Execution complete. Result was null. 3:2

THE TIMES METHOD
IS PASSING AN
ARGUMENT INTO
THE CLOSURE

CLOSURES

□ THE IMPLICIT 'IT' ARGUMENT



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The main area contains a Groovy script and its output. The script is:

```
3.times {  
    println "The argument is ${it}"  
}
```

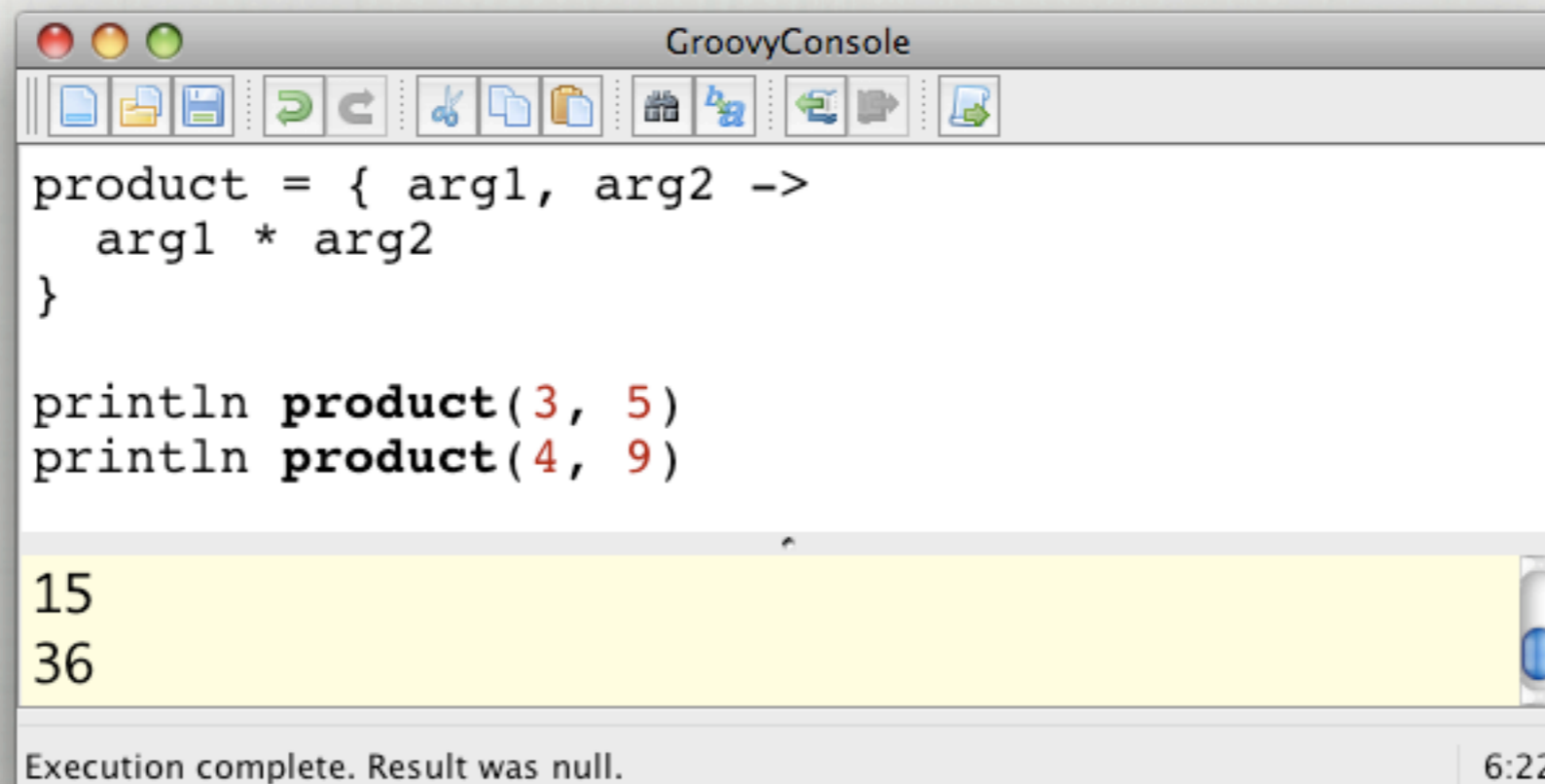
The output, displayed on a yellow background, is:

```
The argument is 0  
The argument is 1  
The argument is 2
```

At the bottom of the window, a status bar indicates "Execution complete. Result was null." and a timer shows "2:32".

CLOSURES

- CLOSURES MAY ACCEPT MULTIPLE ARGUMENTS



```
GroovyConsole
product = { arg1, arg2 ->
  arg1 * arg2
}

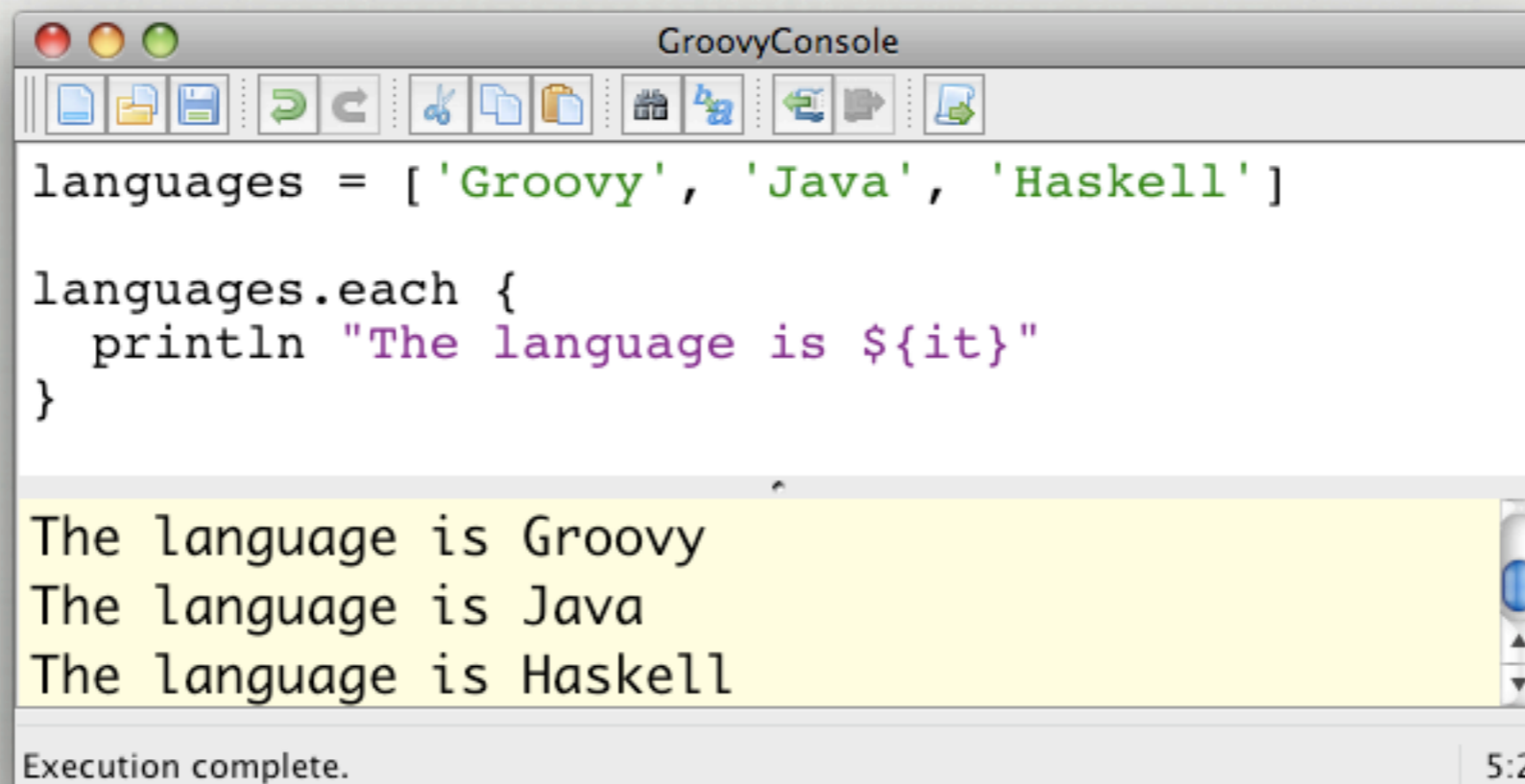
println product(3, 5)
println product(4, 9)

15
36

Execution complete. Result was null. 6:22
```

CLOSURES

- CLOSURES SIMPLIFY COLLECTION ITERATION



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The code in the editor is:

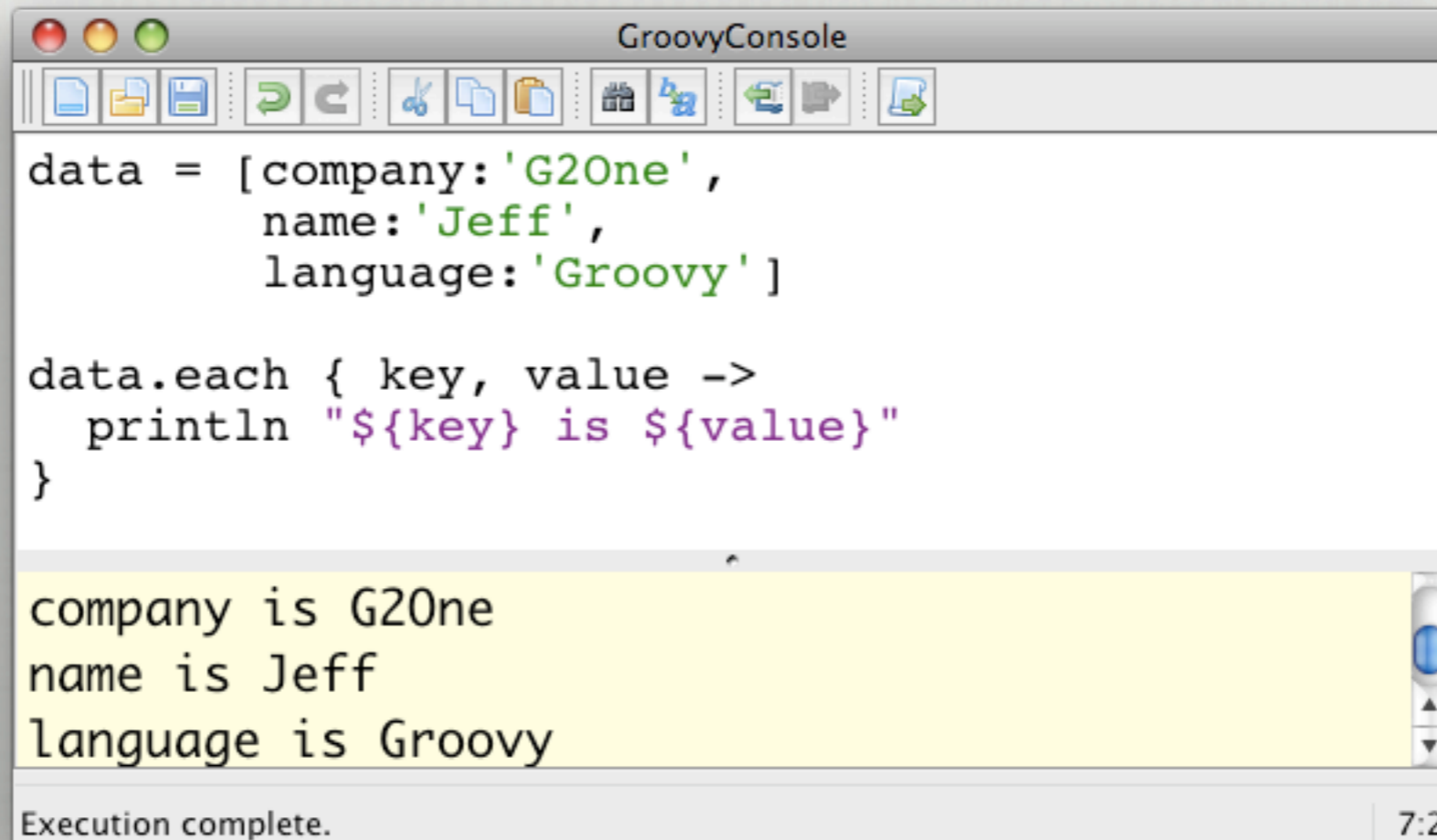
```
languages = [ 'Groovy', 'Java', 'Haskell' ]  
  
languages.each {  
    println "The language is ${it}"  
}
```

The output area shows the following text:

```
The language is Groovy  
The language is Java  
The language is Haskell
```

At the bottom of the window, it says "Execution complete." and "5:2".

CLOSURES



The screenshot shows a window titled "GroovyConsole" with a toolbar containing icons for file operations and execution. The main area contains Groovy code that defines a list and iterates over it. The output of the code is displayed in a yellow-highlighted area below the code. At the bottom of the window, it says "Execution complete." and "7:2".

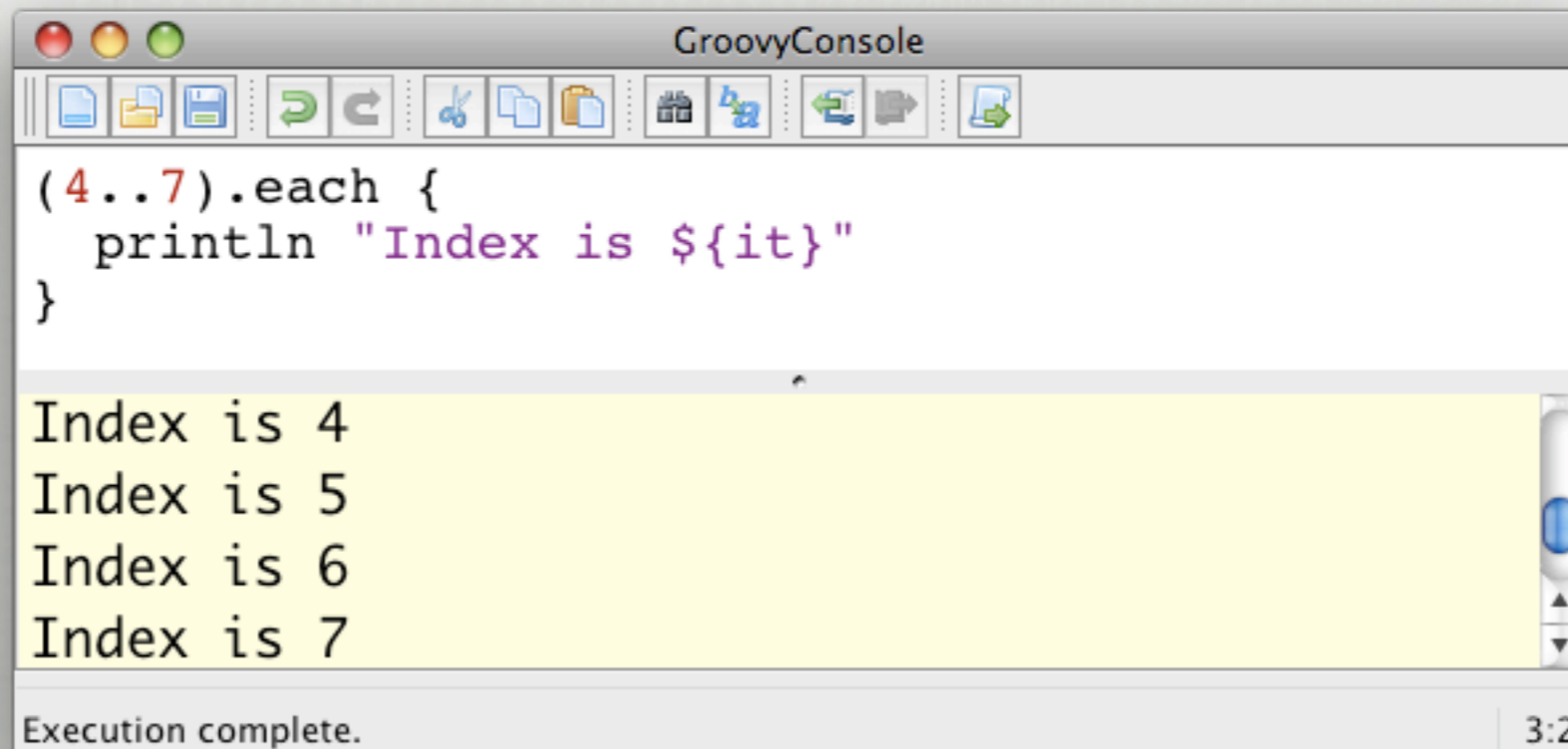
```
data = [company: 'G2One',
        name: 'Jeff',
        language: 'Groovy']

data.each { key, value ->
    println "${key} is ${value}"
}
```

```
company is G2One
name is Jeff
language is Groovy
```

Execution complete. 7:2

CLOSURES



The image shows a screenshot of a GroovyConsole window. The window title is "GroovyConsole". The toolbar contains icons for file operations (new, open, save, print), navigation (back, forward), and execution (run, stop). The code entered is:

```
(4..7).each {  
    println "Index is ${it}"  
}
```

The output of the code is displayed in a yellow-highlighted area:

```
Index is 4  
Index is 5  
Index is 6  
Index is 7
```

At the bottom of the window, it says "Execution complete." and "3:2".

GROOVY BEANS

- GROOVY BEANS / POJOS
- SIMILAR TO POJOS
 - ...BUT GROOVIER
- ELIMINATES BOILERPLATE CODE

POJO

```
public class Person {  
    private String firstName;  
    private String lastName;  
  
    public Person() {  
    }  
  
    public Person(String firstName, String lastName) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
}
```

POJO

```
public String getFirstName() {  
    return firstName;  
}  
  
public void setFirstName(String firstName) {  
    this.firstName = firstName;  
}  
  
public String getLastName() {  
    return lastName;  
}  
  
public void setLastName(String lastName) {  
    this.lastName = lastName;  
}  
}
```

POJO

- MODERN JAVA IDES GENERATE MOST OF THAT CODE
 - DEVELOPER DECLARES FIELDS
 - IDE GENERATES CONSTRUCTORS
 - IDE GENERATES GETTERS/SETTERS

IF THE IDE CAN GENERATE ALL OF THAT CODE,
WHY CAN'T THE COMPILER OR THE RUNTIME?

GROOVY BEANS

- GROOVY BEANS ELIMINATE ALL OF THE BOILERPLATE CODE
- NO NEED TO WRITE GETTERS/SETTERS
- SELDOM NEED TO WRITE CONSTRUCTORS

GROOVY BEANS

```
class BaseballTeam {  
    def cityName  
    def teamName  
}
```

```
myTeam = new BaseballTeam(teamName: 'Cardinals',  
                           cityName: 'St. Louis')  
  
println myTeam.teamName  
println myTeam.cityName
```

GROOVY BEANS

- PROPERTY ACCESS LOOKS LIKE FIELD ACCESS

```
myTeam = new BaseballTeam()  
  
// myTeam.setTeamName('Cardinals')  
myTeam.teamName = 'Cardinals'  
  
// myTeam.setCityName('St. Louis')  
myTeam.cityName = 'St. Louis'  
  
// println myTeam.getTeamName()  
println myTeam.teamName
```

GROOVY BEANS

```
class BaseballTeam {  
    def cityName  
    def teamName  
  
    def getDisplayName() {  
        "${cityName} ${teamName}"  
    }  
}
```

```
myTeam = new BaseballTeam()  
  
myTeam.teamName = 'Cardinals'  
myTeam.cityName = 'St. Louis'  
  
// println myTeam.getDisplayName()  
println myTeam.displayName
```

BUILDERS

- BUILDERS ARE A POWERFUL CONCEPT
- METAPROGRAMMING MAKES BUILDERS A SNAP IN GROOVY
- SEVERAL BUILDERS ARE BUNDLED WITH GROOVY
 - SWINGBUILDER, MARKUPBUILDER, ETC...
- YOU CAN WRITE YOUR OWN

MARKUPBUILDER

```
def xmlBuilder =
  new groovy.xml.MarkupBuilder()

xmlBuilder.teams {
  teams {
    team(name: 'Cardinals') {
      player('Albert Pujols')
      player('Ozzie Smith')
    }
    team(name: 'Rams') {
      player('Torry Holt')
    }
  }
}
```

```
<teams>
  <teams>
    <team name='Cardinals'>
      <player>Albert Pujols</player>
      <player>Ozzie Smith</player>
    </team>
    <team name='Rams'>
      <player>Torry Holt</player>
    </team>
  </teams>
</teams>
```

LINKS

- [HTTP://GROOVY.CODEHAUS.ORG/](http://groovy.codehaus.org/)
- [HTTP://GRAILS.ORG/](http://grails.org/)
- [HTTP://WWW.ABOUTGROOVY.COM/](http://www.aboutgroovy.com/)
- [HTTP://WWW.GROOVYBLOGS.ORG/](http://www.groovyblogs.org/)
- [HTTP://GROOVY.DZONE.COM/](http://groovy.dzone.com/)

QSA