PROGRAMMATIC EVALUATION OF APACHE GROOVY

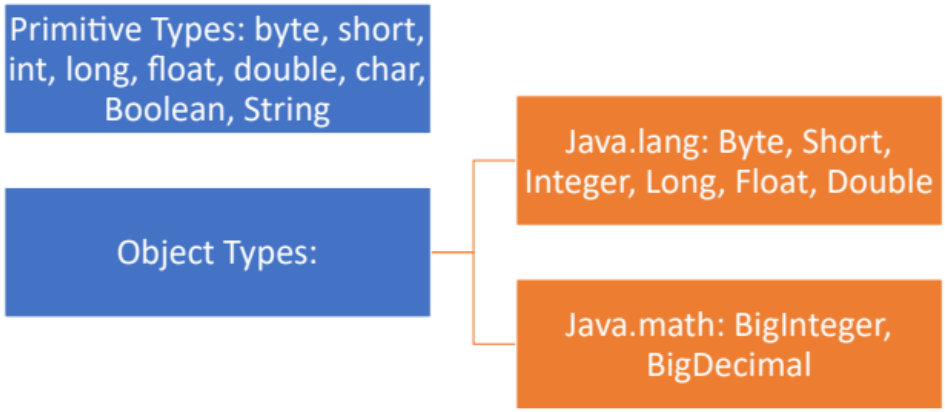
Ganesh Kumar Rajasekar - G37969806
Prudhvi Lakshmi Sesha Sai Cheedella - G22981341
Sai Nikhil Varada – G43353336
Yashwanth Raj Varadharajan - G47635180

Introduction

Names, Binding & Scopes

- Names start with letters, digits & underscore character.
- Uses both static & dynamic type binding
- All variables created are public by default
- Global variables – no keyword required, accessed throughout the script
- Local variables, block-scoped – use ‘def’ or use the apt. datatype keyword

Data Types


Primitive Types: byte, short, int, long, float, double, char, Boolean, String
Object Types:
Java.lang: Byte, Short, Integer, Long, Float, Double
Java.math: BigInteger, BigDecimal

Expressions and Assignment Statements

- To initialize a variable, we have to use the ”def” keyword.
- Single assignment

```
1 static void main(String[] args) {
2   def result = 10 + 5  // result will be 15
3   println(result)
4 }
```

- Multiple Assignment

```
1 static void main(String[] args)
2 { def (x, y) = [10, 20]
3 }
```

Loops

- For Loop

```
1 static void main(String[] args)
2 { for (int i = 0; i < 5; i++) {
3     println("Iteration $i")
4   }
5 }
```

- While Loop

```
1 static void main(String[] args)
2 { def count = 0
3   while (count < 5) {
4     println("Count: $count")
5     count++
6   }
7 }
```

- Do While

```
1 static void main(String[] args)
2 { def value = 0
3   do {
4     println("Value: $value")
5     value++
6   } while (value < 5)
7 }
```

Apace Groovy also supports for loops like for-each, and for-in like python.

Object-Oriented Programming

Apache Groovy fully supports Object-Oriented Programming (OOP) principles

- Class:

We have created a class called Student inside which we have two variables, student name and gwid number, default constructor and function to display the values of the variables

```groovy
class Student {
  String student_name
  int student_gwid

  Student(String student_name, int student_gwid){
    this.student_name = student_name
    this.student_gwid = student_gwid
  }

  void studentInfo( ){
    println("Student Name: ${student_name}")
    println("Student GWID: ${student_gwid}")
  }
}
```

Object-Oriented Programming

- Object

We now create two objects, student1 and student2. We have assigned two different values to the objects and use the studentInfo() function inside the class and print the values.

```
def student1 = new Student("Yashwanth", 47635180)
def student2 = new Student("Sai Nikhil", 47635170)

println("Student 1 Information:")
student1.studentInfo()

println("Student 2 Information:")
student2.studentInfo()
```

Concurrency

Apache groovy is a versatile language that is capable of running multiple processes simultaneously.

In the provided example, we have developed a program called "myTask" with two threads, named "thread1" and "thread2." We have utilized the "sleep" function to better visualize the working of program.

When we execute the program, we can observe that both "thread1" and "thread2" run at the same time.

```groovy
def myTask = new Runnable() {
  void run() {
    for (int i = 1; i <= 5; i++) {
      println("Thread ${Thread.currentThread().id}: Count $i")
      sleep(500)
    }
  }
}

def thread1 = new Thread(myTask)
def thread2 = new Thread(myTask)

thread1.start()
thread2.start()

thread1.join()
thread2.join()

println("All threads have finished.")
```

GPARS

- Dedicated library for the era of multi-core processors and concurrent programming.
- Uses best concepts from emerging languages:
  - Actors library in Scala
  - SafeVariable class inspired by Agents in Clojure
- Covers three main areas:
  - Starting and stopping concurrent tasks
  - Coordinating concurrent tasks
  - Controlling access to shared mutable state

Functions

- **Parallel collections** with *fork/join* and *map/filter/reduc*e operations are concepts that hide the work of starting and stopping concurrent tasks from the programmer and coordinate these tasks in a *predefined manner*
- **Actors** are a frame in which asynchronous ops can be performed and be controlled explicitly.
- **Dataflow** variables, operators, and streams coordinate concurrent tasks *implicitly* such that downstream data consumers automatically wait for data providers.
- If your tasks need to access a shared mutable state, you can ***delegate*** the coordination of concurrent state changes to an ***agent***.

Example

```groovy
import static groovyx.gpars.actor.Actors.*
//not necessary, just showing that a single-threaded pool can still handle multiple actors
defaultPooledActorGroup.resize 1
final def console = actor {
    loop {
        react {
            println 'Result: ' + it
        }
    }
}
final def calculator = actor {
    react {a ->
        react {b ->
            console.send(a + b)
        }
    }
}
calculator.send 2
calculator.send 3
calculator.join()
```

- The event-driven actor receives two numeric messages, generates a sum and sends it
- Loop() – used to iterate through incoming messages.
- React() - to receive incoming messages
- reply() – to send messages
- All are passed closure method body parameters

Event Handling

1. Event handling in Apache Groovy is similar to other programming languages, but it has some additional features and syntax that make it easier to implement.
2. Groovy also provides an event bus that can be used to communicate between different parts of an application using events.
3. Event handling is a powerful way to decouple different parts of an application and make it more modular and reusable. Groovy provides a number of features that make it easy to implement event handling in your applications.

```
try {
  // code that might throw an exception
} catch (SomeException e) {
  // handle the exception
}
```

```
try {
  // code that might throw an exception
} catch (SomeException | AnotherException e) {
  // handle the exception
}
```

Exception Handling

- Exception handling in Apache Groovy is the same as in Java. We use the try-catch block to catch an exception and handle it.
- We can also use the multi-catch block to catch multiple exceptions at the same time.

```
class MyClass {
  def buttonClickListener = {
    println "Button clicked!"
  }

  def init() {
    // Register the listener with the button
    button.addActionListener(buttonClickListener)
  }
}

def myClass = new MyClass()
myClass.init()

// Click the button
button.click()
```

Functional Programming

- Apache Groovy is a dynamic programming language for the Java platform that supports multiple programming paradigms, including functional programming.
- Groovy provides a number of features that make it well-suited for functional programming, including:
  - Closures
  - Higher order functions
  - Currying
  - Function composition
  - Pure functions
  - Immutability
  - Recursion
  - Tail call optimisation

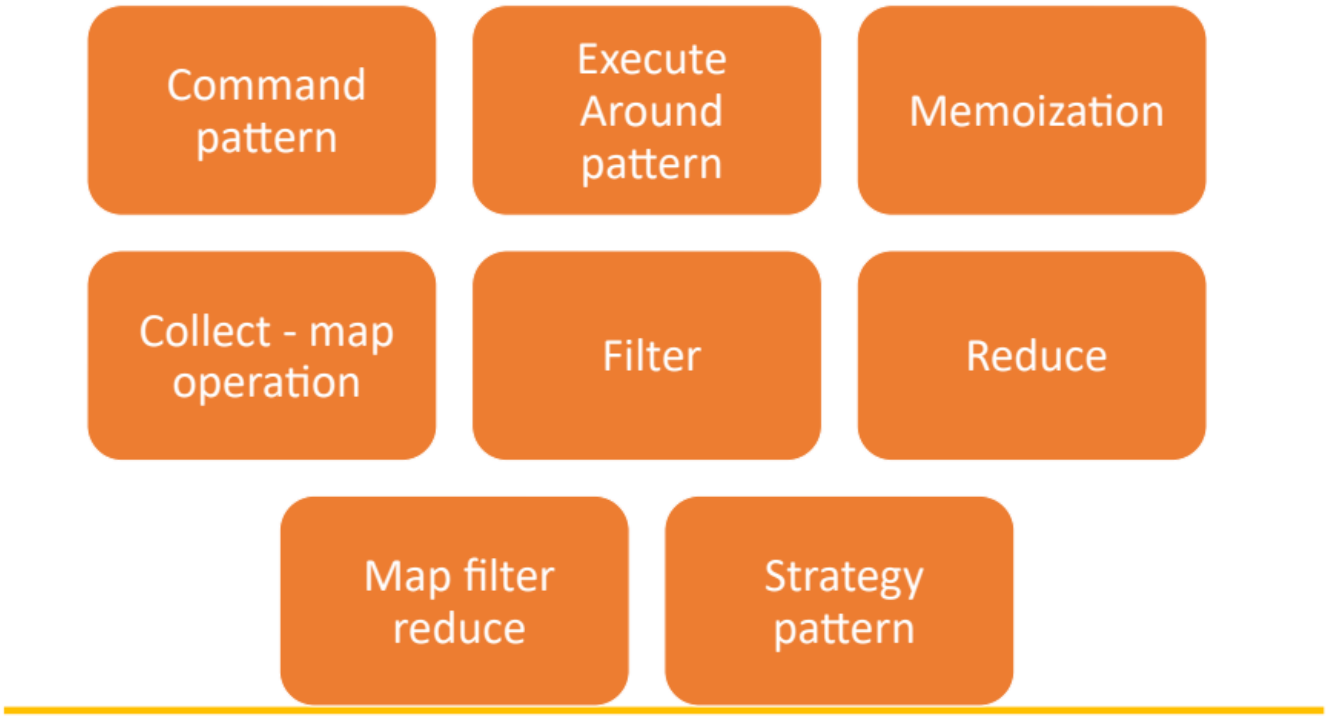
```
def map(f, xs) {
    result = []
    for (x in xs) {
        result.append(f(x))
    }
    return result
}

def filter(p, xs) {
    result = []
    for (x in xs) {
        if (p(x)) {
            result.append(x)
        }
    }
    return result
}

def reduce(f, xs, initial) {
    result = initial
    for (x in xs) {
        result = f(result, x)
    }
    return result
}

def main() {
    print(map(lambda x: x * 2, [1, 2, 3, 4, 5]))
    print(filter(lambda x: x % 2 == 0, [1, 2, 3, 4, 5]))
    print(reduce(lambda x, y: x + y, [1, 2, 3, 4, 5], 0))
}

if (__name__ == '__main__') {
    main()
}
```


Command pattern
Execute Around pattern
Memoization
Collect - map operation
Filter
Reduce
Map filter reduce
Strategy pattern

Fitness Consultation App

A WEB APPLICATION WHERE USERS CAN SIGN UP TO SEEK CONSULTATION FROM FITNESS TRAINERS.

TWO PARTIES – TRAINER & CLIENT

Features

- Both can sign in/sign up
- Clients can view and search trainers based on the services provided by them
- Clients can receive consultation by *scheduling appointments for in-person meetings*, watching *videos uploaded* by the trainer.
- All of the above within a transaction called '***consultation***' between trainer & client.
- Clients will be able to give a rating & feedback to their trainers, average of which will be the Trainer's overall rating

Tech Stack

- Frontend: React
- Backend: Grails Framework (Full-stack framework for Apache groovy)
- Database: MySQL

THANK YOU