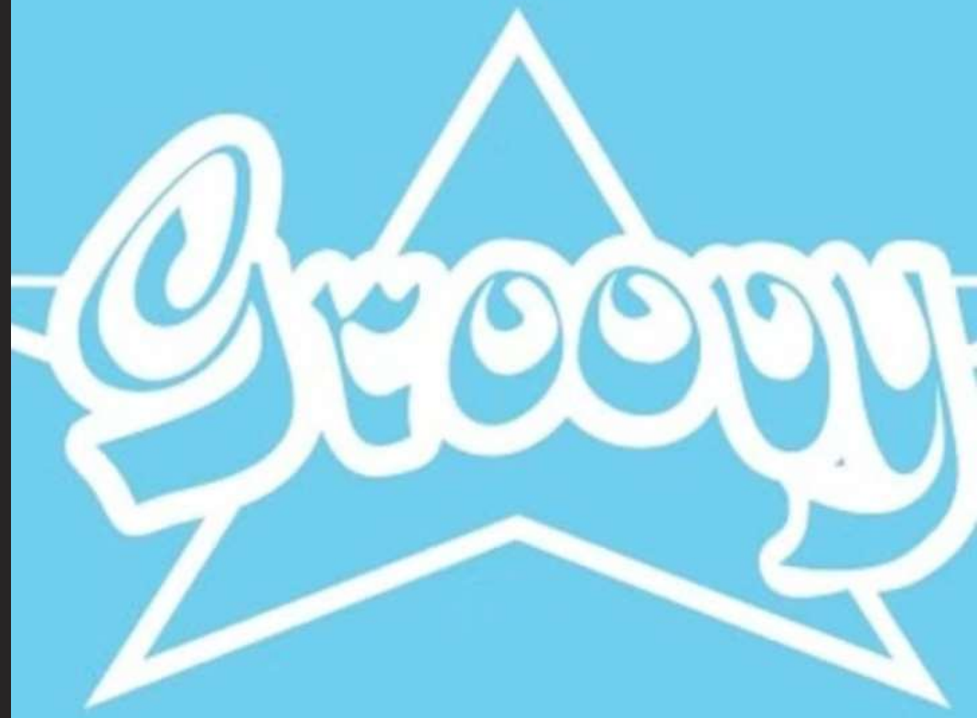
Conceitos Básicos de Groovy

Groovy é uma linguagem de programação dinâmica que roda na JVM. Ela combina a sintaxe familiar do Java com recursos dinâmicos, como tipagem dinâmica e metaprogramação. Groovy foi influenciada por linguagens como Python e Ruby.



Java Vs. Groovy

```java
public class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Olá, Mundo!");
    }
}
```

Java

```groovy
class OlaMundo {
    static void main(String[] args) {
        println("Olá, Mundo!")
    }
}
```

Groovy

Palavras-chave e Palavras Reservadas

1

Palavras-chave

Palavras com significados especiais na sintaxe, como def, return, if, for, etc.

2

Palavras Reservadas

Palavras com significados especiais em contextos específicos, como null, true, false, etc.

Exemplo Palavras Chaves e Reservadas

```groovy
class Pessoa {
  def nome = null
  def idade = null

  Pessoa(String nome, int idade) {
    this.nome = nome
    this.idade = idade
  }

  String apresentar() {
    "Olá, meu nome é $nome e minha idade é $idade anos."
  }
}

def pessoa = new Pessoa("Klaudio", 20)
println(pessoa.apresentar()) //Output: Olá, meu nome é Klaudio e minha idade é 20 anos.
```

Vinculações de Tipos

Tipagem Dinâmica

Groovy é dinamicamente tipada, os tipos são determinados em tempo de execução.

Fracamente Tipada

Groovy realiza conversões implícitas entre tipos, o que pode levar a comportamentos inesperados.

Exemplo de Vinculações de Tipos

```
1 static void main(String[] args) {
2   def mensagem = "Olá, Mundo!"
3   println(mensagem) // Output: Olá, Mundo!
4
5   mensagem = 42
6   println(mensagem) // Output: 42
7
8   mensagem = [1, 2, 3]
9   println(mensagem) // Output: [1, 2, 3]
10
11 }
```

Escopos de Variáveis

1

Variáveis Globais

Acessíveis de qualquer lugar do programa.

2

Variáveis Locais

Acessíveis apenas dentro do bloco em que foram declaradas.

3

Escopo Dinâmico

Variáveis são resolvidas em tempo de execução (padrão).

4

Escopo Estático

Variáveis são resolvidas em tempo de compilação (usando closures).

Exemplo Escopos de Variáveis

```groovy
// Exemplo variável Global
def variavelGlobal = 10

void metodo1() {
  println("Variável global dentro do método 1: $variavelGlobal")
}

void metodo2() {
  println("Variável global dentro do método 2: $variavelGlobal")
}

metodo1()
metodo2()

// Exemplo variável Local
void metodo() {
  def variavelLocal = 20
  println("Variável local dentro do método: $variavelLocal")
}

metodo()
```

```groovy
// Escopo estático
class Exemplo {
  static int variavelEstatica = 30

  void metodo1() {
    println("Variável estática dentro do método: $variavelEstatica")
  }
}

def objeto = new Exemplo()
objeto.metodo1()

//Escopo dinâmico
def metodo2() {
  def variavelLocal = 40
  println("Variável local dentro do método: $variavelLocal")
}

metodo2()
```

Tipos de Dados

Tipos Primitivos

boolean, byte, short, int, long, float, double.

Tipos de Objetos

String, List, Map, Set, Date, File.

Tipos Especiais

Closure, Range, Pattern.

Exemplos Tipos de Dados

```groovy
def booleano = true
def inteiro = 10
def flutuante = 3.14
def caractere = 'a'

println(booleano.getClass())
// Output: class java.lang.Boolean
println(inteiro.getClass())
// Output: class java.lang.Integer
println(flutuante.getClass())
// Output: class java.lang.Double
println(caractere.getClass())
// Output: class java.lang.String
```

Tipos Primitivos

```groovy
def texto = "Olá, mundo!"
def lista = [1, 2, 3, 4, 5]
def mapa = [nome: "Alice", idade: 30]
def arquivo = new File("arquivo.txt")

println(texto.getClass())
// Output: class java.lang.String
println(lista.getClass())
// Output: class java.util.ArrayList
println(mapa.getClass())
// Output: class java.util.LinkedHashMap
println(arquivo.getClass())
// Output: class java.io.File
```

Tipos de Objetos

```groovy
def nulo = null
def faixa = 1..5
def bloco = { println("Isso é um bloco de código!") }

println(nulo.getClass())
// Output: class java.lang.NullPointerException
println(faixa.getClass())
// Output: class org.codehaus.groovy.runtime.typehandling.RangeInfo
bloco()
// Output: Isso é um bloco de código!
```

Tipos Especiais

Expressões

1

Literais

Representações diretas de valores de dados, como números, strings, listas, etc.

2

Agregados

Coleções de dados, como listas, conjuntos e mapas.

3

Chamadas de Função

Execução de código encapsulado em funções ou métodos.

4

Condicionais

Condicionais Instruções if, else-if, else e operador ternário.

5

Expressões Iterativas

Loops for, while e do-while.

Exemplo Expressões

```
// Tipo inteiro
def inteiro = 123
// Tipo ponto flutuante
def flutuante = 3.14
// Tipo string
def texto = "Exemplo de texto"
// Tipo lista
def lista = [1, 2, 3]
// Tipo mapa
def mapa = [nome: "Alice", idade: 30]
```

Literais

```
// Lista de números
def numeros = [1, 2, 3, 4, 5]
// Soma dos números
def soma = numeros.sum()
// Quadrados dos números
def quadrados = numeros.collect { it * it }
```

Agregados

```
def saudacao(nome) {
  return "Olá, $nome!"
}
def mensagem = saudacao("Rodrigo")
println(mensagem)
// Output: Olá, Rodrigo!
```

Chamadas de Função

```
def numero = 0 // Número para verificação
// Expressões Iterativas
for (numero; numero <= 10; numero++) {
  numero++
}
// Expressão condicional
def mensagem = (numero > 0) ? "Positivo" : "Negativo"
println(mensagem) // Output: Positivo
```

Condicionais e Expressões Iterativas

Comandos

Atribuição

Atribuição de valores a variáveis (simples, múltipla ou simultânea).

Chamada de Procedimento

Chamada de função ou método que realiza uma tarefa específica.

Comando Condicional

Executa um bloco de código se uma condição for verdadeira (if, else).

Comando Iterativo

Executa um bloco de código repetidamente (while, for).

Exemplo Comandos

```groovy
def soma(int a, int b) {
  return a + b
}

def resultado = soma(5, 3)
if (resultado > 0) {
  println("A soma é: $resultado")
  // Output: 8
} else {
  for (def i = 0; i < 10; i++) {
    resultado++;
  }
}
```

Parametrização de abstrações

Passagem por Cópia

O valor do parâmetro é copiado para a função, de modo que as alterações no parâmetro dentro da função ou procedimento não afetam a variável original.

```
1 def incrementarPorCopia(int valor) {
2     valor++
3     println("Dentro da função: $valor")
4 }
5
6 //Chamando a função com um valor
7 def numero = 5
8 incrementarPorCopia(numero)
9 println("Fora da função: $numero")
10 //Output: Fora da função: 5
```

Passagem por Referência

A referência ao objeto é passada para a função, de modo que as alterações no parâmetro dentro da função ou procedimento afetam a variável original.

```
1 def incrementarPorReferencia(int[] valores) {
2     valores[0]++
3     println("Dentro da função: ${valores[0]}")
4 }
5
6 // Chamando a função com um array
7 def numeros = [5]
8 incrementarPorReferencia(numeros)
9 println("Fora da função: ${numeros[0]}")
10 //Output: Fora da função: 6
```

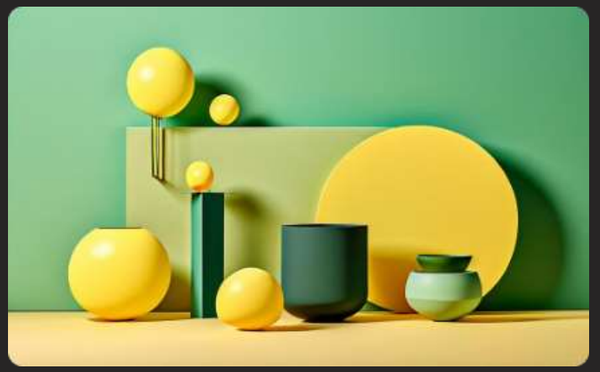
Unidades de Programa

Pacotes

Diretórios que contêm classes e recursos relacionados.

Tipos Abstratos de Dados

Representados por classes que encapsulam dados e comportamentos.



Objetos e Classes

Objetos são instâncias de classes que definem sua estrutura e comportamento.

Exemplos Unidades de Programa

```groovy
class Pessoa {
    def nome
    def idade

    Pessoa(nome, idade) {
        this.nome = nome
        this.idade = idade
    }

    def saudacao() {
        println("Olá, meu nome é ${nome} e eu tenho ${idade} anos.")
    }
}

def pessoa = new Pessoa("João", 24)
pessoa.saudacao()
//Output: Olá, meu nome é João e eu tenho 24 anos.
```

Sequenciadores

Desvios Incondicionais	break, continue
Escapes	return, throw
Exceções	try-catch

```
1 def dividir(a, b) {
2     try {
3         return a / b
4     } catch (ArithmeticException ex) {
5         println("Erro: Divisão por zero")
6     }
7 }
8
9 def resultado = dividir(10, 0)
10 println(resultado)
```