Apostila de Exercícios Groovy – F1

Desenvolvedor Groovy

Exercício 1

```
package aula

class Exercicio1 {
  static void main (args) {
    // Exer1
    println "Ola mundo groovy"

    // Exer2
    // interpolação de string:
    // cancatena automaticamente - $variavel ou $objeto.atributo
    String nome = "Fernando"
    int idade = 36

    String frase = "O $nome tem $idade anos."
    println frase
  }
}
```

Exercício 1.2

```
class Exercicios {

  @Test
  void exercicio1ponto2() {
    int a = 10
    println a.class // tem atributos e métodos.
    println a.toString()
    println 12l.class.name

    // muda 2 coisas do java:
    // 1 - coloca "g" no literal para virar BigInteger.
    println 11g.class.name

    // 2 - literal flutuante é considerado BigDecimal por padrão.
    BigDecimal valor = 200.50
    println valor
  }
```

Exercício 2

```
package classes

class Cliente {
  String nome
  Date data
  Integer somar(Integer v1, Integer v2) {
    v1 + v2
  }
}

  @Test
  void exercicio2() {
    Cliente c = new Cliente()
    c.setNome "Fer" // sem parenteses.
    c.setData new Date()
    println c.somar (10, 10)
    println c.getNome() + " "+ c.getData()
  }
```

Exercício 3

```
@Test
void exercicio3() {
  // veja que não fizemos construtor na classe Cliente
  Cliente c = new Cliente()
  println c.getNome() + " - " + c.getData()

  c = new Cliente(nome: "fernando")
  println c.getNome() + " - " + c.getData()

  c = new Cliente(data: new Date())
  println c.getNome() + " - " + c.getData()

  c = new Cliente(nome: "fernando", data: new Date())
  println c.getNome() + " - " + c.getData()
}
```

Exercício 4

```
@Test
void exercicio4() {
  Cliente c = new Cliente(nome: "fernando", data: new Date())
  println c["nome"]
  c["nome"] = "marcão"
  println c["nome"]
}
```

Exercício 5

```
package classes;

// 3 motivos para fazer essa classe em java:
// 1. Gerar uma classe em java e ver se eles gostam de voltar a digitar o que o  groovy faz.
// 2. Provar que funciona o mix de java e grovvy.
// 3. Usar o recurso de Direct field access operator.
public class Produto {

  private String nome;
  private double valor;

  public Produto(String nome, double valor) {
    this.nome = nome;
    this.valor = valor;
  }
  public String getNome() {
    System.out.println("getNome");
    return nome;
  }
  public void setNome(String nome) {
    System.out.println("setNome");
    this.nome = nome;
  }
  public double getValor() {
    System.out.println("getValor");
    return valor;
  }
  public void setValor(double valor) {
    System.out.println("setValor");
    this.valor = valor;
  }
}
```

```
@Test
void exercicio5() {
  Produto p = new Produto("CD", 12.00)
  println p.nome + "-" + p.valor
  p.nome = "CD Calcinha Preta"
  p.valor = 15.00
  println p.nome + "-" + p.valor
}
```

Exercício 6

```
 @Test
 void exercicio6() {
   // import classes.Cliente as Xu
   Xu c = new Xu(nome: "fernando", data: new Date())
   println  c.getNome() + " - " + c.getData()

   //import static javax.swing.JFrame.EXIT_ON_CLOSE as ex
   println ex
 }
```

Exercício 7

```
package classes

class Venda {
   double vender(double valor, int taxa = 10) {
      double rs = valor * taxa / 100
      rs // só para lembrar do return como ultima linha.
   }
}
```

```
 @Test
 void exercicio7() {
   Venda v = new Venda()
   println  v.vender(100)
   println  v.vender(100, 15)
 }
```

Exercício 8

```
package classes

class Somar {
  double somar(double[] valores) {
    double rs = 0;
    for (double v: valores) {
      rs += v;
    }
    rs
  }
}

 @Test
 void exercicio8() {
    Somar soma = new Somar()
    println soma.somar(10)
    println soma.somar(10, 10)
    println soma.somar(10, 10, 10)
    // depurar dentro da classe somar.
 }
```

Exercício 9

```
@Test
void exercicio9() {
   Cliente c = null
   c?.nome = "Fernando"
   println c?.getNome()
   // em java geraria null point exception

   Cliente c2 = new Cliente()
   c2?.nome = "Fernando"
   println c2?.getNome()
}
```

Exercício 10

```
@Test
void exercicio10() {
  List<String> colecao = new ArrayList<>()
  colecao.add("fernando")
  colecao.add(null)
  colecao.add("anny")
  colecao.add("rita")
  println colecao
  colecao = colecao*.toUpperCase()
  println colecao
  colecao = colecao*.replace("A", "@")
  println colecao
}
```

Exercício 11

```
@Test
void exercicio11() {
  // Construtor lança 2 exception, veja javadoc.
  URL url = new URL("http://www.google.com.br")
  println url
  // depois faça dar errado para ver o lancamento da exception
  //URL errado = new URL("hbla blalbal")
}
```

Exercício 12

```
@Test
void exercicio12() {
  String nome = null
  if (nome) {
    println true
  } else{
    println false
  }
  nome = "Fer"
  if (nome) {
    println true
  } else{
    println false
  }
  int valor = 0
  if (valor) {
    println true
  } else{
    println false
  }
  valor = 1
  if (valor) {
    println true
  } else{
    println false
  }
  List<String> colecao = new ArrayList(0)
  if (colecao) {
    println true
  } else{
    println false
  }
  colecao.add("fer")
  if (colecao) {
    println true
  } else{
    println false
  }
  // nenhum desses funcionam em java.
}
```

Exercício 13

```
package classes

class Nota {
  Integer itens
  Double valor

  Nota plus(Nota nota) {
    println "plus"
    Nota novo = new Nota();
    novo.itens = this.itens + nota.itens
    novo.valor = this.valor + nota.valor
    novo
  }

  Nota next() {
    println "next"
    this.itens += 1
    this.valor *= 2
    this
  }
}
```

```
@Test
void exercicio13() {
  Nota n1 = new Nota(itens: 2, valor: 20)
  Nota n2 = new Nota(itens: 2, valor: 20)
  Nota n3 = n1+ n2
  println n3.valor + " - "+ n3.itens
  n1++
  println n1.valor + " - "+ n1.itens
}
```

Exercício 13.2

```
@Test
void exercicio13ponto2() {
  BigDecimal vl = 10
  println vl
  vl = vl + 1
  println vl
  vl++
  println vl

  int a = 2
  vl = vl + a;
  println vl
  println vl - 5

  Date data = new Date()
  println  data
  data++
  println data
  data++
  println data + 10
  // consulte o documentação do groovy para saber detalhes das sobrecargas.
  // se caso não existir, vc pode implementar.
}
```

Exercício 13.3

```
package classes

trait Animal {
  String nome
  abstract void pular()
  void falar() {
    println "Animal "+ nome + " falando.."
  }
}
```

```
package classes

trait Lutador {
  String arma
  abstract void correr()
  void lutar() {
    println "Lutador "+ arma + " luitando..."
  }
}
```

```groovy
package classes

// O plugin de eclipse de groovy não sabe tratar trait, ele vai pensar que é uma interface
// se vc mandar o eclipse implementar ele vai gerar todas as implementações....
class Pessoa implements Animal, Lutador {
  @Override
  void pular() {
    println "pessoa " + nome + " pulando"
  }

  @Override
  void correr() {
    println "pessoa " + nome + " correndo"
  }
}
```

Exercício 14

```groovy
package classes

import groovy.transform.ToString

@ToString // Exemplo 1
//@ToString(excludes=["idade"])  // Exemplo 2
//@ToString(includeNames=true, excludes="idade, salario") // Exemplo 3
class Funcionario {
  String nome
  Integer idade
  Double salario
}

 @Test
 void exercicio14() {
   Funcionario f = new Funcionario(nome: "Fer", idade: 10, salario: 1500.59)
   println f
 }
```

Exercício 15

```
package classes

import groovy.transform.EqualsAndHashCode

@EqualsAndHashCode
class Funcionario2 {
  String nome
  Integer idade
  Double salario
}

 @Test
 void exercicio15() {
   Funcionario2 f1 = new Funcionario2(nome: "Fer", idade: 10, salario: 10)
   Funcionario2 f2 = new Funcionario2(nome: "Fer", idade: 10, salario: 10)
   // 1) execute na 1 sem a anotação para dar false.
   println f1.equals(f2)

   // 2) Acrescente a notação e depois executa denovo para dar true.
   println f1.equals(f2)
 }
```

Exercício 16

```
package classes

import groovy.transform.Immutable

@Immutable
class Pedido {
  String cliente
  Integer numero
}

 @Test
 void exercicio16() {
   //1) Parte
   Pedido p = new Pedido(cliente:"fernando", numero: 10)
   println p.cliente + "-" + p.numero
   println p

   //2) Parte - tenta alterar alguns atributo e vera erro de propriedade readOnly.
   //p.cliente = "fer"
 }
```

Exercício 17

```
package classes

@Singleton
class Conexao {
  Double valor
}
```

```
@Test
void exercicio17() {
  //1) Parte
  Conexao.instance.valor = 10
  println Conexao.instance.valor
  Conexao con =  Conexao.instance
  con.valor = 11
  println Conexao.instance.valor

  //2) Parte - tente criar uma instancia e vera o erro q não pode instanciar uma classe singleton.
  //Conexao x = new Conexao()
}
```

Exercício 18

```
package classes

import groovy.transform.builder.Builder

@Builder
class Comida {
  String fruta
  String bebida
  String doce
}
```

```
@Test
void exercicio18() {
  Comida comida = Comida.builder().fruta("maca").bebida("coca cola").doce("casadinho").build()
  println comida.fruta
  println comida.bebida
  println comida.doce
}
```

Exercício 19

```
@Test
void exercicio19() {
  def objeto = "texto"
  println objeto.getClass()

  objeto = 10
  println objeto.getClass()

  objeto = 10.00
  println objeto.getClass()

  objeto = new Nota(itens: 2, valor: 20)
  println objeto.getClass()
  println objeto.valor

  objeto = new Pedido(cliente:"fernando", numero: 10)
  println objeto.getClass()
  println objeto.cliente
}
```

Exercício 20

```
package classes

class Teste {
  def metodo(valor) {
    valor + 1
  }
}
```

```groovy
@Test
void exercicio20() {
  def teste = new Teste()
  def v1 = teste.metodo("fer")
  println v1.getClass()
  println v1

  v1 = teste.metodo(5)
  println v1.getClass()
  println v1

  v1 = teste.metodo(new BigDecimal(5))
  println v1.getClass()
  println v1

  def data = new Date()
  println data
  v1 = teste.metodo(data)
  println v1.getClass()
  println v1
}
```

Exercício 21

```groovy
@Test
void exercicio21() {
  def colecao = new ArrayList<String>()
  colecao.add("fer")
  colecao.add("anny")
  // executando tipo string
  for (item in colecao) {
    println item
  }

  colecao = new ArrayList<Integer>()
  colecao.add(1)
  colecao.add(2)
  // executando tipo integer
  for (item in colecao) {
    println item
  }

  colecao = "fernando esta aqui parado"
  // executando tipo carater
  for (item in colecao) {
    println item
  }

  colecao = 10
  // executando tipo inteiro unico
  for (item in colecao) {
    println item
  }
}
```