E-fólio Global | Instruções para a realização do E-fólio

UNIDADE CURRICULAR: Linguagens de Programação

CÓDIGO: 21077

DOCENTE: Ricardo José Vieira Baptista

A preencher pelo estudante

NOME: Ricardo Alexandre Castro Lopes Lobo

N.º DE ESTUDANTE: 2100622

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 25 de Setembro 2023

TRABALHO / RESOLUÇÃO:

GRUPO I

OCALM

```
(* Verifica se duas listas têm o mesmo tamanho *)

let same_size lst1 lst2 =

  List.length lst1 = List.length lst2



(* Calcula o valor com base em dois inteiros *)

let calc_value a b =

  if a = b then 1

  else if a > b then 2 * (a - b)

  else (a + b) / 2

(* lista final ou lista vazia*)

let create_list lst1 lst2 =

  if not (same_size lst1 lst2) then []

  else List.map2 calc_value lst1 lst2



(* exemplo com listas de igual tamanho *)

let () =

  let lst1 = [1; 2; 3; 4; 5] in

  let lst2 = [5; 4; 3; 2; 1] in
```

```
  let result = create_list lst1 lst2 in
  List.iter (Printf.printf "%d ") result;
  print_newline ()
```

PROLOG

```
% Verifica se duas listas têm o mesmo tamanho.
tamanho_igual([], []).
tamanho_igual([_|T1], [_|T2]) :-
    tamanho_igual(T1, T2).


% Calcula o valor final de cada elemento
valor(X, Y, Z) :-
    X == Y, !,
    Z = 1.0.
valor(X, Y, Z) :-
    X > Y, !,
    Z is 2.0 * (X - Y).
valor(X, Y, Z) :-
    Z is 0.5 * (X + Y).


% Constrói a lista resultado.
lista([], [], []).
```

```
lista([H1|T1], [H2|T2], [H3|T3]) :-
    valor(H1, H2, H3),
    lista(T1, T2, T3).

% predicado principal
nova_lista(L1, L2, L3) :-
    tamanho_igual(L1, L2),
    lista(L1, L2, L3),
    !.
nova_lista(_, _, []).

% espera-se
% 4 - 1 = 3 * 2 = 6.0
% 2 + 7 = 9 * 0.5 = 4.5
% 1.0
%
run :-
    nova_lista([4,2,5,6], [1, 7,5,6], L),
    write(L).
```

JAVA

```
import java.util.List;
```

```java
import java.util.ArrayList;

public class App {

public static List<Double> compareLists(List<Integer> list1, List<Integer>
list2) {
List<Double> resultList = new ArrayList<>();

// Verifica se as listas têm o mesmo tamanho
if (list1.size() != list2.size()) {
return resultList; // Retorna uma lista vazia
}

for (int i = 0; i < list1.size(); i++) {
int num1 = list1.get(i);
int num2 = list2.get(i);

if (num1 == num2) {
resultList.add(1.0);
} else if (num1 > num2) {
resultList.add(2.0 * (num1 - num2));
} else {
resultList.add((double)(num1 + num2) / 2);
}
}

return resultList;
}

public static void main(String[] args) {
List<Integer> list1 = List.of(4, 5, 1, 6);
List<Integer> list2 = List.of(3, 6, 1, 6);
List<Integer> list3 = List.of(3, 6, 1, 6, 10);

// espera-se
// 4 - 3 x 2 = 2.0
// 5 + 6 / 2 = 5.5
// 1.0
// 1.0

// exemplo listas com o mesmo tamanho
List<Double> result1 = compareLists(list1, list2);
System.out.println(result1);

// exemplo listas com tamanho diferente
List<Double> result2 = compareLists(list1, list3);
System.out.println(result2);
}
}
```

GRUPO II

Questão 1

```
% movimentos setembro 2023

movimento('ContaA', '+', 50, 1, 20230901, "alimentacao").

movimento('ContaB', '-', 100, 2, 20230902, "saude").

movimento('ContaC', '+', 150, 0, 20230903, "educacao").

movimento('ContaA', '+', 25, 1, 20230904, "saude").

movimento('ContaD', '-', 200, 3, 20230905, "restauracao").

movimento('ContaE', '+', 80, 1, 20230906, "saude").

movimento('ContaB', '+', 120, 2, 20230907, "alimentacao").

movimento('ContaA', '-', 50, 1, 20230908, "educacao").

% movimentos outubro 2023

movimento('ContaD', '+', 45, 0, 20231009, "alimentacao").

movimento('ContaC', '+', 90, 2, 20231010, "educacao").

movimento('ContaE', '-', 70, 1, 20231011, "saude").

movimento('ContaB', '+', 65, 1, 20231012, "restauracao").

movimento('ContaA', '+', 55, 1, 20231013, "alimentacao").
```

```
% movimentos novembro 2023

movimento('ContaC', '-', 40, 0, 20231114, "educacao").

movimento('ContaD', '+', 35, 2, 20231115, "educacao").

movimento('ContaE', '+', 20, 1, 20231116, "saude").

movimento('ContaA', '-', 60, 0, 20231117, "saude").

movimento('ContaB', '+', 10, 1, 20231118, "alimentacao").

movimento('ContaC', '+', 5, 0, 20231119, "educacao").


% para mostrar os nomes dos meses

nome_do_mes(1, 'Janeiro').

nome_do_mes(2, 'Fevereiro').

nome_do_mes(3, 'Março').

nome_do_mes(4, 'Abril').

nome_do_mes(5, 'Maio').

nome_do_mes(6, 'Junho').

nome_do_mes(7, 'Julho').

nome_do_mes(8, 'Agosto').

nome_do_mes(9, 'Setembro').

nome_do_mes(10, 'Outubro').

nome_do_mes(11, 'Novembro').

nome_do_mes(12, 'Dezembro').
```

```
% encontra os movimentos que estão ente data de inicio e fim

movimentos_entre_datas(DataInicio, DataFim, Lista) :-

    findall(

        (Conta, Tipo, Valor, Comissao, Data, Categoria),

        (movimento(Conta, Tipo, Valor, Comissao, Data, Categoria),
Data >= DataInicio, Data =< DataFim),

        Lista).



% Caso base: a soma de uma lista vazia é 0.

total_movimentos([], 0).

% Caso recursivo

total_movimentos([(_, Tipo, Valor, _, _, _)|T], Total) :-

    total_movimentos(T, TotalRestante),

    ( Tipo = '+' ->

        Total is Valor + TotalRestante

    ;

        Total is TotalRestante - Valor

    ).


% Mesma funcionalidade, mas mais direto
```

```
mes_da_data(Data, Mes) :-
    Mes is (Data // 100) mod 100.


% cria uma lista de meses unicos com base na sublista de movimentos
meses_unicos(Movimentos, ListaMeses) :-
    findall(
        Mes,
        (member((_, _, _, _, Data, _), Movimentos), mes_da_data(Data, Mes)),
        MesesRepetidos
    ),
    sort(MesesRepetidos, ListaMeses).


% criar uma sublista de movimentos que ocorram num mes
movimentos_de_mes(Movimentos, Mes, MovimentosMes) :-
    findall(
        Movimento,
        (member(Movimento, Movimentos), Movimento = (_, _, _, _, Data, _), mes_da_data(Data, Mes)),
        MovimentosMes
    ).
```

```
%
lista_de_listas_com_mes(Movimentos, Meses, ListaFinal) :-
    meses_unicos(Movimentos, Meses),
    findall(
        MovimentosMes,
        (member(Mes, Meses), movimentos_de_mes(Movimentos, Mes,
MovimentosMes)),
        ListaFinal
    ).

total_da_lista(Movimentos, Total) :-
    total_movimentos(Movimentos, Total).

incluir_por_categoria([], _, []).
incluir_por_categoria([H|RestoMovimentos], Categoria, [H|Resto]) :-
    H = (_, _, _, _, _, CategoriaAtual),
    CategoriaAtual = Categoria,
    incluir_por_categoria(RestoMovimentos, Categoria, Resto).
incluir_por_categoria([(_, _, _, _, _, CategoriaAtual)|RestoMovimentos],
CategoriaDesejada, Lista) :-
    CategoriaAtual \= CategoriaDesejada,
```

```
    incluir_por_categoria(RestoMovimentos, CategoriaDesejada, Lista).



total_por_categoria_e_mes(Movimentos, Mes, Categoria, Total) :-

    movimentos_de_mes(Movimentos, Mes, MovimentosMes),

    incluir_por_categoria(MovimentosMes, Categoria,
MovimentosCategoria),

    total_movimentos(MovimentosCategoria, Total).


% predicado principal

valoresPorCategEntreDatas(DataInicio, DataFim, Categoria,
ListaResultados) :-

    movimentos_entre_datas(DataInicio, DataFim, Movimentos),

    meses_unicos(Movimentos, Meses),

    findall(

        (Mes, Categoria, Total),

        (

            member(Mes, Meses),

            total_por_categoria_e_mes(Movimentos, Mes, Categoria, Total)

        ),

        ListaResultados

    ).
```

```
imprimir_totais_por_mes([]).

imprimir_totais_por_mes([(Mes, Categoria, Total)|Resto]) :-

    nome_do_mes(Mes, NomeMes),

    format("Mês: ~w (~w) - Total: ~2f~n", [NomeMes, Categoria,
Total]),

    imprimir_totais_por_mes(Resto).


run :-

    write("Lista Educacao"),

    nl,

    valoresPorCategEntreDatas(20230901, 20231231, "educacao",
ListaEducacao),

    imprimir_totais_por_mes(ListaEducacao),

    nl,

    write("Lista Saude"),

    nl,

    valoresPorCategEntreDatas(20230901, 20231231, "saude",
ListaSaude),

    imprimir_totais_por_mes(ListaSaude),

    nl,

    write("Lista Restauracao só Setembro"),
```

nl,

valoresPorCategEntreDatas(20230901, 20230930, "restauracao", ListaRestauracao),

imprimir_totais_por_mes(ListaRestauracao).

Questão 2

Parte 1

Cliente.java

```
import java.time.LocalDate;

import java.util.ArrayList;
import java.util.List;

public class Cliente {
private int numeroCliente;
private String nome;
private String agencia;
private String cidade;
private LocalDate dataAbertura;
private List<Movimento> movimentos;
private double saldo;

public Cliente(int numeroCliente, String nome, String agencia, String cidade, LocalDate dataAbertura) {
this.numeroCliente = numeroCliente;
this.nome = nome;
this.agencia = agencia;
this.cidade = cidade;
this.dataAbertura = dataAbertura;
this.movimentos = new ArrayList<>();
this.saldo = 0;
}
```

```
public int getNumeroCliente() {
return numeroCliente;
}

public void setNumeroCliente(int numeroCliente) {
this.numeroCliente = numeroCliente;
}

public String getNome() {
return nome;
}

public void setNome(String nome) {
this.nome = nome;
}

public String getAgencia() {
return agencia;
}

public void setAgencia(String agencia) {
this.agencia = agencia;
}

public String getCidade() {
return cidade;
}

public void setCidade(String cidade) {
this.cidade = cidade;
}

public LocalDate getDataAbertura() {
return dataAbertura;
}

public void setDataAbertura(LocalDate dataAbertura) {
this.dataAbertura = dataAbertura;
}

public List<Movimento> getMovimentos() {
return movimentos;
}

public void setMovimentos(List<Movimento> movimentos) {
this.movimentos = movimentos;
```

```
}

public double getSaldo() {
return saldo;
}

public void setSaldo(double saldo) {
this.saldo = saldo;
}

public double verSaldoAtual() {
return saldo;
}
```

Movimento.java

```
import java.time.LocalDate;

public class Movimento {
private LocalDate data;
private double valor;

public Movimento(LocalDate data, double valor) {
this.data = data;
this.valor = valor;
}

public LocalDate getData() {
return data;
}

public double getValor() {
return valor;
}
}
```

SistemaBancario.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class SistemaBancario {
private List<Cliente> clientes;
```

```
public SistemaBancario() {
this.clientes = new ArrayList<>();
}

public void adicionarCliente(Cliente cliente) {
clientes.add(cliente);
}


public List<Cliente> getClientes() {
return clientes;
}

public void setClientes(List<Cliente> clientes) {
this.clientes = clientes;
}
}
```

App.java

```
import java.time.LocalDate;


public class App {
public static void main(String[] args) {

// ver saldo atual de um cliente acabado de criar
Cliente jose = new Cliente(123, "José Castro", "Aliados", "Porto",
LocalDate.of(2010, 1, 20));
Cliente maria = new Cliente(124, "Maria Castro", "Boavista", "Porto",
LocalDate.of(2012, 1, 20));
Cliente joana = new Cliente(125, "Joana Castro", "Rossio", "Lisboa",
LocalDate.of(2013, 1, 20));
Cliente miguel = new Cliente(126, "Miguel Castro", "Marques", "Lisboa",
LocalDate.of(2017, 1, 20));
// ver saldo do cliente jose
jose.verSaldoAtual();

// o saldo ao abrir a conta
System.out.println("----------------------------------------------");
System.out.printf("O saldo inicial do cliente %s é de %.2f euros %n",
jose.getNome(), jose.verSaldoAtual());

// fazer um depósito
```

```
System.out.println("----------------------------------------------");
jose.efetuarMovimento(LocalDate.of(2023, 9, 25), 100);
System.out.printf("O saldo do cliente %s é de %.2f euros, após depósito %n",
jose.getNome(), jose.verSaldoAtual());
// fazer um levantamento
System.out.println("----------------------------------------------");
jose.efetuarMovimento(LocalDate.of(2023, 9, 27), -50);
System.out.printf("O saldo do cliente %s é de %.2f euros, após levantamento
%n", jose.getNome(), jose.verSaldoAtual());


// listas os movimentos do cliente jose
System.out.println("----------------------------------------------");
System.out.printf("Lista de movimentos do cliente %s: %n %n",
jose.getNome());
jose.imprimirMovimentos();

SistemaBancario banco = new SistemaBancario();

// adicionar clientes ao banco
banco.adicionarCliente(jose);
banco.adicionarCliente(maria);
banco.adicionarCliente(joana);
banco.adicionarCliente(miguel);

// todos os clientes do banco
System.out.println("----------------------------------------------");
System.out.println("Lista de todos os Clientes: ");
banco.imprimirClientes();


// clientes do Porto
System.out.println("----------------------------------------------");
System.out.println("Lista dos clientes do Porto: ");
banco.imprimirClientesPorCidade("Porto");
}
}
```

Cliente.java

```
import java.time.LocalDate;

import java.util.ArrayList;
import java.util.List;

public class Cliente {
private int numeroCliente;
```

```
private String nome;
private String agencia;
private String cidade;
private LocalDate dataAbertura;
private List<Movimento> movimentos;
private double saldo;

public Cliente(int numeroCliente, String nome, String agencia, String cidade,
LocalDate dataAbertura) {
this.numeroCliente = numeroCliente;
this.nome = nome;
this.agencia = agencia;
this.cidade = cidade;
this.dataAbertura = dataAbertura;
this.movimentos = new ArrayList<>();
this.saldo = 0;
}

public void efetuarMovimento(LocalDate data, double valor) {
movimentos.add(new Movimento(data, valor));
saldo += valor;
}

public int getNumeroCliente() {
return numeroCliente;
}

public void setNumeroCliente(int numeroCliente) {
this.numeroCliente = numeroCliente;
}

public String getNome() {
return nome;
}

public void setNome(String nome) {
this.nome = nome;
}

public String getAgencia() {
return agencia;
}

public void setAgencia(String agencia) {
this.agencia = agencia;
}

public String getCidade() {
return cidade;
```

```
}

public void setCidade(String cidade) {
this.cidade = cidade;
}

public LocalDate getDataAbertura() {
return dataAbertura;
}

public void setDataAbertura(LocalDate dataAbertura) {
this.dataAbertura = dataAbertura;
}

public List<Movimento> getMovimentos() {
return movimentos;
}

public void setMovimentos(List<Movimento> movimentos) {
this.movimentos = movimentos;
}

public double getSaldo() {
return saldo;
}

public void setSaldo(double saldo) {
this.saldo = saldo;
}

public double verSaldoAtual() {
return saldo;
}

public void imprimirMovimentos() {
movimentos.stream()
.sorted((m1, m2) -> m1.getData().compareTo(m2.getData()))
.forEach(m -> System.out.println("Data: " + m.getData() + ", Valor: " +
m.getValor()));
System.out.println("Saldo Atual: " + verSaldoAtual());
}
}
```

Movimento.java

```
import java.time.LocalDate;
```

```java
public class Movimento {
private LocalDate data;
private double valor;

public Movimento(LocalDate data, double valor) {
this.data = data;
this.valor = valor;
}

public LocalDate getData() {
return data;
}

public double getValor() {
return valor;
}
}
```

SistemaBancario.java

```java
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class SistemaBancario {
private List<Cliente> clientes;

public SistemaBancario() {
this.clientes = new ArrayList<>();
}

public void adicionarCliente(Cliente cliente) {
clientes.add(cliente);
}

public void imprimirClientes() {
clientes.forEach(c -> System.out.println("Cliente: " + c.getNome() + ", Cidade: "
+ c.getCidade() + " (" + c.getAgencia() + ")"));
}

public void imprimirClientesPorCidade(String cidade) {
List<Cliente> clientesDaCidade = clientes.stream()
.filter(c -> c.getCidade().equalsIgnoreCase(cidade))
.collect(Collectors.toList());

clientesDaCidade.forEach(c -> System.out.println("Cliente: " + c.getNome() + ",
Cidade: " + c.getCidade()));
}
```

```
public List<Cliente> getClientes() {
return clientes;
}

public void setClientes(List<Cliente> clientes) {
this.clientes = clientes;
}
}
```