

**DISCIPLINA: PARADIGMAS DE PROGRAMAÇÃO**

**RELATÓRIO JAVASCRIPT**

**(2022)**

**ALUNOS:**

* **Jefferson Gomes de Almeida - 192080016;**
* **Ângelo Gabriel Paz da Silva - 192080075;**
* **Kennedy Johnson de Sousa Dantas - 192080156**

**Prof(ª). Janderson Jason Barbosa Aguiar**

  CAMPUS CAMPINA GRANDE, março de 2022

1.

% fatos

cursa(angelo,pp).

cursa(caio,pp).

cursa(dennis,pp).

cursa(eduarda,pp).

cursa(elioenai,pp).

cursa(eliseu,pp).

cursa(ewerton,pp).

cursa(gustavobarros,pp).

cursa(gustavomedeiros,pp).

cursa(jefferson,pp).

cursa(joao,pp).

cursa(jose,pp).

cursa(kennedy,pp).

cursa(klayton,pp).

cursa(lara,pp).

cursa(luanderlandy,pp).

cursa(lucas,pp).

cursa(mariana,pp).

cursa(metheussales,pp).

cursa(mateussousa,pp).

cursa(matheusrocha,pp).

cursa(metheuscosta,pp).

cursa(patricio,pp).

cursa(rafaela,pp).

cursa(renan,pp).

cursa(romenildo,pp).

cursa(simeone,pp).

cursa(thiago,pp).

cursa(wilton,pp).

cursa(gustavobarros, aps).

cursa(gustavomedeiros,aps).

cursa(metheussales,aps).

cursa(matheussousa,aps).

cursa(metheusrocha,aps).

cursa(metheuscosta,aps).

cursa(eduarda,tec).

cursa(lara,tec).

cursa(mariana,tec).

cursa(refaela,tec).

cursa(angelo,tebd).

cursa(caio,tebd).

cursa(carlos,tebd).

cursa(dennis,tebd).

cursa(elioenai,tebd).

cursa(eliseu,tebd).

cursa(ewerton,tebd).

cursa(jefferson,tebd).

cursa(joao,tebd).

cursa(jose,tebd).

cursa(kennedy,tebd).

cursa(klayton,tebd).

cursa(luanderlandy,tebd).

cursa(lucas,tebd).

cursa(patricio,tebd).

cursa(renan,tebd).

cursa(romenildo,tebd).

cursa(simeone,tebd).

cursa(thiago,tebd).

cursa(wilton,tebd).

obrigatorio(pp).

obrigatorio(aps).

eletiva(tec).

eletiva(tebd).

leciona(janderson,pp).

leciona(janderson,aps).

leciona(paulo,tec).

leciona(vladimir,tebd).

% regras

professor(X,Y) :-

cursa(Y,Z),leciona(X,Z).

i)

?- obrigatorio(X).

X = pp ;

X = aps.

ii)

?- professor(janderson,X).

X = angelo ;

X = caio ;

X = dennis ;

X = eduarda ;

X = elioenai ;

X = eliseu ;

X = ewerton ;

X = gustavobarros ;

X = gustavomedeiros ;

X = jefferson ;

X = joao ;

X = jose ;

X = kennedy ;

X = klayton ;

X = lara ;

X = luanderlandy ;

X = lucas ;

X = mariana ;

X = metheussales ;

X = mateussousa ;

X = matheusrocha ;

X = metheuscosta ;

X = patricio ;

X = rafaela ;

X = renan ;

X = romenildo ;

X = simeone ;

X = thiago ;

X = wilton ;

X = gustavobarros ;

X = gustavomedeiros ;

X = metheussales ;

X = matheussousa ;

X = metheusrocha ;

X = metheuscosta ;

false.

iii)

?- professor(X,kennedy).

X = janderson ;

X = vladimir.

iv)

?- eletiva(X),leciona(Y,X).

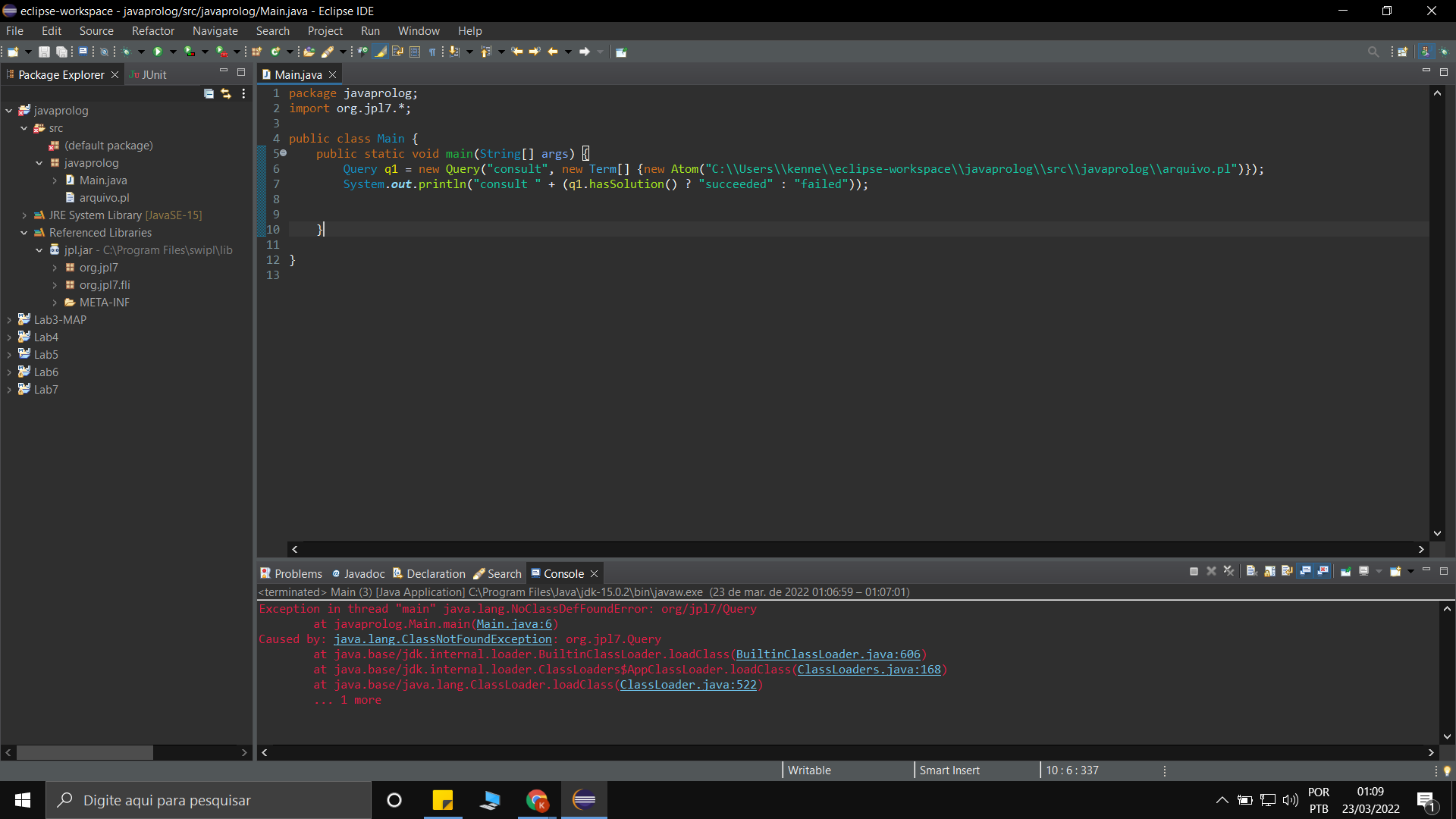
X = tec,

Y = paulo ;

X = tebd,

Y = vladimir.

b) Não conseguimos rodar prolog em java, as instruções do slide não funcionam, tivemos que buscar vídeos na internet e mesmo fazendo igual é ensinado no vídeo não deu certo rodar



2.a)

main:: IO()

main = do

putStrLn "Inserir os dados do aluno (notas entre 0 e 100):"

putStrLn "-Frequencia:"

freq <- getLine

let frequencia = (read freq :: Int)

putStrLn "-Nota 1:"

nota01 <- getLine

let nota1 = (read nota01 :: Int)

putStrLn "-Nota 2:"

nota02 <- getLine

let nota2 = (read nota02 :: Int)

putStrLn "Situação do aluno:"

situacaoAluno frequencia nota1 nota2

situacaoAluno :: Int -> Int -> Int -> IO()

situacaoAluno fequencia nota1 nota2 = do

let media = div (nota1 + nota2) 2

if fequencia < 25 then

putStrLn "Reprovado por falta"

else if media < 30 then

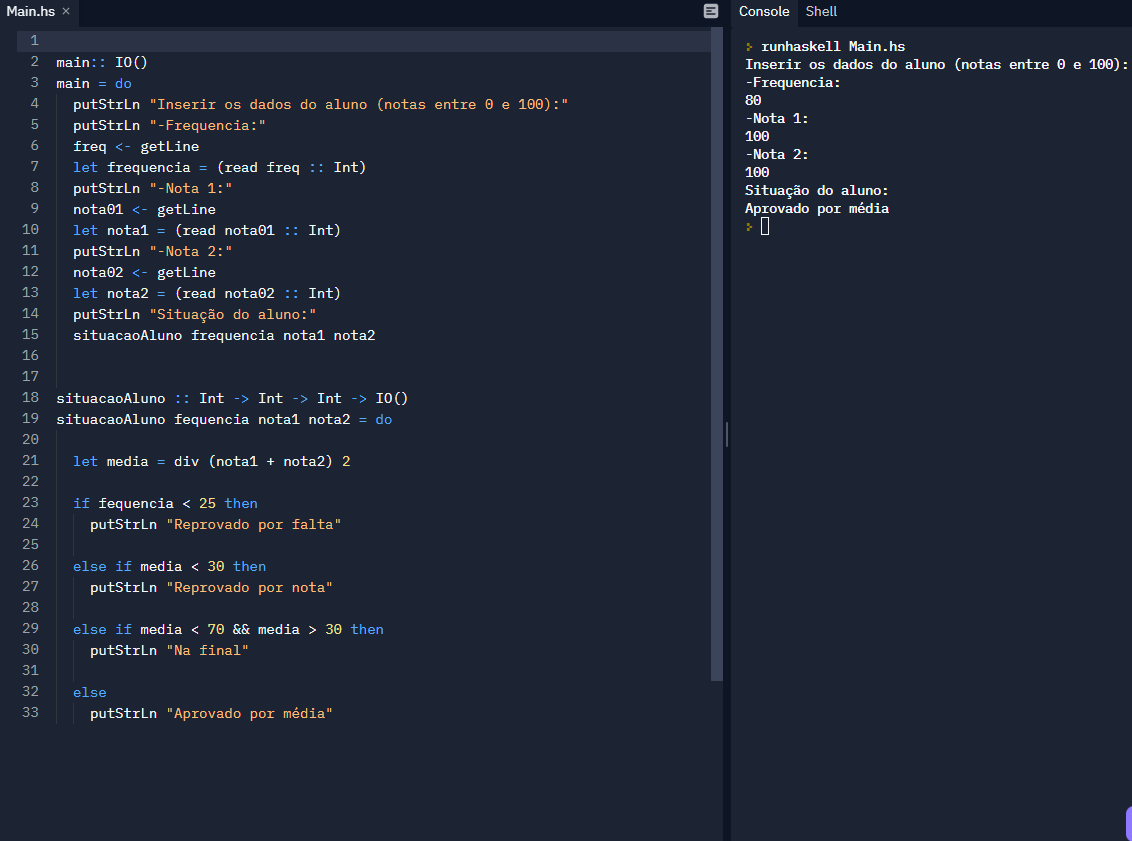
putStrLn "Reprovado por nota"

else if media < 70 && media > 30 then

putStrLn "Na final"

else

putStrLn "Aprovado por média"



b)

main:: IO()

main = do

let lista = [9,1,8,2,5,7,3,6,4]

let listaOrdenada = qsort lista

let listaPares = pares listaOrdenada

print (show listaPares)

qsort :: [Int] -> [Int]

qsort [] = []

qsort (x:xs) = qsort large ++ mid ++ qsort small

where

small = [y | y<-xs, y<x]

mid = [y | y<-xs, y==x] ++ [x]

large = [y | y<-xs, y>x]

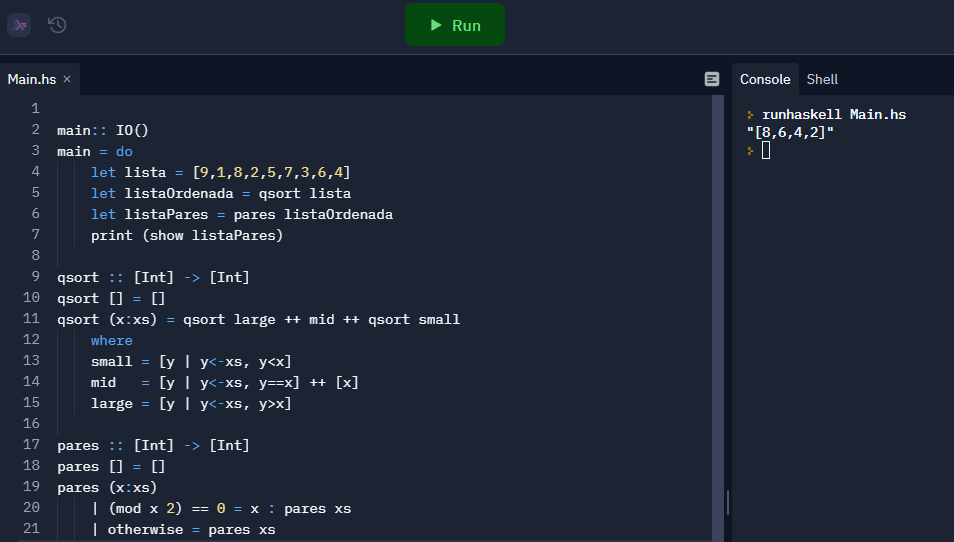
pares :: [Int] -> [Int]

pares [] = []

pares (x:xs)

| (mod x 2) == 0 = x : pares xs

| otherwise = pares xs



3.

def quicksort(array):

if len(array) <= 1:

return array

higher = [i for i in array[1:] if i <= array[0]]; lower = [i for i in array[1:] if i > array[0]]; array = quicksort(lower) + [array[0]] + quicksort(higher); return [num for num in lista if num % 2 == 0]

lista = [9,1,8,2,5,7,3,6,4]

lista = quicksort(lista)

print(lista)

def split(w):

x=w[0]

left=[]

right=[]

for i in range(1,len(w)):

if w[i]>=x:

left+=[w[i]]

else:

right+=[w[i]]

return left,x,right

def quicksort(w):

if len(w)<2:

return w

else:

w1,x,w2=split(w)

w = quicksort(w1)+[x]+quicksort(w2)

return [num for num in w if num % 2 == 0]

lista = [9,1,8,2,5,7,3,6,4]

lista = quicksort(lista)

print(lista)

**Readability**: A facilidade de manutenção é determinada em grande parte, pela **legibilidade** dos programas, ela se tornou uma medida importante da qualidade dos programas e das linguagens. A legibilidade deve ser considerada no contexto do domínio do problema. Em relação aos códigos, a legibilidade do segundo está bem melhor em relação ao primeiro, pelo motivo que não possui vários comandos de código em uma única linha.

**Writability**: É a medida da facilidade em que uma linguagem pode ser usada para criar programas para um domínio de problema escolhido. A maioria das características da linguagem que afetam a legibilidade também afetam a **capacidade de escrita**. Deve ser considerada no contexto do domínio de problema-alvo da linguagem. Em relação aos códigos, o código com melhor capacidade de escrita seria o primeiro, que em apenas poucas linhas faz a mesma coisa que o segundo código, mesmo sendo na mesma linguagem.

4)

a) O paradigma orientado a aspectos é um paradigma baseado em meta informações. Programação Orientada a Aspectos é um modelo de programação que possibilita a devida separação de responsabilidades, considerando funcionalidades que são essenciais para um grupo de objetos, mas não são de responsabilidade direta deles.

AOP é dita ortogonal porque geralmente envolve conceitos que independem da camada e não tem relação direta com os requisitos funcionais de um sistema.

**AspectJ** é uma linguagem de programação pertencente ao [paradigma orientado a aspectos](https://pt.wikipedia.org/wiki/Paradigma_orientado_%C3%A0_aspectos), que consiste basicamente em retirar os interesses transversais de um programa em [Java](https://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o)).

b) Programação concorrente ou programação simultânea é um paradigma de programação para a construção de [programas de computador](https://pt.wikipedia.org/wiki/Programas_de_computador) que fazem uso da execução simultânea de várias tarefas computacionais interativas, que podem ser implementadas como programas separados ou como um conjunto de [*threads*](https://pt.wikipedia.org/wiki/Thread_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)) criadas por um único programa. Programação concorrente é relacionada com [programação paralela](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_paralela), mas foca mais na interação entre as tarefas.

A principal vantagem do uso da programação concorrente é o aumento do desempenho dos programas, pois é possível aumentar a quantidade de tarefas executadas em um determinado período de tempo.

O grande desafio da programação concorrente é o compartilhamento de recursos, a comunicação e a interação entre os programas que são executados concorrentemente

c) Semântica denotacional , em que cada frase da língua é interpretada como uma *denotação* , ou seja, um significado conceitual que pode ser pensado abstratamente. Essas denotações são frequentemente objetos matemáticos que habitam um espaço matemático, mas não é obrigatório que o sejam. Como uma necessidade prática, as denotações são descritas usando alguma forma de notação matemática, que por sua vez pode ser formalizada como uma metalinguagem denotacional. Por exemplo, a semântica denotacional de linguagens funcionais frequentemente traduz a linguagem em **teoria de domínio.**