**Curso POO teoria** [**#08a**](https://www.youtube.com/results?search_query=%2308a) **- Relacionamento de Agregação**

|  |
| --- |
| Luta |
| * desafiado * desafiante * rounds * aprovada |
| + marcarLuta()  + lutar() |

Agregação:

A agregação é uma forma de associação caracterizada por um relacionamento do tipo parte-todo, em que uma das pontas da

associação é a parte é o todo.

Por exemplo as classes Turma, Estudante e Instrutor, o Estudante é ligado a Turma, o Instrutor também e ligado a Turma.

Ou seja Turma é um todo, enquanto os Estudantes e o Instrutor são partes do todo.

**RELEMBRANDO**

Objetos: instância de uma classe

Classe: encapsula dados e funcionalidades

Uma classe liga a outra

Agregação é conhecida como relacionamento tem um: como uma luta tem um lutador.

Classe Luta

//Atributos

Desafiado é Instância, ou seja um objeto da classe Lutador

Desafiante é uma instância da outra classe Lutador

Privado desafiado: Lutador

Privado desafiante: Lutador

Privado rounds: inteiro

Privado aprovado: Lógico

|  |
| --- |
| * desafiado * desafiante * rounds * aprovada |
| + marcarLuta()  + lutar() |

Publico Metodo marcarLuta()

FimMetodo

Publico Metodo Lutar()

FimMetodo

Publico setDesafiado(d:Lutador)

desafiado=d

fimMetodo

Publico getDesafiado()

Retorne desafiado

FimMetodo

Publico setDesafiante(de:Lutador)

Desafiante = d;

FimMetodo

Publico getDesafiante()

Retorne desafiante

FimMetodo

Publico setRounds (r:inteiro)

rounds = r

FimMetodo

Publico getRounds ()

Retorne rounds

FimMetodo

Publico setAprovada(a: Logico)

aprovada=a

FimMetodo

Publico getAprovada()

Retorne aprovada

FimMetodo

OBS:

Só pode ser marcada entre lutadores da mesma categoria

Desafiado e desafiante devem ser lutadores diferentes

<> = Sinal de Diferente

Classe Luta

//Métodos

Publico método marcarLuta(l1,l2: Lutador)

Se(l1.getCategoria() = l2.getCategoria())

e (l1 <> l2) então

aprovada = verdadeiro

desafiado = l1

desafiante = l2

senao

aprovada = falso

desafiado = nulo

desafiante = nulo

fimse

FimMetodo

Publico método lutar()

Se (aprovada)

Desafiado.apresentar()

Desafiante.apresentar()

Vencedor = aleatório(0...2)

escolha(vencedor)

caso 0: //Empate

Escreva(“Empatou!”)

desafiado.empatarLuta()

desafiante.empatarLuta()

caso 1: //Ganhou Desafiado

Escreva( desafiado.getNome())

desafiado.vencerLuta()

desafiante.perderLuta()

caso 2: //Ganhou Desafiante

Escreva(desafiante.getNome() )

desafiado.perderLuta()

desafiante.vencerLuta()

fim escolha

senao

Escreva(“ Luta não pode acontecer ”)

Fimse

FimMetodo

FimClasse

//Programa Principal

L: vetor [0 ... 5]

L[0]: novo Lutador (“PrettyBoy”,

“França” ,

31, 1.75 , 68.9 , 11 , 3 , 1)

L[1]: novo Lutador (“Putscript”,

“Brasil” ,

29, 1.68 , 57.8 , 14 , 2, 3)

L[2]: novo Lutador (“Snapshadow”,

“EUA” ,

35, 1.65 , 80.9 , 12 , 3,1)

L[3]: novo Lutador (“DeadCole”,

“Austrália” ,

23, 1.93 , 81.6, 13 , 0,2)

L[4]: novo Lutador ()

L[0] = novo Lutador (......)

L[1] = novo Lutador (......)

L[2] = novo Lutador (......)

L[3] = novo Lutador (......)

L[4] = novo Lutador (......)

L[5] = novo Lutador (......)

UEC01 = new Luta()

UEC01.marcarLuta()

UEC01.lutar()

A programação orientada a objetos se preocupa em produzir um software que tenha as seguintes características:

Comer nada

Confiável

Oportuna

Manutenível

Extensível

Reutilizável

Natural

Objeto é uma instância de uma classe.

3 Pilares da POO

Impressora e H P

Encapsulamento

Herança

Polimorfismo