POO aula 8: Agregação

"Objetos são instâncias de classe, todo objeto para existir precisa de uma classe"

Relacionamento estudado da aula: Agregação

o'que é agregação: É uma maneira de definir uma associação entre classes, onde no contexto

agregação, uma classe é composta por uma ou mais instâncias de outras classes.

Onde também é conhecida por ser uma relação de "has-a" ou "tem-uma", diferente da "is-a"

ou "é-uma"(esse sendo um conceito de herança).

Nesse exemplo da aula, iremos criar uma classe Luta, onde vai ser associada às instâncias

(objetos) da classe Lutador, perceba que a relação entre eles se encaixa um pouco no conceito

de agregação pois, toda Luta "tem-um" lutador(nesse caso são vários[\*]) e todo lutador tem

nenhuma ou várias lutas(sendo 0..\*)

A classe exemplo da aula é a classe Luta

tendo os seguintes atributos privados(para obedecer o encapsulamento)

desafiado:tipo abstrato de dados(ou seja vai ser colocado um instanciamento de classe)

desafiante:tipo abstrato de dados

round:int

aprovada:bool

onde as regras de cada luta são:

Só pode ser marcada entre lutadores da mesma categoria.

-Desafiado e desafiante devem ser lutadores diferentes.

-Só pode acontecer se estiver aprovada.

-Só pode ter como resultado a vitória de um dos lutadores ou o empate.

em algoritmo a classe vai ser a seguinte

<> = Diferente em algoritmo

classe Luta

//atributos

privado desafiado: Lutador

privado desafiante: Lutador

privado rounds: Inteiro

privado aprovada: Lógica

//Métodos Especiais

publico método getDesafiado()

retorne desafiado

FimMétodo

publico método setDesafiado(dd: Lutador)

desafiado = dd

FimMétodo

publico método getDesafiante()

retorne desafiante

FimMétodo

publico método setDesafiante(dt: Lutador)

desafiante = dt

FimMétodo

publico método getRounds()

retorne rounds

FimMétodo

publico método setRounds(r:Inteiro)

rounds = r

FimMétodo

publico método getAprovada()

retorne aprovada

FimMétodo

publico método setAprovada(aprvd: Lógico)

aprovada = aprvd

FimMétodo

//Métodos

publico método marcarLuta(lutador[0], lutador[1]: Lutador)

se (lutador[0].getCategoria() == lutador[1].getCategoria)

e (lutador[0] <> lutador[1]) então

aprovada = verdadeiro

desafiado = lutador[0]

desafiante = lutador[1]

senao então

aprovada = falso

desafiado = nulo

desafiante = nulo

FimMétodo

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: //Desafiado Ganhou

escreva(desafiado.getNome(), "Ganhou")

desafiado.ganharLuta()

desafiante.perderLuta()

caso 2: //Desafiante Ganhou

escreva(desafiante.getNome(), "Ganhou")

desafiante.ganharLuta()

desafiado.perderLuta()

senao

escreva("Luta não pode acontecer")

FimSe

FimMétodo

\*Aqui será o programa principal(index) Min 28

// Programa Principal

L: vetor[0..5]

L[0] = novo Lutador (“Pretty Boy”, “França”, 31, 1.75, 68.9, 11, 3, 1)

L[1] = novo Lutador (“Putsscript”, “Brasil”, 29, 1.68, 57.8, 14, 2, 3)

L[2] = novo Lutador (“Snapshadow”, “EUA”, 35, 1.65, 80.9, 12, 2, 1)

L[3] = novo Lutador (“Dead Code”, “Austrália”, 28, 1.93, 81.6, 13, 0, 2)

L[4] = novo Lutador (“Ufocobol”, “Brasil”, 37, 1.70, 119.3, 5, 4, 3)

L[5] = novo Lutador (“Nerdaard”, “EUA”, 30, 1.81, 105.7, 12, 2, 4)

UEC01 = novo Luta()

UEC01.marcarLuta(L[0], L[1])

UEC01.lutar()

classe Lutador

FimClasse

(...)

Classe Luta

(...)

FimClasse