## **RELACIONAMENTOS EM POO** O que são os Relacionamentos em POO? O que é o conceito de **Associação**? Como podemos representar uma Associação com ferramentas UML? Ilustre o uso das ferramentas UML para ilustrar uma Associação Como a Associação pode ser realizada no momento do desenvolvimento do código? O que é o conceito de Agregação? Como podemos representar uma Agregação com ferramentas UML? Ilustre o uso das ferramentas UML para ilustrar uma agregação - reservada para a questão acima -

Como a Agregação pode ser realizada no momento

do desenvolvimento do código?

## **RELACIONAMENTOS EM POO**

Relacionamentos: ocorrem quando os elementos de uma classe se relacionam com os elementos de outra classe. Na POO temos 3 tipos diferentes de relacionamentos dependendo do seu nível de entrelaçamento e grau de impacto sobre os elementos das classes relacionadas. Esses relacionamentos são:

- Associação (Relacionamento Fraco Muitos para Muitos);
- Agregação (Relacionamento Médio 1 para Muitos);
- Composição (Relacionamento Forte Relacionamento de Morte);

OBS: Cuidado para não confundir com o conceito de Herança! Relacionamentos e Herança são conceitos diferentes.

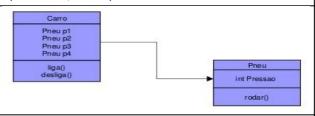
## Associação:

Ocorre quando temos classes completamente independentes entre si mas que eventualmente estarão relacionadas. A Associação é considerada um relacionamento muitos para muitos, onde não há herança e nem dependência entre elas, sua relação será totalmente eventual. Podemos exemplificá-la pela relação que existe entre professores e alunos, 1 aluno pode ter vários professores e 1 professor pode ter vários alunos. Um não depende do outro para existir. Professores podem existir sem alunos e e alunos podem existir sem professores, mas eventualmente eles terão de trabalhar juntos.

Uma Associação deve ser representada pelos seguintes elementos:

- Relaciomento: uma seta que direciona para a classe atribuindo valores associados;
- Grau de Relacionamento: é sempre muitos para muitos;

OBS: esse é o grau de relacionamentos mais fraco, onde nenhum elemento depende do outro, esse é o tipo de relacionamento menos utilizado.



Primeiro: temos que ter 2 classes, onde uma será a Classe Abstraída - aquela que irá construir as instâncias que serão associadas - e a outra será a Classe associadora - aquela que irá coletar os objetos da outra classe;

Segundo: vamos atribuir o elemento que desejamos associar a uma variável, depois vamos atribuir essa variável ao elemento da classe associadora;

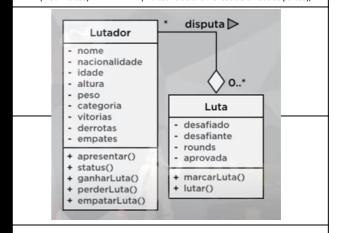
Terceiro: qualquer modificação acontecerá na variável, os elementos das classes são totalmente independentes:

## Agregação:

É o como chamamos um relacionamento entre duas ou mais classes, onde é possível que uma classe possa utilizar as instâncias da outra classe atribuíndo essas instâncias sobre os próprios objetos gerados na segunda classe e gerendo operações sobre essas instâncias. Quando um objeto recebe a instância de outra classe temos uma atribuição do tipo: "**Tem 1**", onde um objeto de uma classe contém outro obieto de outra classe. Esse tipo de dado recebe o nome de "Tipo Abstrato", pois ele é justamente objeto abstraído de uma classe, por isso recebe esse nome.

Uma Agregação deve ser representada pelos seguintes elementos:

- Relaciomento: uma linha com um losângulo na ponta onde a agregação está sendo recebida;
- Multiplicidade: semelhante as cardinalidades em bancos de dados, a multiplicidade demonstra o número de interações que pode haver entre uma classe e outra para aquele tipo de relacionamento. Dentre esses números temos:
- 0..\* (nenhum ou muitos) 1..\* (1 ou muitos)
- 1 (1 relacionamento de ambas as partes)
  \* (muitos relacionamentos de ambas as partes):



Primeiro: temos que ter 2 classes, onde uma será a Classe Abstraída - aquela que irá construir as instâncias que serão agregadas - e a outra será a Classe Agregadora - aquela que irá agregar as instâncias da outra classe, geralmente dentro de um vetor;

Segundo: devemos criar um método de inserção dos tipos abstratos para a classe agregadora;

Terceiro: depois de atribuídos sobre a classe agregadora, é só utilizar os métodos sobre as instâncias:

| RELACIONAMENTOS EM POO   | RELACIONAMENTOS EM POO   |
|--|--|
| O que é o conceito de <b>Composição</b> ?                                      | Composição: A composição é uma agregação que possui dependência entre os objetos, ou seja, se o objeto principal for destruído, os objetos que o compõe não podem existir mais. Há a chamada relação de morte, onde, se a classe principal for deletada as outras serão inutilizadas. Um exemplo é a relação entre uma a universidade e os departamentos. Além da universidade possuir vários departamentos, eles só podem existir se a universidade existir. Há uma dependência.          |
| Como podemos <b>representar uma Composição com</b><br><b>ferramentas UML</b> ? | Uma Composição deve ser representada pelos seguintes elementos:  - Relaciomento: uma linha com o Losângulo Preto na ponta da Classe Principal;  - Grau de Relacionamento: é sempre um relacionamento de morte, se a classe principal morrer as outras morrem;  OBS: esse é o grau de relacionamentos mais forte.   |
| llustre o uso das ferramentas UML para ilustrar<br>uma Composição              | ContaCorrente  float Saido float ChequeEspecial deposita() saque() extrato()  ContaCorrente contas() Pupanca poups() iniciaBanco() sabreConta() sabreConta() sabreConta() sabreConta() sabreConta() sabreConta() sabreConta()  |
| Como a Composição pode ser realizada no momento do desenvolvimento do código?  | Primeiro: temos que ter 2 classes ou mais onde uma será a Classe Principal - aquela que irá compor as classes dependentes - e a(s) outra(s) será a Classe Dependente - aquela que irá ser inicializada pela Classe Principal;  Segundo: temos que fazer com que o elemento dependente seja inicializado somente dentro da Classe Principal;  Terceiro: qualquer modificação ou eliminação sobre a Classe Principal iria resultar em mudanças ou inutilização sobre as Classes Dependentes; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |