Programação Orientada a Objetos

Introdução à Análise Orientada a Objetos (AOO)

Cristiano Lehrer, M.Sc.

ybadoo

Processo de Desenvolvimento de Software

- Um **processo de software** mostra os vários estágios do desenvolvimento de software.
  - Um **processo iterativo** é uma estratégia iterativa e incremental para desenvolvimento de software.
  - Outro modo de pensar a respeito do processo é como uma estratégia 'evolutiva'.
  - Cada iteração aperfeiçoa e elabora gradualmente um produto básico em um produto amadurecido.
  - Os processos iterativos precisam ser cuidadosamente monitorados para se garantir que eles não sejam simplesmente reduzidos a 'cavar' uma solução.
    - A AOO e o POO fornecem tal verificação de sanidade.

Análise Orientada a Objetos (AOO) (1/4)

- **Análise Orientada a Objetos** é um processo que usa uma estratégia orientada a objetos para ajudá-lo a entender o problema que está tentando resolver.
  - No final da análise, você deverá entender o domínio do problema e seus requisitos em termos de classes e interações de objetos.
- **Sistema** é o termo da AOO para um conjunto de objetos que interagem.
  - Você pode dizer que esses objetos constituem um sistema ou modelo do problema.

Análise Orientada a Objetos (AOO) (2/4)

- Os **requisitos** são os recursos ou características que o sistema deve ter para resolver determinado problema.
- Um modo de descobrir esses usos é através de análise de casos de uso.
  - Através da análise você definirá vários casos de uso.
  - Um caso de uso descreve como um usuário vai interagir com o sistema.

ybadoo

Análise Orientada a Objetos (AOO) (3/4)

- **Análise de casos de uso** é o processo de descoberta de casos de uso através da criação de cenários e histórias com usuários em potencial ou existentes de um sistema.
  - Um caso de uso descreve a interação entre o usuário do sistema e o sistema – como o usuário utilizará o sistema do seu próprio ponto de vista.
  - Para definir seus casos de uso, você deve:
    - Identificar os atores.
    - Criar uma lista preliminar de casos de uso.
    - Refinar e nomear os casos de uso.
    - Definir a sequência de eventos de cada caso de uso.
    - Modelar seus casos de uso.

Análise Orientada a Objetos (AOO) (4/4)

- Um **ator** é tudo que interage com o sistema.
  - Pode ser um usuário humano, outro sistema de computador ou um chimpanzé.
- Um **cenário** é uma sequência ou fluxo de eventos entre o usuário e o sistema.
  - Condições prévias são aquelas condições que devem ser satisfeitas para que um caso de uso comece.
  - Condições posteriores são os resultados de um caso de uso.

ybadoo

Projeto Orientado a Objetos (POO) (1/3)

- **Projeto Orientado a Objetos** é o processo de construir o modelo de objeto de uma solução.
  - Dito de outra maneira, POO é o processo de dividir uma solução em vários objetos constituintes.
- O **modelo de objeto** é o projeto dos objetos que aparecem na solução de um problema.
  - O modelo final de objeto pode conter muitos objetos não encontrados no domínio.
  - O modelo de objeto descreverá as várias responsabilidades, relacionamentos e estrutura do objeto.

ybadoo

Projeto Orientado a Objetos (POO) (2/3)

- O POO continua o trabalho da AOO, pegando o domínio e transformando-o em uma solução para seu problema.
  - Através do processo de POO, você pega seu modelo de domínio e constrói o modelo de objetos de sua solução.
  - O modelo de objetos descreve os aspectos arquitetonicamente significativos de seu sistema, como a estrutura e os relacionamentos dos objetos – como os objetos se encaixam.
  - No final do POO, você deverá ter uma boa ideia do que implementará no código.

Projeto Orientado a Objetos (POO) (3/3)

- A grosso modo, existem cinco passos iterativos que você pode seguir, enquanto realiza o POO:
  - Gerar a lista de objetos inicial.
  - Refinar as responsabilidades de seus objetos.
  - Desenvolver os pontos de interação.
    - Um ponto de interação é qualquer lugar onde um objeto use outro.
  - Detalhar os relacionamentos entre os objetos.
  - Construir seu modelo.

ybadoo

Unified Modeling Language (UML) (1/3)

- Uma **linguagem de modelagem** é uma notação gráfica para descrever um projeto de software.
  - A linguagem também inclui várias regras para distinguir entre desenhos corretos e incorretos.
  - São essas regras que tornam a UML uma linguagem de modelagem e não apenas um punhado de símbolos para desenho.
- Uma **metodologia** define um procedimento para projetar software.
  - As linguagens de modelagem capturam esse projeto graficamente.

ybadoo

Unified Modeling Language (UML) (2/3)

- A UML é uma linguagem de modelagem padrão.
  - A UML consiste na notação para descrever cada aspecto de um projeto de software.
- Uma metodologia ou processo descreve como projetar software.
  - Uma metodologia frequentemente contém uma linguagem de modelagem.

ybadoo

Unified Modeling Language (UML) (3/3)

- A UML apresenta um rico conjunto de ferramentas de modelagem.
  - Como resultado, existem muitas informações que você pode colocar em seus modelos.
  - Cuidado para não tentar usar toda a notação existente ao modelar.
  - Use apenas notação suficiente para transmitir seu projeto.
  - Lembre-se sempre de que o objetivo de seu modelo é transmitir seu projeto.
  - Faça o que precisar para transmiti-lo e fique com isso.

ybadoo

Modelando uma Classe (1/4)

- Visibilidade
  - - privado
  - # protegido
  - + público

Nome da classe
Atributos da classe
Métodos da classe

BankAccount
- balance : double
+ depositFunds(amount : double) : void + getBalance() : double # setBalance(amount : double) : void + withdrawFunds(amount : double) : double

Modelando uma Classe (2/4)

- Dicas para a modelagem eficiente:
  - Sempre faça a você mesmo a pergunta “o que eu estou tentando transmitir?”.
    - A resposta o ajudará a decidir exatamente o que você precisa modelar.
  - Sempre faça a você mesmo a pergunta “para quem eu estou tentando transmitir a informação?”.
    - A resposta ditará o modo como você vai modelar.
  - Sempre tente produzir o modelo mais simples que ainda tenha êxito em transmitir seu projeto.

ybadoo

Modelando uma Classe (3/4)

- Dicas para a modelagem eficiente:
  - Não fique preso à linguagem de modelagem.
    - Embora você não deva ser vago demais na semântica, não deve deixar que o fato de seguir a notação perfeitamente o impeça de concluir seus diagramas.
    - Os perigos da paralisia ao modelar são reais – especialmente quando você está começando.
    - Não se preocupe se seu modelo não estiver 100% perfeito.
    - Preocupe-se somente se seu modelo não transmite corretamente o projeto.

Modelando uma Classe (4/4)

- Dicas para a modelagem eficiente:
  - Finalmente, lembre-se de que a UML (ou qualquer outra linguagem de modelagem) é simplesmente uma ferramenta para ajudá-lo a transmitir o projeto.
    - Ela não é um instrumento em si mesma.
    - No final do dia, você ainda precisará produzir código.

ybadoo

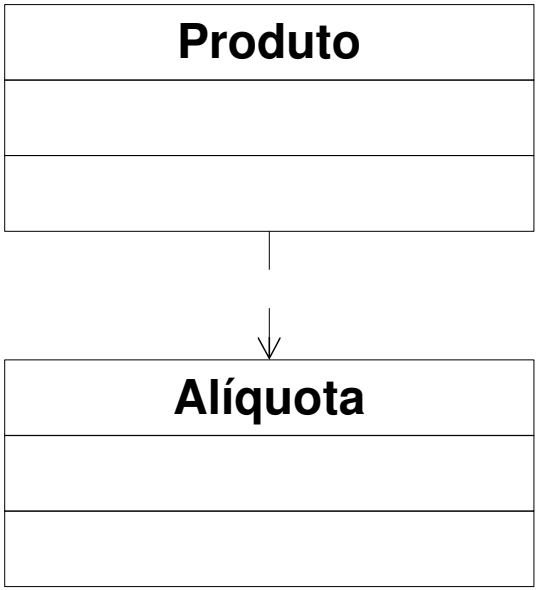
Modelando um Relacionamento de Classe (1/2)

- A UML reconhece três tipos de alto nível de relacionamento de objetos:
  - Dependência
    - Em um relacionamento de dependência, um objeto é dependente da especificação de outro objeto.
      - Se a especificação mudar, você precisará atualizar o objeto dependente.
  - Associação
    - Uma associação indica que um objeto contém outro objeto.
      - Nos termos da UML, quando se está em um relacionamento de associação, um objeto está conectado a outro.
  - Generalização
    - Um relacionamento de generalização indica um relacionamento entre o geral e o específico.
      - Se você tem um relacionamento de generalização, então sabe que pode substituir uma classe filha pela classe progenitora.

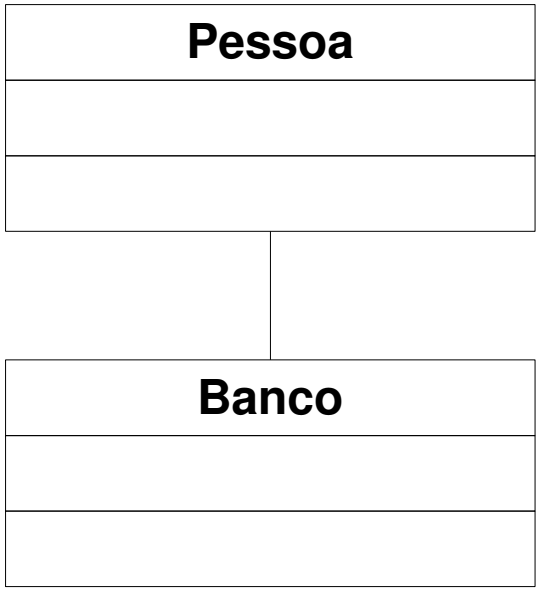
ybadoo

Modelando um Relacionamento de Classe (2/2)

Dependência



Associação



Generalização