Atividade 1: Modelagem Orientada a Objetos

POO - BSI - Ifes Serra

29 de outubro de 2024

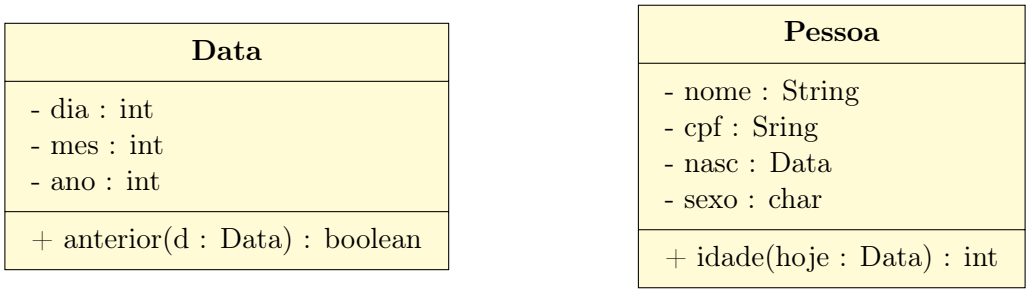
1 Diagrama de Classes UML

A Linguagem de Modelagem Unificada (UML) ajuda a modelar sistemas de diversas maneiras. Um dos tipos mais populares na UML é o **diagrama de classes**, que descreve o que deve estar presente no sistema a ser modelado incluindo as classes, seus atributos, suas operações e seus relacionamentos com outras classes. A UML foi criada para padronizar a descrição dos diagramas OO. Um diagrama de classes permite ilustrar modelos de dados para sistemas de informação (simples ou complexos), entender melhor a visão geral dos esquemas de uma aplicação, expressar visualmente as necessidades específicas de um sistema, e fornecer uma descrição do sistema independente de implementação.

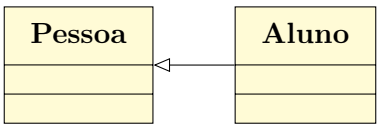
Uma classe no diagrama é representada por um retângulo dividido em três partes:

- A parte superior contém o nome da classe e é sempre necessária.
- A parte do meio contém os atributos da classe. Cada atributo é exibido em uma linha separada. Cada linha possui o nome do atributo e seu tipo, separados por um ":".
- A parte inferior inclui as funcionalidades da classe (métodos). Cada operação é exibida em uma linha separada. O método é descrito por seu nome e sua assinatura (lista de parâmetros entre parênteses e separados por vírgula, e o tipo do retorno do método após o ":").

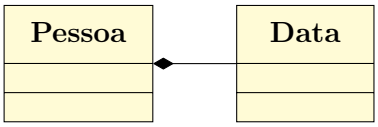
Atributos e métodos possuem modificadores de acesso, que definem sua visibilidade: público (+), privado (-) e protegido (#). Métodos e atributos estáticos são representados sublinhados. Exemplo:



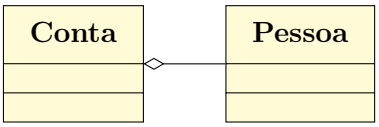
Diagramas de classe também podem exibir o relacionamento entre as classes do sistema. Nesta atividade, iremos considerar os seguintes tipos de relacionamentos:



1. A seta entre as classes Aluno e Pessoa indica **herança** entre elas. Neste caso, Aluno herda de Pessoa. Um objeto da classe Aluno também "é um" objeto da classe Pessoa, mas pode conter atributos e métodos próprios.

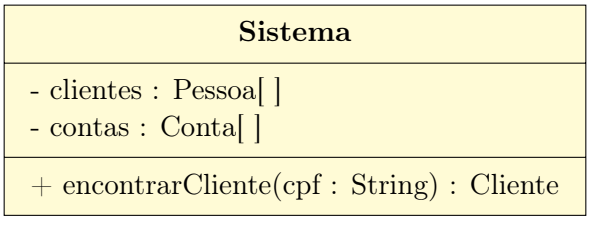


2. O losango fechado entre Pessoa e Data indica uma relação de **composição**. Neste caso, podemos dizer que a classe Pessoa "tem" um objeto da classe Data (a data em que a pessoa nasceu). Por ser uma composição, o objeto da classe **Data** vai ser instanciado no momento em que a avaliação for criada e existirá apenas ali dentro do objeto da classe **Pessoa**. Neste exemplo, o sistema não precisa armazenar uma Data sem que ela esteja associada ao uma Pessoa específica. Quando o objeto da classe Pessoa for destruído, o objeto com sua data de nascimento será destruído junto. Podemos dizer que a data é uma "parte da" Pessoa.



3. O losango aberto entre Conta e Pessoa indica uma relação de **agregação**. Novamente, podemos dizer que a classe Conta "tem" um objeto da classe Pessoa (o titular da conta). Entretanto, neste caso, o objeto Pessoa não será instanciado apenas quando a Conta for criada. A Pessoa pode existir no sistema sem a Conta existir. Quando a conta é criada, as pessoas já existiam no sistema, e a conta faz apenas uma referência a uma delas como sendo o titular dessa conta. Quando o objeto da classe Conta for destruído, seu titular continuará existindo no sistema (ele pode ter ou vir a ter outras contas no banco, por exemplo).

Por fim, neste trabalho, usaremos uma notação única para representar um conjunto de objetos (independente de este conjunto ser armazenado como um array, uma lista, etc). No exemplo abaixo, por exemplo, a classe sistema possui um conjunto de pessoas e um conjunto de contas:



2 O Problema: Pedidos na Cantina

Você está cursando BSI no Ifes e sempre sente muita fome durante as aulas. Entretanto, sua fome é menor do que sua disposição de caminhar até a cantina para buscar o lanche. Geralmente, você fica com preguiça de ir até lá e acaba passando fome, o que te atrapalha a se concentrar nas aulas. Você decidiu, então, criar um aplicativo para facilitar sua vida e ainda ganhar dinheiro com isso. O aplicativo permite que um aluno faminto e ~~preguiçoso~~ ocupado

faça um pedido na cantina do Ifes e que algum outro aluno ~~precisando de dinheiro~~ prestativo vá lá buscar o pedido pra ele. O aluno que fez o pedido paga R$1,00 de taxa de entrega (80% vai para o aluno entregador e o restante para você, dono do sistema).

Inseguro com o desenvolvimento do aplicativo, você decidiu pedir ajuda ao professor de POO. Sua tarefa inicial é mostrar pra ele o diagrama de classes do aplicativo. Mas, para não passar vergonha com o professor, você quer se esforçar para que utilize corretamente, sempre que pertinente, as boas práticas de orientação a objetos, como **herança**, **reescrita de métodos**, **sobrecarga de métodos**, **encapsulamento**, reuso de código, etc.

3 As Classes e Atributos

Seu sistema deve ser capaz de representar dois tipos de usuários: alunos e administradores. Todos eles possuem cpf, nome e senha. Além dessas informações, um aluno possui também um saldo (valor depositado por ele que pode ser utilizado como crédito para realizar seus pedidos). Quando ele faz um pedido, o valor do pedido é descontado deste saldo. Quando ele realiza a entrega de um pedido, sua comissão é adicionada a esse mesmo saldo. Já o administrador, ao invés do saldo, armazena seu endereço de email.

Cada produto vendido pela cantina possui as seguintes informações: código, nome, descrição, quantidade em estoque e valor unitário.

Cada pedido feito no sistema armazena um código, um conjunto de itens no carrinho, um aluno cliente (que realizou o pedido), um aluno entregador (que fará a entrega do pedido), o local de entrega e a data do pedido. Além disso, há um valor indicando se ele já foi entregue ou não. O aluno entregador não é atribuído inicialmente assim que o pedido é realizado.

Cada item no carrinho identifica um produto, uma quantidade e uma observação (por exemplo: "trazer catchup").

Os locais de entrega representam as salas do Ifes e possuem um identificador, um bloco, uma sala e um andar (a sala "902T", por exemplo, é formada pelo bloco "9", sala "02", andar "T").

Por fim, a classe principal do sistema armazena cinco conjuntos: os produtos cadastrados, os pedidos já realizados, os alunos no sistema, os usuários administradores e as salas onde os pedidos podem ser entregues.

4 As funcionalidades

Todo usuário possui métodos para validar seu acesso (dada sua senha) e alterar sua senha (dada a senha atual e a nova).

Um aluno, especificamente, possui métodos para inserir créditos em seu saldo (dado um valor a ser inserido), e retirar créditos de seu saldo. Este segundo deve indicar se foi possível realizar a retirada de créditos (não é possível fazer um pedido se o aluno não tiver crédito suficiente). O crédito nunca pode ser negativo.

Os usuários administradores possuem um método mais especializado para validação do acesso. Além da lógica já existente no método comum a todos os usuários, ele solicita a digitação de um código de acesso enviado para seu email a cada tentativa de login.

Um pedido deve possuir um método para calcular o valor total do pedido, incluindo todos os itens com suas quantidades especificadas no carrinho. Há ainda um método para indicar se o produto está disponível para entrega no sistema (ou seja, ainda não foi entregue e não possui um entregador associado a ele). Por fim, há um método para atribuir um entregador a ele (dado um aluno).

Uma data deve possuir um método que compare-a com uma outra data recebida como parâmetro. Este método vai retornar um número inteiro (-1, se a data em questão for anterior à outra, 0 caso sejam iguais e 1 caso seja posterior). Além disso, deve possuir um método para exibi-la no formato "20/12/2024".

A classe principal, por fim, possui as seguintes funcionalidades:

- Login: Faz a leitura dos dados do usuário, chama a validação apropriada dependendo do tipo de usuário e depois exibe seu menu.
- Menu: Dado um usuário, exibe as opções disponíveis no sistema para ele. Se o usuário passado como parâmetro for um aluno, sua implementação deve ser diferente da implementação para um administrador.
- Filtrar pedidos: Retorna um conjunto de pedidos, dependendo do parâmetro recebido. Se receber duas datas, retorna os pedidos realizados entres essas duas datas. Se receber um aluno, retorna todos os pedidos já realizados por ele. Se receber um booleano, retorna todos os pedidos ainda disponíveis ou não para entrega.

As demais funcionalidade serão especificadas na próxima fase do sistema.

5 O diagrama de classes UML

Nesta primeira atividade, você deve criar ***apenas*** o diagrama de classes UML do sistema descrito neste documento, contendo todas as classes, atributos, métodos e relacionamentos entre as classes.

É importante que você utilize corretamente, sempre que pertinente, as boas práticas de orientação a objetos, como **herança**, **reescrita de métodos**, **sobrecarga de métodos**, **encapsulamento**, reuso de código, etc.

Não é necessário incluir os construtores das classes no diagrama, nem se preocupar com a leitura dos dados por enquanto.

Você pode utilizar programas específicos para modelagem OO, editores de imagens ou até mesmo desenhá-lo num papel. Deve ser enviado apenas um arquivo único contendo o diagrama completo (pdf, jpeg ou png).

6 Observações

1. Esta atividade vale 20 pontos e deve ser entregue até 05/09.
2. Deve ser enviado apenas um PDF único ou um arquivo único com a imagem do diagrama de classes UML de sistema, considerando as boas práticas de POO (como reuso de código, por exemplo, sempre que possível).
3. A atividade pode ser feita em grupos de até **dois** integrantes.
4. Atividades entregues após o prazo serão automaticamente rejeitadas.
5. Atividades consideradas plágio terão nota 0 para quem copiou e para quem forneceu a atividade, e serão enviadas ao Conselho de Ética.
6. A atividade deve ser enviada na sala da disciplina do AVA.
7. Em caso de dúvidas na especificação da atividade ou na própria atividade, contate-me em sala de aula.