Atividade 1: Modelagem Orientada a Objetos

POO - BSI - Ifes Serra

22 de abril de 2025

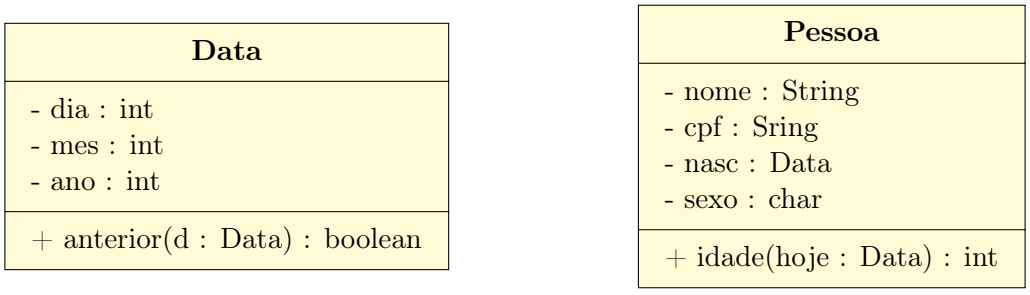
1 Diagrama de Classes UML

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.

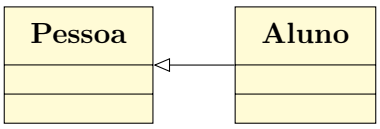
Uma classe no diagrama é representada por um retângulo dividido em três partes:

- 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 ":").

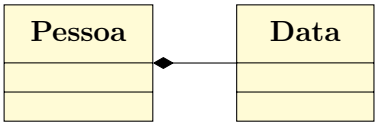
Atributos e métodos possuem modificadores de acesso, que definem sua visibilidade: público (+), privado (-) e protegido (#). Métodos e atributos estáticos são representados sublinhados. Exemplo:



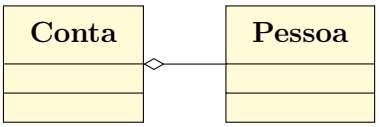
Diagramas de classe também podem exibir o relacionamento entre as classes do sistema. Nesta atividade, iremos considerar os seguintes tipos de relacionamentos:



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.

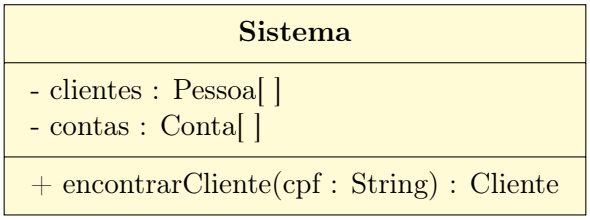


2. O losango fechado entre Pessoa e Data indica uma relação de **composição**. Neste caso, podemos dizer que a classe Pessoa "tem" um objeto da classe Data (a data em que a pessoa nasceu). Por ser uma composição, o objeto da classe **Data** vai ser instanciado no momento em que a avaliação for criada 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: Sistema Acadêmico

Cansado das instabilidades do Sistema Acadêmico da sua faculdade, você decidiu implementar um novo e tentar vendê-lo. Inicialmente, o Sistema vai armazenar dados sobre alunos, professores, turmas e atividades.

Sua tarefa inicial é criar o diagrama de classes do sistema acadêmico para mostrar ao seu professor de POO. 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: um **Aluno** e um **Professor**. Todos eles possuem *nome* e *cpf*. Além dessas informações, um aluno possui também uma *matrícula* e um professor possui um *salário*.

Existem também dois tipos de avaliação: **Prova** e **Trabalho**. Ambos possuem um *nome* (exemplo: "Trabalho 1", "Prova 3"), uma *data* de aplicação e o *valor* da atividade (a nota máxima que o aluno pode tirar nesta atividade).

As provas, especificamente, possuem também um número n de questões e um conjunto de objetos do tipo **AlunoProva**. Cada um desses objetos armazena um *aluno* que fez essa prova e sua *nota* em cada uma das n questões.

Já os trabalhos possuem também um número máximo m de integrantes e um conjunto de objetos do tipo **GrupoTrabalho**. Cada um desses objetos armazena os *alunos* que fizeram esse trabalho em grupo e a *nota* total do grupo.

Todas as datas no sistema devem ser modeladas como uma classe contendo 3 números inteiros (*dia*, *mês* a *ano*).

Uma **Turma** possui *nome*, o *ano* e o *semestre* em que foi ofertada (por exemplo, semestre 2 do ano 2024), o *professor* responsável, o conjunto de *alunos* matriculados e o conjunto de *avaliações* que o professor aplicou.

Por fim, a classe **Sistema** deve armazenar o conjunto de *alunos*, de *professores* e de *turmas*.

4 As funcionalidades

Cada atividade deve possuir um método que, dada a matrícula de um aluno, deve ser capaz de retornar a nota do aluno nessa atividade (independente de essa atividade ser uma prova ou um trabalho). Se for necessário, crie métodos auxiliares nas classes AlunoProva e GrupoTrabalho para que essas notas sejam calculadas.

Uma turma possui um método chamado *notas*. Se ele receber um aluno como parâmetro, ele imprime a nota do aluno em cada atividade dessa turma. Caso não receba nada, ele exibe a nota de cada aluno matriculado na disciplina em cada atividade da disciplina e, ao final, a média geral da turma.

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

- Criar novo professor e inseri-lo no conjunto de professores.
- Criar novo aluno e inseri-lo no conjunto de alunos.
- Criar nova turma e inseri-la no conjunto de turmas.
- Listar as médias gerais de cada turma cadastrada.

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 29/04.
2. Deve ser enviado um único arquivo com a imagem do diagrama de classes UML do 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. Lembre-se de identificar os integrantes do grupo.
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.