Disciplina: Programação Orientada a Objetos
Professor: Antonio Henrique Pinto Selvatici

AULA 2 – Classes e objetos

ROTEIRO:

- Rever programa da aula anterior
- Introdução a classes (atributos e métodos)
    a. Atributos
    b. Métodos
- Encapsulamento
- Diagrama de classes
- Exercício: especificação e implementação do sistema bancário

1. Programa da aula anterior

2. Introdução a classes

Os objetos são formas de descrever componentes específicos de um sistema. Tudo que pode ser considerado como uma unidade, pode também ser um objeto.

Os objetos podem ser categorizados, ou seja, há diferentes categorias de objetos. No nosso exemplo, temos objetos que representam contas bancárias, e objetos que representam clientes. Uma classe descreve — de maneira abstrata — todos os objetos de um tipo particular. Assim, no exemplo acima podemos criar a classe "Conta Bancária" e a classe "Cliente". Podemos também ter a classe Banco, que conteria as definições relativas ao objeto que representa o banco. E, cada objeto que representa um cliente específico, com seu nome e CPF, é uma instância da classe "Cliente".

É importante sabermos que em uma linguagem orientada a objetos tradicional, em geral não codificamos objetos específicos, mas sim suas classes. Ou seja criamos especificações de como os objetos de uma classe podem ser, e posteriormente usamos essas especificações para criar os objetos (instanciação)

Fazem parte dessa especificação do objeto (classe) informar seus métodos e atributos. Os métodos são as ações que os objetos de uma classe podem realizar, enquanto os atributos são valores que cada objeto guarda consigo de forma a parametrizar essas ações. Os atributos são também conhecidos como variáveis de instâncias e guardam o estado do objeto. Os métodos são expressos na forma de funções que recebem parâmetros e podem retornar um valor, além de terem acesso às variáveis correspondentes a uma instância (objeto) específico. Ao conjunto formado pelo nome do método, seus tipos de parâmetros de entrada e o tipo da saída chamamos de **assinatura do método**.

Por exemplo: dado um problema, modelar a classe para resolver esse problema.

- A Escola de Natação Aqua precisa de um sistema que efetue alguns cálculos para gerenciar as atividades de seus alunos
- As informações relevantes para cada aluno são: nome, sexo, altura, peso e ano de nascimento
- Com base no ano de nascimento do aluno, deseja-se saber a idade atual do aluno
- Para iniciar as atividades físicas é necessário saber o peso ideal de cada aluno
- Fórmulas:
    - Masculino: (72.7 * altura) – 58
    - Feminino: (62.1 * altura) – 44.7
- Finalmente, é necessário gerar um relatório com os dados de cada aluno cadastrado
- Pede-se para identificar:
    - classe
    - variáveis de instância (atributos)
    - Métodos (comportamento) e suas assinaturas
    - Implemente em Java a classe e o método para cálculo da altura ideal

3. Encapsulamento

A eficácia da programação orientada a objetos depende muito ideia de **encapsulamento**. Isso significa que as informações (variáveis) necessárias para executar uma tarefa estão encapsuladas (agrupadas) junto às funções que de fato a executam, ou seja, estão encapsuladas no mesmo objeto. Isso diminui o acoplamento entre as partes do código, já que não é mais necessário acessar diretamente as variáveis necessárias para realizar uma tarefa e passá-las como parâmetros; agora basta saber qual objeto que é capaz de executar essa tarefa e pedir para que seja executada. Para pensar: no exercício anterior, como seria o programa que calcularia o peso ideal de cada aluno na programação estruturada e na POO?

Para reforçar a ideia de encapsulamento, as linguagens orientadas a objeto tradicionais trazem a possibilidade de **ocultação de informações**. Isso significa que cada atributo ou método de uma classe tem a sua visibilidade especificada, ou seja, determina que objetos podem acessá-los. Para indicar a visibilidade, de um atributo ou método, usamos um modificador do tipo public, private ou protected, ou ainda sem modificador. Veja a visibilidade do método ou atributo a depender do modificador.

Modificador	Classe	Pacote	Subclasse	Globalmente
Public	*Sim*	*Sim*	*Sim*	*Sim*
Protected	*Sim*	*Sim*	*Sim*	*Não*
Sem Modificador (Padrão)	*Sim*	*Sim*	*Não*	*Não*
Private	*Sim*	*Não*	*Não*	*Não*

Por enquanto, sempre que um atributo precisar ser acessado ou alterado de fora da classe, vamos manter a visibilidade como pública.

4. Exercício

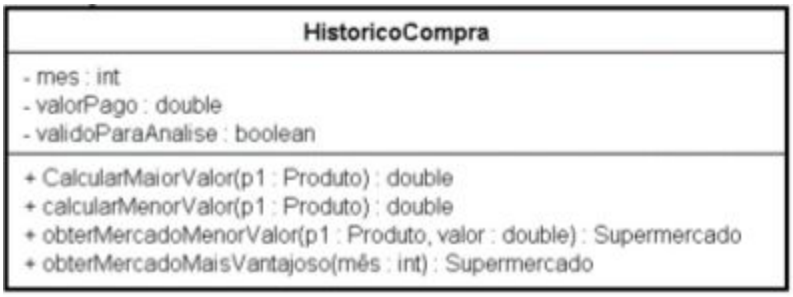
Transformar o sistema bancário desenvolvido na última aula em um sistema com programação orientada a objetos. Vamos pensar nas classes necessárias, quais os métodos e atributos mais adequados, focando no encapsulamento dos métodos e atributos.

5. Diagrama de classes

A UML define uma forma de visualizarmos a principais classes a serem implementadas no sistema. Dentro da chamada visão lógica, que explica como os diferentes componentes se conectam, temos o diagrama de objetos e o diagrama de classes. Enquanto o diagrama de objetos mostra uma "fotografia" do sistema, exemplificando que objetos poderiam estar instanciados e quais os valores de seus atributos, o diagrama de classes mostra as classes que devem ser implementadas, quais são seus métodos e atributos e como elas se associam

Uma classe no diagrama de classes é representada por três divisões: a do nome da classe, em cima, a dos atributos no meio e a dos métodos embaixo. Junto aos métodos e atributos é indicada a visibilidade de cada um com um símbolo de "+" para visibilidade pública e "-" para visibilidade privada.

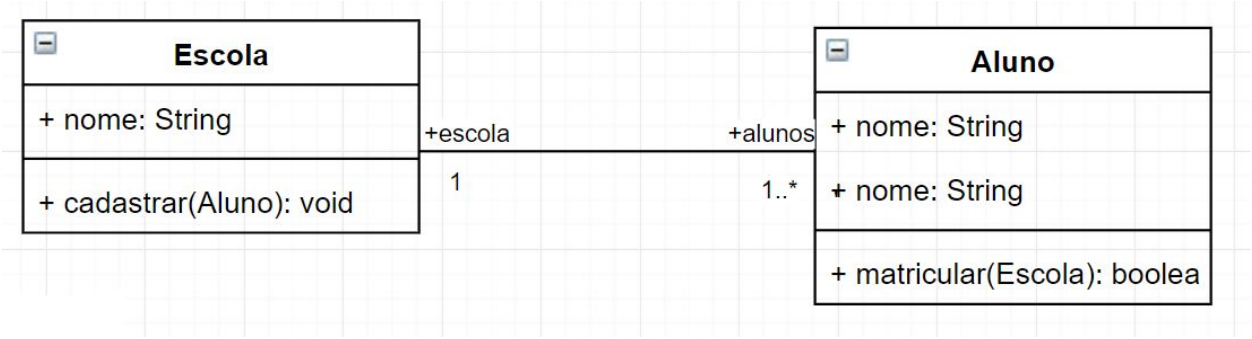
Os atributos e parâmetros dos métodos são indicados da forma `<nome>: <tipo>`, onde `<nome>` é o nome do atributo/método, e `<tipo>` é seu tipo ou classe.



Exemplo de representação da classe **HistoricoCompra**.
Associação

Quando os objetos de uma classe necessitam se comunicar com objetos de outra (ou até da mesma) classe, temos uma associação entre elas. Nesse caso, indicamos a associação com um traço ligando as classes. Essa ligação pode ainda indicar os nomes de papéis em cada extremidade, a cardinalidade, direção e restrições.

Vamos analisar o exemplo das classes Escola e Aluno. Elas precisam se conversar para atingir o objetivo do sistema (cadastro dos alunos e gerenciamento das aulas). Enquanto o aluno está matriculado apenas em uma escola, a escola pode ter um ou mais alunos. O uso de papéis evidencia, em geral, que essas informações estão armazenadas nos próprios objetos na forma de atributos.



Exemplo de associação com cardinalidade e papéis

Como exercício, vamos criar o diagrama de classes do nosso sistema bancário, denotando os métodos, atributos, visibilidade, papéis e cardinalidade.