Classes e Objetos
Abstração e Encapsulamento

Carlos Olarte

8 de Agosto de 2019

Classes

Classes

- Abstração para agrupar objetos comuns que têm o mesmo comportamento.
- Descrevem de maneira abstrata o comportamento dos objetos.

Objetos

Objetos são instâncias de uma classe:

- Encapsulam um estado
- Respondem às mensagens com a execução de um método.

Os objetos de uma mesma classe compartem o comportamento definido pela sua classe.

Membro vs. valor do membro

- Membro / Atributo: algo que todo objeto de uma determinada classe possui.
- Valor do membro: valor do membro para uma determinada instância da classe.

Toda Pessoa tem o membro `nome`. O objeto `p1` do tipo `Pessoa` tem o valor do seu atributo nome igual a "Ana" e o objeto `p2` tem o valor igual a "João".

Métodos e Construtores

- Construtores: inicializam os atributos da classe.
- Métodos: definem o comportamento da classe: as operações que um objeto pode fazer.
- Dentro de um método, em Python, `self` é a referência ao objeto que invocou o método (uma auto-referência ao próprio objeto).

Objetivo

Nesta aula aprenderemos:

- Aprofundar no conceito de encapsulamento na POO.
- Implementar diagramas de classe simples (UML).
- Utilizar arrays/listas em Python.

Os Quatro Pilares de POO

- Abstração: capacidade de abstrair informações do mundo real.
- Encapsulamento: capacidade de encapsular/esconder informações (dados) nos objetos.
- Herança
- Polimorfismo

Encapsulamento

- A forma como o objeto implementa o seu comportamento deve ser separada do mecanismo de troca de mensagens entre os objetos
- Os membros que não devem ser utilizados fora da classe devem estar escondidos
- Este mecanismo facilita a manutenção e reaproveitamento de código

Encapsulamento

- Os usuários da classe só podem acessar os membros e métodos públicos da classe.
- **Importante**: usuário do sistema não é o usuário da classe
- POO é comumente utilizada no desenvolvimento de bibliotecas
- O usuário da classe é quem utiliza uma biblioteca POO
- As classes devem expôr o mínimo necessário para serem utilizadas
- Similar a funções: os dados mínimos que elas precisam são os parâmetros

Encapsulamento

Modificadores de acesso

- Em POO, existe o conceito de modificadores de acesso, que valem para membros e métodos:
  - Público: o membro/método pode ser acessado/chamado de qualquer lugar
  - Privado: o membro/método só pode ser acessado/chamado de dentro da definição da classe
  - Protegido: utilizado junto do mecanismo de herança
- A implementação dos modificadores é um mecanismo dependente da linguagem. Em C++ e Java, por exemplo, as palavras chaves “‘public“‘ e “‘private“‘ devem ser usadas para esta função

Exemplo Estacionamento

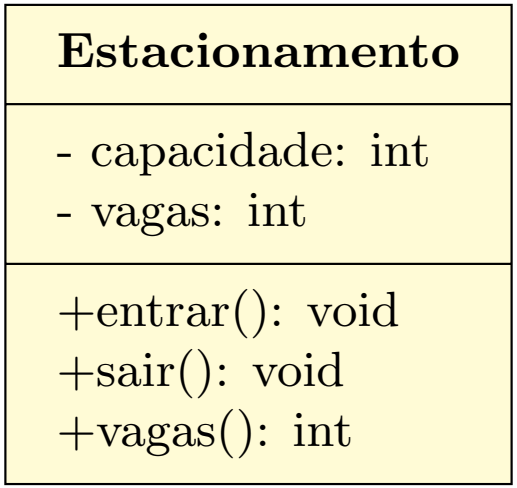
Um estacionamento tem capacidade para um número $n > 0$ de vagas. Devemos controlar quantos carros estão dentro do estacionamento. Os carros podem entrar só se há vagas disponíveis.

Nesse sistema podemos identificar:

- Classes: Carros e o Estacionamento. Por enquanto não vamos modelar os carros.
- Atributos: Um estacionamento possui os atributos *número de vagas* e *capacidade máxima*.
- Métodos: Os carros podem *entrar* e *sair*. Além disso, podemos consultar o número de vagas disponíveis.

Notação UML

Diagrama de classes



Com relação à notação:

- "-" denota um membro de classe *privado*.
- "+" um membro *público*
- "#" um membro *protegido*.
- *vagas(): int* significa que o método "vagas" não tem parâmetros e o método retorna um inteiro.

`https://www.draw.io/`

Questão

Os usuários da classe `Estacionamento` deveriam modificar o atributo `vagas` ?

Questão

Os usuários da classe `Estacionamento` deveriam modificar o atributo `vagas` ?

Não! O valor do atributo vagas não pode ser modificado direitamente (só utilizando os método entrar/sair).

Encapsulamento

- Esconder os membros de uma classe.
- Esconder como funcionam as rotinas (métodos) da classe.
- O encapsulamento facilita o reaproveitamento de código.
- Alterar código resulta mais simples.
- Os usuários da classe só podem acessar os atributos/métodos públicos da classe.

En geral, as classes devem expor o mínimo necessário para serem utilizadas.

Encapsulamento e atributos privados em Python

Ver Jupyter-notebook