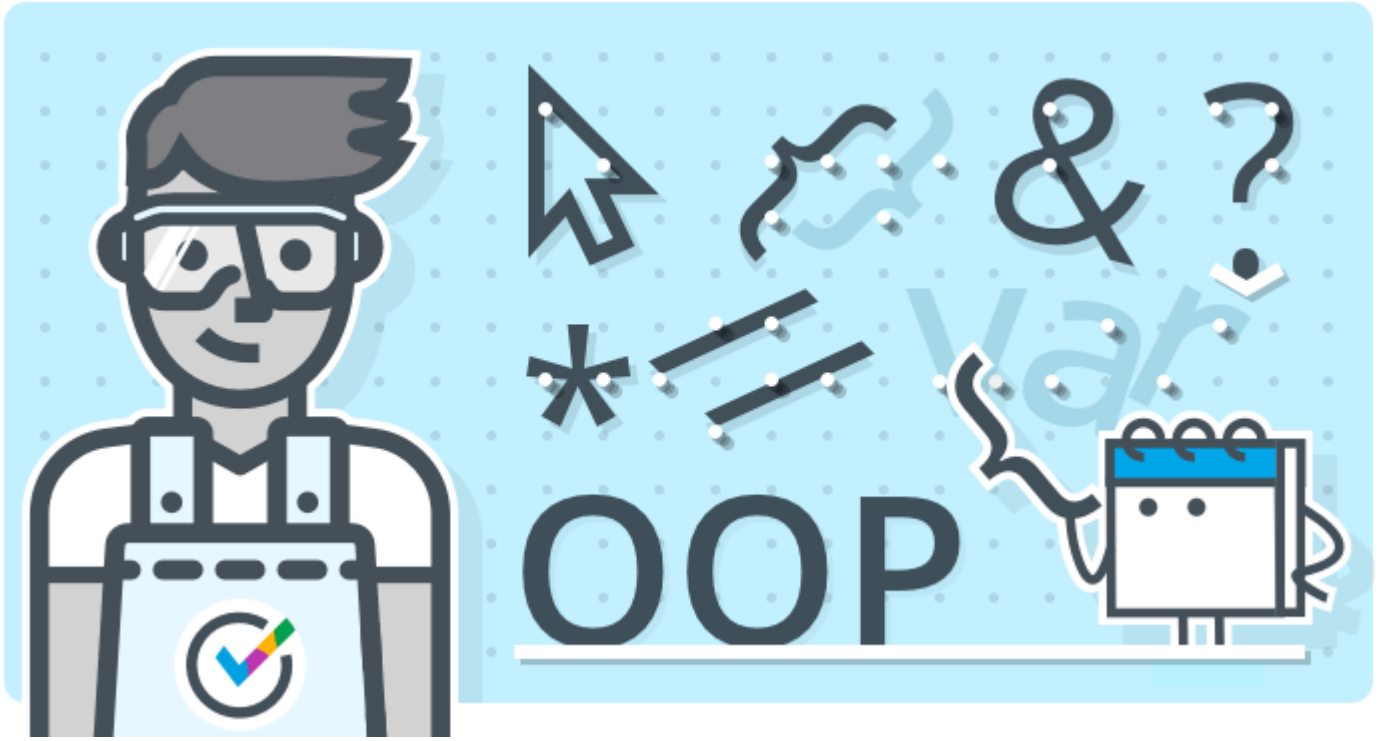
Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

Revisão de Programação Orientada a Objetos



Revisão de Programação Orientada a Objetos

O mundo da orientação a objetos apresenta o conceito de **objetos** que têm **atributos** e **métodos**.

Estes **métodos** são responsáveis pela manipulação dos **atributos**.

Revisão de Programação Orientada a Objetos

O mundo da orientação a objetos apresenta o conceito de **objetos** que têm **atributos** e **métodos**.

Estes **métodos** são responsáveis pela manipulação dos **atributos**.

```python
class Carro:

    def __init__(self: object, is_sedan: bool = False):
        self.__is_sedan = is_sedan
        self.__velocidade = 0
        self.__motorista = None

    def dirigir(self: object, motorista: Pessoa):
        self.__motorista = motorista
        self.acelerar(1)

    def acelerar(self: object, velocidade: int):
        self.__velocidade += velocidade

    def parar(self: object):
        self.__velocidade = 0
```

```python
from datetime import datetime

class Pessoa:

    def __init__(self: object, nome: str):
        self.__nome = nome
        self.__nascimento = datetime.now()
```

```
(ppp)            @darkstar secao02]$ python
Python 3.9.1 (default, Dec 21 2020, 14:28:49)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from aula02 import Pessoa, Carro
>>> angelina = Pessoa('Angelina')
>>> fusca = Carro()
>>> fusca.dirigir(angelina)
>>> fusca.__dict__
{'_Carro__is_sedan': False, '_Carro__velocidade': 1, '_Carro__motorista': Angelina}
>>>
```

No exemplo acima temos uma classe Carro com os atributos ‘is_sedan’ e ‘velocidade’ e os métodos ’acelerar’ e ‘parar’.

Revisão de Programação Orientada a Objetos

Classes:

- Ajudam os desenvolvedores a representar entidades do mundo real;
- Definem os objetos com atributos e comportamentos (métodos);
- Classes possuem construtores (método especial) que proporcionam o estado inicial para os objetos;
- Classes são como templates (modelos), portanto, podem ser facilmente reutilizadas;

```python
from datetime import datetime

class Pessoa:

    def __init__(self, nome):
        self.__nome = nome
        self.__nascimento = datetime.now()
```

```python
class Carro:

    def __init__(self, is_sedan=False):
        self.__is_sedan = is_sedan
        self.__velocidade = 0
        self.__motorista = None

    def dirigir(self, motorista):
        self.__motorista = motorista
        self.acelerar(1)

    def acelerar(self, velocidade):
        self.__velocidade += velocidade

    def parar(self):
        self.__velocidade = 0
```

Revisão de Programação Orientada a Objetos

Atributos:

- Definem as características que nos ajudam a modelar/mapear o objeto através da classe;
- São conhecidos como membros-dados na orientação a objetos;

```python
from datetime import datetime

class Pessoa:

    def __init__(self, nome):
        self.__nome = nome
        self.__nascimento = datetime.now()
```

```python
class Carro:

    def __init__(self, is_sedan=False):
        self.__is_sedan = is_sedan
        self.__velocidade = 0
        self.__motorista = None

    def dirigir(self, motorista):
        self.__motorista = motorista
        self.acelerar(1)

    def acelerar(self, velocidade):
        self.__velocidade += velocidade

    def parar(self):
        self.__velocidade = 0
```

Revisão de Programação Orientada a Objetos

Métodos:

- Representam o comportamento dos objetos, ou seja, as ações que estes objetos podem praticar;
- Os métodos atuam nos atributos além de implementar a funcionalidade desejada para o objeto;

```python
from datetime import datetime

class Pessoa:

    def __init__(self, nome):
        self.__nome = nome
        self.__nascimento = datetime.now()
```

```python
class Carro:

    def __init__(self, is_sedan=False):
        self.__is_sedan = is_sedan
        self.__velocidade = 0
        self.__motorista = None

    def dirigir(self, motorista):
        self.__motorista = motorista
        self.acelerar(1)

    def acelerar(self, velocidade):
        self.__velocidade += velocidade

    def parar(self):
        self.__velocidade = 0
```

Revisão de Programação Orientada a Objetos

Objetos:

- Representam entidades no contexto da aplicação em desenvolvimento;
- Entidades interagem entre si para resolver problemas do mundo real;

- Exemplo:
- Pessoa é uma entidade e Carro também é uma entidade.
- Pessoa dirige Carro para se deslocar de um lugar para outro.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

(ppp)            @darkstar secao02]$ python
Python 3.9.1 (default, Dec 21 2020, 14:28:49)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from model import Pessoa, Carro
>>> angelina = Pessoa('Angelina')
>>> fusca = Carro()
>>> fusca.dirigir(angelina)
>>> fusca.__dict__
{'_Carro__is_sedan': False, '_Carro__velocidade': 1, '_Carro__motorista': <model.Pessoa object at 0x7ff53a3d0af0>}
>>>
```

Geek University

Evolua seu lado geek!

www.geekuniversity.com.br