Introdução aos Paradigmas de Programação e POO

Prof. Ítalo Assis

Ajude a melhorar este material =]

Encontrou um erro? Tem uma sugestão?

Envie e-mail para italo.assis@ufersa.edu.br

Agenda

- Paradigmas de programação
- Orientação a objetos
- Modelos

Paradigmas de programação

- É a forma como a solução para um determinado problema é desenvolvida
- Exemplos:
  - programação orientada a procedimentos;
  - programação orientada a objetos;
  - programação genérica;
  - programação funcional;
  - programação em lógica;
  - programação baseada em regras;
  - programação baseada em restrições;
  - programação orientada a aspectos.

Paradigmas de programação

- Prolog
  - Programação em lógica
  - Outros exemplos

```
estuda(charles,     csc135).
estuda(olivia,      csc135).
estuda(jackson,     csc131).
estuda(arthur,      csc134).

ensina(caio,        csc135).
ensina(pedro,       csc131).
ensina(pedro,       csc171).
ensina(julio,       csc134).

professor(X, Y) :- ensina(X, C), estuda(Y, C).

?- professor(caio, charles).
```

Paradigmas de programação

- Clojure
  - Programação funcional
  - Utilizado na Nubank
  - Outros exemplos

```
user=> (defn say-hello
         [name]
         (println (str "Hello, " name)))


user=> (say-hello "Kim")
Hello, Kim
nil
```

Paradigmas de programação

- A programação estruturada foi o paradigma mais difundido
  - ênfase em sequência, decisão e, iteração
  - normalmente formado por código em um único bloco
- A medida que os programas foram tornando-se mais complexos, surgiu a necessidade de resolver os problemas de uma maneira diferente
- Nesse contexto surge o paradigma da Programação Orientada a Objetos
- O Java suporta quatro paradigmas de programação: procedural, genérica, funcional e **orientada a objetos**

Orientação a objetos

- Dados e as operações que serão realizadas sobre estes formam um conjunto único (objeto)
- A resolução de um problema é dada em termos de interações realizadas entre esses objetos

Orientação a objetos

- Benefícios da abordagem orientada a objetos:
  - Modularidade:
    - uma vez criado, um objeto pode ser passado por todo o sistema;
  - Encapsulamento:
    - detalhes de implementação ficam ocultos externamente ao objeto;
  - Reuso:
    - uma vez criado, um objeto pode ser utilizado em outros programas;
  - Manutenibilidade:
    - manutenção é realizada em pontos específicos do programa (objetos).

Modelos

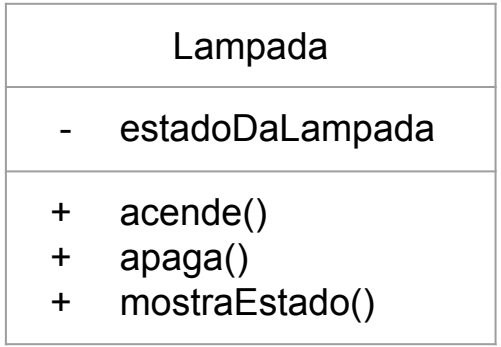
- Representações simplificadas de objetos, pessoas, itens, tarefas, processos, conceitos, ideia...
- Usados no dia a dia, independentemente do uso de computadores
- Normalmente possui dados (informações) e operações (procedimentos) associados a ele
    - Um carro, exemplo, possui dados, como ano e marca, e operações, como acelerar e freiar

Exemplo: Lâmpada

- Consideremos uma lâmpada comum
  - Dados:
    - Estado (ligada ou desligada)
  - Procedimentos
    - Acender
      - modificar seu estado para “ligada”
    - Apagar
      - modificar seu estado para “desligada”
    - Mostrar estado
- Os dados do modelo são somente os relevantes à abstração do mundo real que está sendo feita

Exemplo: Lâmpada

- Podemos representar modelos usando variantes do **diagrama de classes** da **UML** (*Unified Modeling Language*)
  - O retângulo superior mostra o nome do modelo
  - O retângulo central mostra dados
  - E o retângulo inferior mostra as operações
    - Os nomes das operações são seguidos de parênteses
    - Quando houver parâmetros, eles são listados dentro dos parênteses



Exemplo: Lâmpada

```
modelo Lampada
início
    do modelo
    dado estadoDaLampada;
    operação acende()
        início
            estadoDaLampada = aceso
        fim
    operação apaga()
        início
            estadoDaLampada = apagado
        fim
    operação mostraEstado()
        início
            se (estadoDaLampada == aceso)
                imprime "A lâmpada está acesa"
            senão
                imprime "A lâmpada está apagada"
            fim se
        fim
fim do modelo
```

Exemplo: conta bancária simplificada

- Vamos projetar um modelo de uma conta bancária simplificada que somente representa:
  - o nome do correntista
  - o saldo da conta
  - e se a conta é especial ou não
- Se a conta for especial, o correntista terá o direito de retirar mais dinheiro do que tem no saldo (ficar com o saldo negativo)

ContaBancariaSimplificada
- nomeDoCorrentista - saldo - contaEhEspecial
+ abreConta (nome, deposito, ehEspecial) + abreContaSimples (nome) + deposita (valor) + retira (valor) + mostraDados ( )

```
modelo ContaBancariaSimplificada
    início do modelo
    dado nomeDoCorrentista, saldo, contaEhEspecial
    operação abreConta(nome, deposito, ehEspecial)
        inicio
            nomeDoCorrentista = nome
            saldo = deposito
            contaEhEspecial = ehEspecial
        fim
    operação abreContaSimples(nome)
        inicio
            nomeDoCorrentista = nome
            saldo = 0.00
            contaEhEspecial = falso
fim

    operação deposita (valor)
        inicio
            saldo = saldo + valor
fim
```

```
    operação retira(valor)
        inicio
    se (valor >= saldo OU contaEhEspecial)
        saldo = saldo - valor
        fim
    operação mostraDados()
        inicio
    imprime “O nome do correntista é”
        imprime nomeDoCorrentista
    imprime “O saldo é “
        imprime saldo
        se (contaEhEspecial)
    imprime “A conta é especial”
        senão
    imprime “A conta é comum”
        fimse
        fim

fim do modelo
```

Exercício

- Vamos criar um modelo, através de um diagrama de classes, que represente um aluno da UFERSA

Modelos

- É possível criar modelos que contenham somente dados ou somente operações
- Modelos que contenham somente dados são pouco usados
  - Quando criamos modelos para representação de dados é interessante e útil adicionar algumas operações para manipulação desses dados
- Modelos que contenham somente operações podem ser considerados bibliotecas de operações
  - São exemplos desses modelos os grupos de funções matemáticas e de processamento de dados, que não precisem ser armazenados.

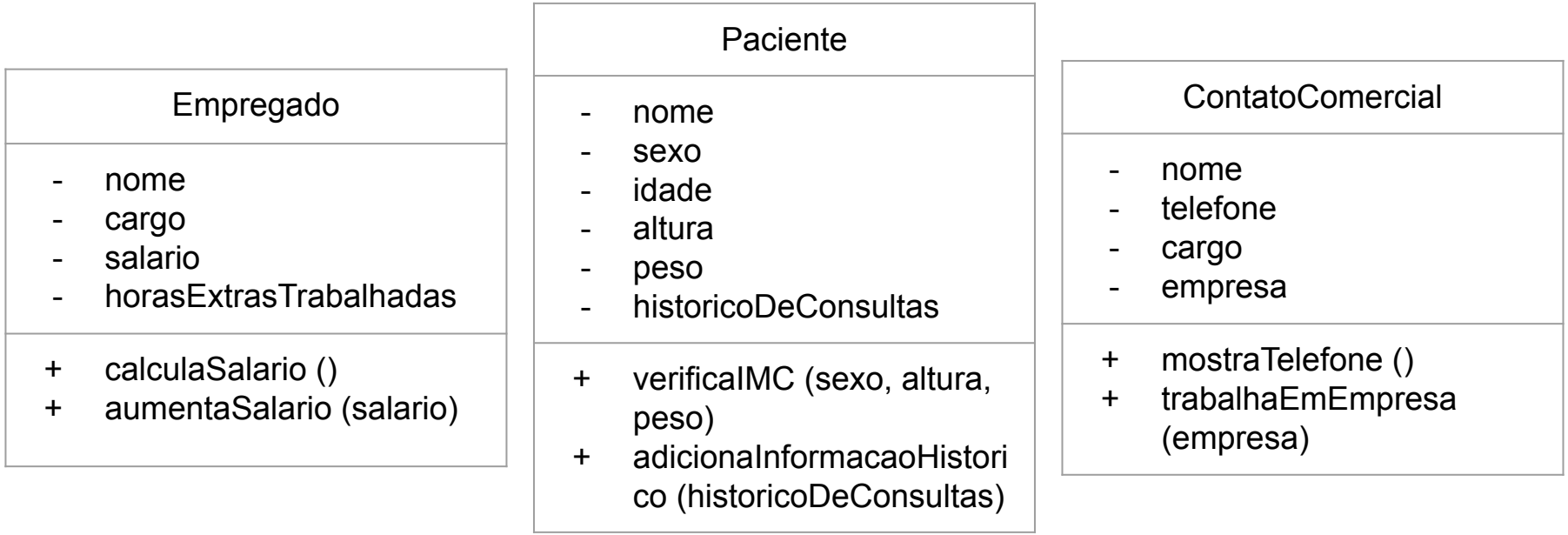
Modelos

- Modelos podem conter submodelos e ser parte de outros modelos
  - Um modelo de casa poderia ser composto por diversos modelos de lâmpada
  - Um modelo de lâmpada pode fazer parte tanto de um modelo de casa quanto de um modelo de carro, etc
- A simplificação inerente aos modelos é, em muitos casos, necessária
  - Dependendo do contexto, algumas informações devem ser ocultas ou ignoradas
- Por exemplo, a representação das informações sobre uma pessoa pode ser feita de maneira diferente, dependendo do contexto

Exemplo

- Vamos construir um modelo de uma pessoa para cada um dos contextos a seguir:
  - Pessoa como Empregado de Empresa
  - Pessoa como Paciente de uma Clínica Médica
  - Pessoa como Contato Comercial

Exemplo


Empregado
- nome
- cargo
- salario
- horasExtrasTrabalhadas
+ calculaSalario ()
+ aumentaSalario (salario)
Paciente
- nome
- sexo
- idade
- altura
- peso
- historicoDeConsultas
+ verificaIMC (sexo, altura, peso)
+ adicionaInformacaoHistorico (historicoDeConsultas)
ContatoComercial
- nome
- telefone
- cargo
- empresa
+ mostraTelefone ()
+ trabalhaEmEmpresa (empresa)

Modelos

- Modelos podem ser reutilizados para representar diferentes objetos, pessoas ou itens
  - O mesmo modelo *Paciente* poderia ser utilizado para representar cada um dos pacientes de uma clínica
  - Os pacientes podem ser representados pelo mesmo modelo, mas os dados individuais de cada um podem ser diferentes
  - João, Pedro e Maria seriam exemplos do modelo Paciente
- Não é necessária a criação de um modelo para cada item, pessoa ou objeto do mundo real
- A criação e uso de modelos é uma tarefa natural, e a extensão dessa abordagem à programação deu origem ao **paradigma de Orientação a Objetos**

Referências

BATISTA, Rogério da Silva; MORAES, Rafael Araújo de. **Introdução à Programação Orientada a Objetos**. 2013. Disponível em: http://proedu.rnp.br/handle/123456789/611. Acesso em: 18 ago. 2021.

STROUSTRUP, Bjarne. **Princípios e práticas de programação com C++**. São Paulo: Bookman, 2012. 1216p. ISBN: 9788577809585.