



Introdução a Orientação a Objetos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Dezembro de 2019



Qualidade de *Software*

- Obter *software* de qualidade é um dos principais objetivos da Engenharia de *Software*;
- Desafio desde os primórdios da computação;
- Glend Myers(1975), escreveu que: obter qualidade em *software* de grande porte, era um dos maiores desafios para época.



Fatores de Qualidade

- **Fatores Externos:**
 - para o usuário o software deve ser:
 - **robusto** – realiza de forma correta suas tarefas, mesmo quando submetido a condições extremas;
 - **confiável** – realiza de maneira simples e correta, as tarefas para qual foi desenvolvido;
 - **eficiente** – de fazer bom uso dos recursos de *hardware*.



Fatores de Qualidade

- **Fatores Externos:**
 - para o usuário o software deve ter:
 - **usabilidade** - deve ser fácil de ser usado;
 - **integridade** - os dados manipulados pelo software devem estar sempre íntegros;
 - **portabilidade** - possibilidade de utilização do *software* em diversos ambientes de *software* e/ou *hardware*.



Fatores de Qualidade

- **Fatores Internos:**
 - visão Estrutural do *Software*:
 - **modularidade** - característica de um *software* que foi construído em módulos;
 - **legibilidade** - qualidade que determina a facilidade de leitura do código-fonte, lógica simples e fácil de ser entendida;
 - **manutenibilidade** - facilidade na hora de realizar a manutenção no *software*.



Paradigmas da Programação

- Ao longo dos anos, a Engenharia de *Software* disponibilizou vários métodos, técnicas e ferramentas para auxiliar os desenvolvedores de *software* a produzirem com maior qualidade e menos tempo;
- Assim foram criados os diversos paradigmas da programação:
 - Programação “desestruturada”,
 - Programação Estruturada,
 - Programação Orientada a Objetos e
 - Programação Orientada a Aspecto.



Programação “Desestruturada”

- Lógica de programação difícil de ser entendida;
- Uso indiscriminado de comandos de desvio de fluxo (*GO TO*);
- Difícil de criar código reutilizável;
- Uma verdadeira “macarronada”, como denominam alguns autores.



Programação Estruturada

- Também conhecida como Programação Imperativa;
- Grande avanço na programação da época;
- Funcionalidades do *software* agrupadas em funções ou procedimentos. Que eram chamados a partir de um programa principal;
- Baniu-se o uso da instrução de desvio de fluxo (*GO TO*);
- Possibilidade de reutilização de código de maneira simples. Desde que devidamente projetados.



Programação Orientada a Objetos

- Com o advento da Programação Orientada a Objetos (POO), houve um salto na organização de sistemas. O mundo passou a ser **modelado** na forma de **classes** e objetos;
- O domínio do problema a ser resolvido caracteriza-se por classes de objetos, que possuem **atributos** e **comportamentos** e que se relacionam entre si;
- As classes podem ser reutilizadas com maior facilidade;
- A área de componentização, conceitos de pacotes e reusabilidade, deixaram a teoria e passaram a frequentar a realidade;
- Apesar destes avanços, o aumento da complexidade (nos sistemas) continua crescendo e tornando o desenvolvimento de sistemas uma atividade onerosa em termos de tempo, complexidade e pessoal capacitado a desenvolvê-los.



Programação Orientada a Aspectos

- A Programação Orientada a Aspecto (POA), tem o objetivo de tentar separar os níveis de preocupação durante o desenvolvimento de *software*;
- Assim sendo, com POA é possível pensar separadamente nos problemas referentes à persistência de dados, modelagem de negócios, segurança, distribuição, auditoria, *logs* e etc.
- A proposta da POA é desenvolver partes do sistema sem se preocupar com as demais partes;
- Cada parte do sistema, que se deseja trabalhar isoladamente pode ser chamada de “aspecto”;
- Por enquanto, seu uso está quase que restrito a área acadêmica.



Fundamentos das Linguagens Orientadas a Objetos

- A ideia básica da Orientação a Objetos (OO) é:
 - o *software* deve ser constituído por objetos que representem os objetos que compõem o mundo real;



Fundamentos das Linguagens Orientadas a Objetos

- A ideia básica da Orientação a Objetos (OO) é:
 - o domínio do problema a ser resolvido caracteriza-se por classes de objetos, que possuem atributos e comportamentos e que se relacionam entre si;

```
1 package principal;
2
3 import java.awt.Color;
4
5 public class Carro {
6     private String marca;
7     private String modelo;
8     private int anoFabricacao;
9     private int anoModelo;
10    private Color cor;
11    private int quantidadePortas;
12    private float preco;
13
14    public Carro(String marca, String modelo, int anoFabricacao,
15                int anoModelo, Color cor, int quantidadePortas, float preco) {
16        this.marca = marca;
17        this.modelo = modelo;
18        this.anoFabricacao = anoFabricacao;
19        this.anoModelo = anoModelo;
20        this.cor = cor;
21        this.quantidadePortas = quantidadePortas;
22        this.preco = preco;
23    }
24
25    public String getMarca() {
26        return marca;
27    }
28
29    public void setMarca(String marca) {
30        this.marca = marca;
31    }
32
33    public String getModelo() {
```

Fundamentos das Linguagens Orientadas a Objetos

- A ideia básica da Orientação a Objetos (OO) é:
 - o modelo utilizado no *software* orientado a objetos, está mais próximo do mundo real do que aquele utilizado no paradigma estruturado.



```
1 package principal;
2
3 import java.awt.Color;
4
5 public class Carro {
6     private String marca;
7     private String modelo;
8     private int anoFabricacao;
9     private int anoModelo;
10    private Color cor;
11    private int quantidadePortas;
12    private float preco;
13
14    public Carro(String marca, String modelo, int anoFabricacao,
15                int anoModelo, Color cor, int quantidadePortas, float preco) {
16        this.marca = marca;
17        this.modelo = modelo;
18        this.anoFabricacao = anoFabricacao;
19        this.anoModelo = anoModelo;
20        this.cor = cor;
21        this.quantidadePortas = quantidadePortas;
22        this.preco = preco;
23    }
24
25    public String getMarca() {
26        return marca;
27    }
28
29    public void setMarca(String marca) {
30        this.marca = marca;
31    }
32
33    public String getModelo() {
```



Fundamentos das Linguagens Orientadas a Objetos

- Para se construir um *software*:
 - usando **Programação Estruturada**, devemos fazer a seguinte pergunta: “**o que o sistema faz?**” e
 - usando **Programação Orientada a Objetos**, perguntaremos da seguinte forma: “**quais são os objetos que compõem o sistema e como eles se relacionam?**”.

Questões-chave na construção de um *software* orientado a objetos

- **Identificar objetos:**

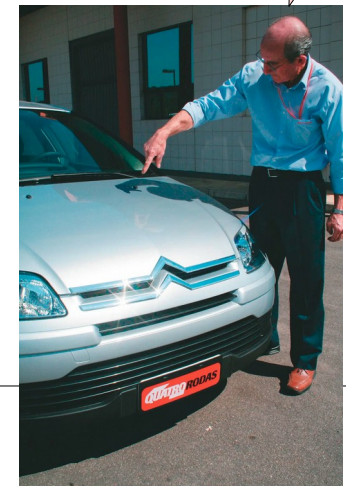
- o *software* deve refletir os objetos do mundo real, ou seja, os objetos estão no mundo real, basta identificá-los;

Objeto **Concessionária**

Objeto **Contrato de Venda**

Objeto **Cliente**

Objeto **Carro**



Questões-chave na construção de um *software* orientado a objetos

- **Descrever os objetos:**
 - a descrição dos objetos deve representar suas características e comportamentos;
 - O elemento que descreve os objetos são as classes.

```
1 package principal;
2
3 import java.awt.Color;
4
5 public class Carro {
6     private String marca;
7     private String modelo;
8     private int anoFabricacao;
9     private int anoModelo;
10    private Color cor;
11    private int quantidadePortas;
12    private float preco;
13
14    public Carro(String marca, String modelo,
15                int anoModelo, Color cor,
16                int quantidadePortas, float preco) {
17        this.marca = marca;
18        this.modelo = modelo;
19        this.anoFabricacao = anoModelo;
20        this.cor = cor;
21        this.quantidadePortas = quantidadePortas;
22        this.preco = preco;
23    }
24
25    public String getMarca() {
26        return marca;
27    }
28
29    public void setMarca(String marca) {
30        this.marca = marca;
31    }
32
33    public String getModelo() {
```

```
1 package principal;
2
3 import java.time.LocalDate;
4
5 public class ContratoDeVenda {
6     private Cliente cliente;
7     private Carro carro;
8     private LocalDate data;
9
10    public ContratoDeVenda(Cliente cliente,
11                          Carro carro, LocalDate data) {
12        super();
13        this.cliente = cliente;
14        this.carro = carro;
15        this.data = data;
16    }
17
18    public Cliente getCliente() {
19        return cliente;
20    }
21
22    public void setCliente(Cliente cliente) {
23        this.cliente = cliente;
24    }
25
26    public Carro getCarro() {
27        return carro;
28    }
```

```
1 package principal;
2
3 public class Cliente {
4     private String nome;
5     private String endereco;
6     private String telefone;
7     private int cpf;
8     private String identidade;
9
10    public Cliente(String nome, String endereco,
11                  String telefone, int cpf,
12                  String identidade) {
13        super();
14        this.nome = nome;
15        this.endereco = endereco;
16        this.telefone = telefone;
17        this.cpf = cpf;
18        this.identidade = identidade;
19    }
20
21    public String getNome() {
22        return nome;
23    }
24
25    public void setNome(String nome) {
26        this.nome = nome;
27    }
28
29    public String getEndereco() {
30        return endereco;
31    }
```

```
1 package principal;
2
3 import java.util.ArrayList;
4
5 public class Concessionaria {
6     private ArrayList<Carro> carros;
7
8     public Concessionaria(ArrayList<Carro> carros) {
9         this.carros = carros;
10    }
11
12    public ArrayList<Carro> getCarros() {
13        return carros;
14    }
15
16    public Carro getCarro(int posicao) {
17        return carros.get(posicao);
18    }
19
20    public void setCarros(ArrayList<Carro> carros) {
21        this.carros = carros;
22    }
23 }
```


Questões-chave na construção de um *software* orientado a objetos

3. Descrever as relações entre as classes de objetos:

- o tipo de relacionamento existente entre dois objetos (por exemplo: é um, possui, compõe, etc.) deve ser representado diretamente no *software*.



RELACIONAMENTO:
concessionária
possui carro(s).



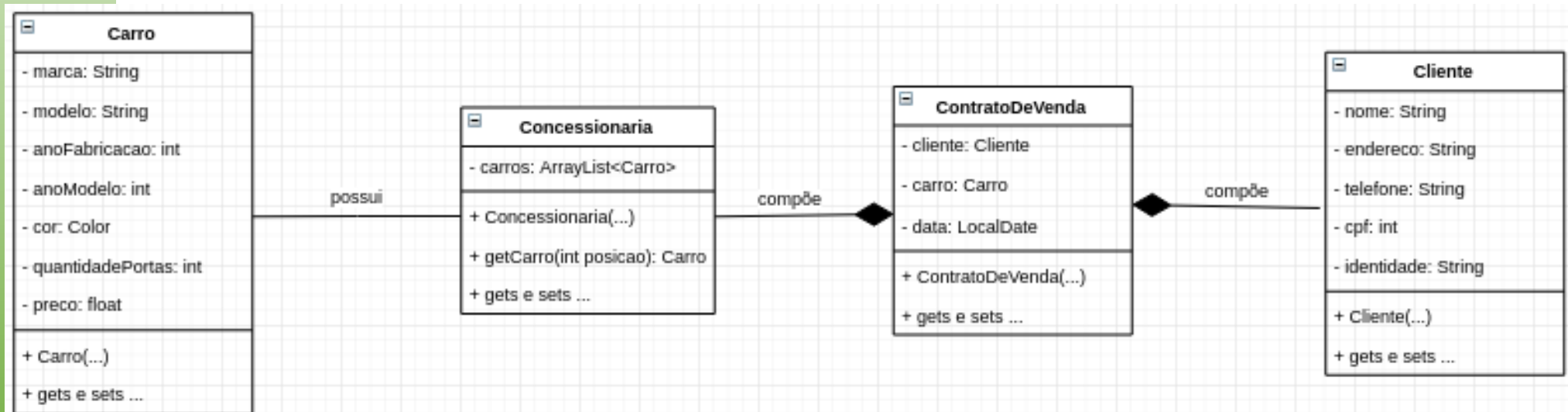
RELACIONAMENTO:
Concessionária compõe
contrato de venda com
o objeto Carro vendido.



RELACIONAMENTO:
Cliente compõe
contrato de venda.

Questões-chave na construção de um *software* orientado a objetos

3. Descrever as relações entre as classes de objetos, em UML (*Unified Modeling Language*):





Orientação a Objetos

- Os **objetos** encontrados são categorizados em **classes**, como no *slide* anterior;
- **as classes constituem o núcleo de um programa Orientado a Objetos (OO);**
- Conclusão:
 - um *software* OO, é constituído por classes que descrevem o comportamento e as características de objetos, que interagem entre si;
 - **o programa principal tem a função de criar os objetos principais e iniciar a execução.**



Bibliografia

- 4 RODAS. **Como avaliar um carro usado com olhos de profissional**. Disponível em: <<https://quatorrodas.abril.com.br/auto-servico/como-avaliar-um-carro-usado-com-olhos-de-profissional/>>. Acesso em: 27 Dez. 2019.
- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6. ed., São Paulo: Pearson, 2005.
- DESENHOS INFANTIS.COM. **Galeria de Imagens**. Disponível em: <<https://www.desenhosinfantis.com/Imagens/carro-vermelho.jpg>>. Acesso em: 27 Dez. 2019.
- FERREIRA, Kecia Aline Marques. *Slides* da disciplina de Programação de Computadores II. CEFET-MG, 2009.
- GO!FIT. **Contrato de prestação de serviços: O que é e como elaborar um para o seu negócio?**. Disponível em: <<http://gofitweb.com/blog/contrato-de-prestacao-de-servicos-o-que-e-e-como-elaborar-um-para-o-seu-negocio/>>. Acesso em: 27 Dez. 2019.
- RESENDE, A. M. P. de; SILVA C. C. **Programação Orientada a Aspectos em Java**. Rio de Janeiro: Brasport, 2005.
- RODA RIO. **Concessionária carioca é a nova cara da PSA Peugeot Citroën**. <<https://rodario.com.br/2019/03/22/concessionaria-carioca-e-a-nova-cara-da-psa-peugeot-citroen/>>. Acesso em: 27 Dez. 2019.