



**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
RIO GRANDE DO SUL
Câmpus Feliz

Orientação a Objetos em Java

Prof. Moser Fagundes

Programação II

Sumário

- Introdução
- Classes
- Atributos
- Métodos
- Escopo
- Objetos
- Exercícios

Programação procedural

Até agora, programamos usando uma abordagem **estruturada**, onde o problema é decomposto em uma **série de instruções** conforme o exemplo abaixo.

```
public class Exemplo {  
    public static void main(String[] args) {  
  
        float f = 1.5f;  
        double d = 2.7;  
        int i1 = (int) f;  
        int i2 = (int) d;  
  
        System.out.println(f);  
        System.out.println(d);  
        System.out.println(i1);  
        System.out.println(i2);  
    }  
}
```

Quais seriam as limitações desta abordagem?

Programação orientada a objetos

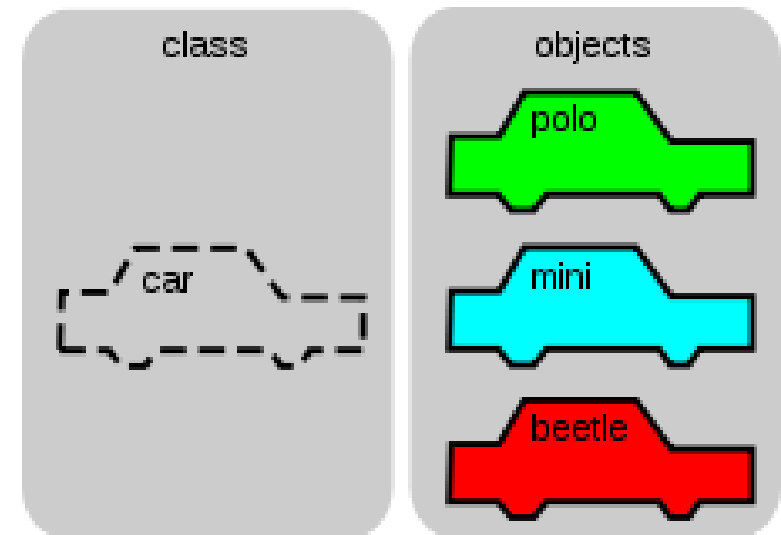
Na **Programação Orientada a Objetos** (POO), o desenvolvimento de software é baseado na **composição** e **interação** entre unidades conhecidas como **objetos**.

Por que aprender POO?

- **Primeira razão (livre e espontânea pressão)**
É o paradigma de programação do momento no mercado.
- **Segunda razão (aproveitar o potencial de Java)**
Java é uma linguagem projetada para desenvolvimento orientado a objetos.
- **Terceira razão (modularidade)**
O desenvolvimento orientado a objetos permite um melhor reaproveitamento de código e organização do mesmo. Facilita também a manutenção do projeto.

O que é a programação OO?

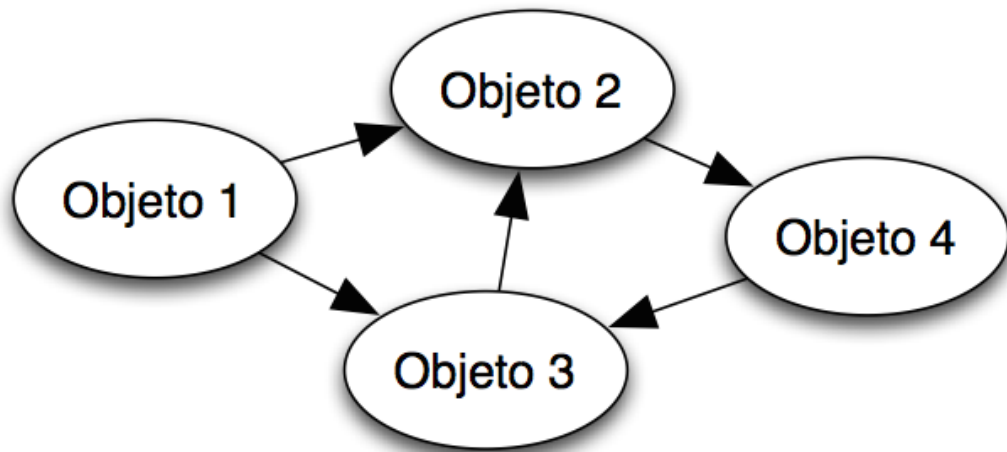
- Paradigma de programação no qual um programa é estruturado em **objetos**.
- Permite que objetos do mundo real (*carros, filmes, bicicletas, celulares, etc.*) que possuem características semelhantes, sejam **abstraídos** e **mapeados** em objetos no programa de computador.
- Objeto é uma entidade que combina **estrutura de dados** e **comportamento funcional**.



abstrair = focar nos aspectos relevantes
ignorando características menos importantes

O que é a programação OO?

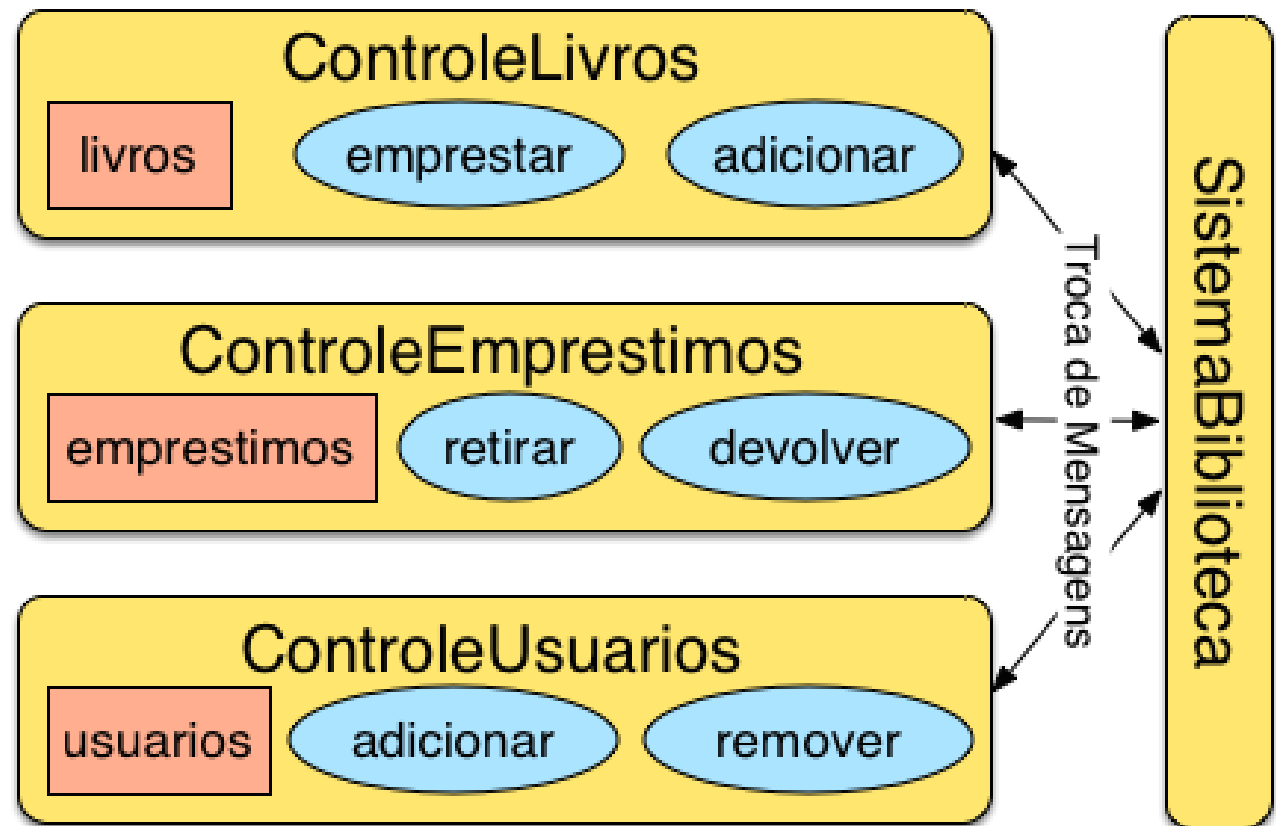
No paradigma **orientado a objetos**, decomponemos o problema em **objetos**. A solução será construída a partir da **interação entre esses objetos** (através da **troca de mensagens** – execução de métodos).



Exemplo de organização orientada a objetos

Exemplo de organização de um **sistema** de uma biblioteca com orientação a objetos:

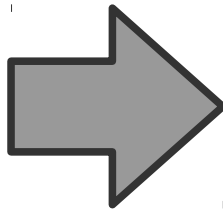
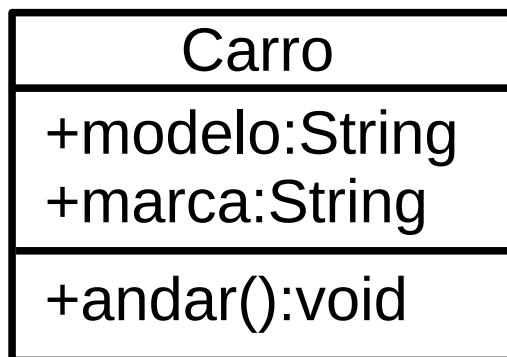
- Dados e métodos relacionados ficam na mesma classe
- Comunicação por troca de mensagens (chamadas de métodos)



Classe

- Uma classe representa um tipo **abstrato de dado**. As classes são usadas para representar objetos complexos que têm **estado** e **comportamento**.
- Uma classe é um **gabarito para a definição de objetos** que possui:
 - **Atributos**: são propriedades da classe e armazena o seu estado;
 - **Métodos**: são funcionalidades (*comportamento*) que permitem a modificação do seu estado.

Diagrama de classes simplificado




Código Java

```
public class Carro {
    public String modelo;
    public String marca;


    public void andar() {
        System.out.println("Rodando");
    }
}
```

Classe

Convenção: nome de classe SEMPRE começa por letra Maiúscula.



```
public class Carro {  
    public String modelo;  
    public String marca;  
  
    public void andar() {  
        System.out.println("Rodando");  
    }  
}
```




```
public class Pessoa {  
    public String nome;  
    public int idade;  
  
    public void nascer() {  
        nome = "Luke Skywalker";  
        idade = 0;  
    }  
}
```


Atributo

- Os atributos definem as **propriedades/características** que o objeto possui. Um atributo é identificado por um **nome** e um **tipo**.
- Os atributos podem **opcionalmente** estar inicializados, conforme o exemplo da classe **Carro** abaixo. No exemplo da **Pessoa** não há inicialização.
- **Convenção**: o nome de um atributo deve **SEMPRE** começar por uma letra **minúscula**!

```
public class Carro {  
    public String modelo = "Civic";  
    public String marca = "Honda";  
  
    public void andar() {  
        System.out.println("Rodando");  
    }  
}
```



```
public class Pessoa {  
    public String nome;  
    public int idade;  
  
    public void nascer() {  
        nome = "Luke Skywalker";  
        idade = 0;  
    }  
}
```



Método

- Os métodos definem as **ações** (funcionalidades) da classe, ou seja, o que os objetos dessa classe **fazem**.
- Um método é composto por:
 - um **identificador** para o método (**o nome do método**),
 - o **tipo** para o **valor de retorno**,
 - e **opcionalmente por sua lista de argumentos**, sendo cada argumento identificado por seu tipo e nome.
- **Convenção:** o nome de um método deve **SEMPRE** começar por uma letra **minúscula**!

Exemplos de métodos

```
public class Pessoa {  
    public String nome  
    public int idade;
```

```
    public void nascer() {  
        nome = "Luke Skywalker";  
        idade = 0;  
    }
```

```
    public void falar(String frase) {  
        System.out.println(nome + " diz: " + frase);  
    }
```

```
    public int getIdade() {  
        return idade;  
        // 0 que vier depois nunca é executado  
    }
```

```
}
```

Método sem parâmetro.
O **void** significa que o método não retorna nada.

Método com parâmetro.
Como o método acima, também não retorna nada.

Método sem parâmetro.
Retorna o atributo idade.

Exemplos de classes

Vamos criar as seguintes classes:

- Livro
- Filme
- Cidade
- Computador

Que outras classes podemos criar?

Escopo

Os **atributos** são acessíveis em **qualquer lugar dentro da classe**, mas as variáveis criadas dentro de um método são inacessíveis fora deste.

```
public class Carro {
```

```
    public String modelo = "Civic";  
    public String marca = "Honda";
```

Os atributos são
acessíveis de
qualquer lugar
dentro da classe

```
    public void imprimir() {  
        System.out.println(modelo + " da " + marca);  
    }
```

```
    public void andar() {  
        String ruido = "Rodando";  
        System.out.println(ruido);  
    }
```

As variáveis são
acessíveis somente
dentro do método

```
}
```

Objeto

- **Objetos são instâncias de classe**: quando instanciamos um objeto de uma classe, estamos criando um **novo item** do conjunto representado por essa classe.
- Cada objeto possui:
 - **Estado**: objetos diferentes podem ter valores diferentes para os seus atributos.
 - **Comportamento**: definido pelos métodos, que são comuns a todos objetos da mesma classe.

Objeto

- **Exemplos** de objetos das classes:
 - **Carro**: Civic, Fiesta, S10, Corolla, HRV, ...
 - **Livro**: O Mundo de Sofia, Tutorial Java, ...
 - **Filme**: Matrix, Sexto Sentido, Avatar, Blade Runner, ...
 - **Cidade**: ?
 - **Computador**: ?

Criação de objetos

Em Java, a criação de um objeto se dá através do operador **new**.

No exemplo abaixo, **cada linha** cria **um** objeto da classe **Carro**.

```
Carro c1 = new Carro();  
Carro c2 = new Carro();  
Carro c3 = new Carro();
```

Exemplo com um objeto

No exemplo abaixo, criamos um objeto da classe **Carro** e depois chamamos os seus métodos.

```
package exemplos;
```

```
public class TestaCarro {  
    public static void main(String[] args) {  
        Carro c1 = new Carro();  
        c1.andar();  
        c1.imprimir();  
    }  
}
```

Exemplo com dois objetos

No exemplo abaixo, criamos objetos da classe **Carro**, colocamos valores nos seus atributos e chamamos os seus métodos.

```
package exemplos;
```

```
public class TestaCarro {  
    public static void main(String[] args) {  
        Carro c1 = new Carro();  
        c1.andar();  
        c1.imprimir();  
  
        Carro c2 = new Carro();  
        c2.modelo = "S10";  
        c2.marca = "GM";  
        c2.andar();  
        c2.imprimir();  
    }  
}
```

Comparação de objetos

Note que **c1** e **c2** são objetos diferentes apesar de terem o mesmo valor em seus atributos.

```
package exemplos;
```

```
public class ComparaCarros {  
    public static void main(String[] args) {
```

```
        Carro c1 = new Carro();  
        Carro c2 = new Carro();
```

```
        if(c1 == c2) {  
            System.out.println("Iguais");  
        } else {  
            System.out.println("Diferentes");  
        }
```

```
    }
```

```
}
```

Exemplos

Crie objetos para as seguintes classes:

- Livro
- Filme
- Cidade
- Computador

Teste os objetos executando os seus métodos.

Exercício

Lista de exercícios no Moodle.