Faculdade POSITIVO Londrina

# Orientação a objetos em Java

Profª Dra Liliane Santana Oliveira Kashiwabara

Faculdade POSITIVO Londrina

# Programação Orientado a Objetos

- Paradigma de programação em que o programador visualiza seu programa em execução como uma coleção de objetos
- Forma de programar **mais próxima** de como percebemos os elementos mundo real do que outros tipos de programação
- Os objetos possuem **estados** e **comportamentos**
- Objetos **comunicam-se** através de **mensagens**

Faculdade POSITIVO Londrina

Abstração

Encapsulamento

Princípios da programação OO

Polimorfismo

Herança

Faculdade POSITIVO Londrina

Concentra-se apenas em partes do sistema

Abstração

Ex: um médico é especialista em apenas uma área.

Encapsulamento

Princípios da programação OO

Polimorfismo

Herança

Faculdade POSITIVO Londrina

Abstração

Encapsulamento

Princípios da programação OO

Polimorfismo

Herança

Detalhes internos da implementação dos objetos se mantém ocultos

Exemplo: para telefonar para alguém você simplesmente pega um telefone e disca o número destino

Abstração

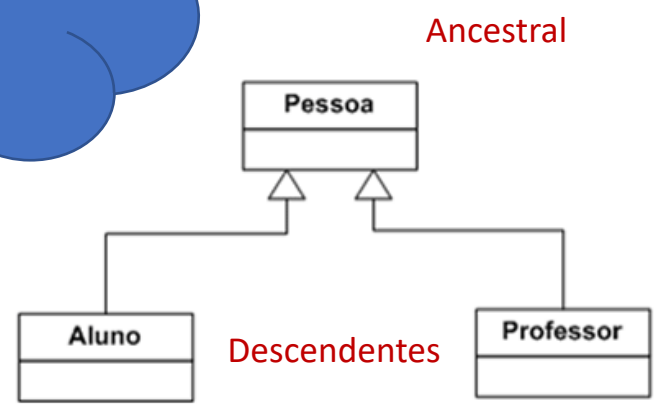
Encapsulamento

Princípios da programação OO

Polimorfismo

Herança

Reusabilidade




Os descendentes herdam as características de seus ancestrais

Faculdade POSITIVO Londrina

Abstração

Como dois objetos respondem a uma mesma mensagem

Encapsulamento

Princípios da programação OO

Polimorfismo

Herança

**Animal**

Comportamento: emite som

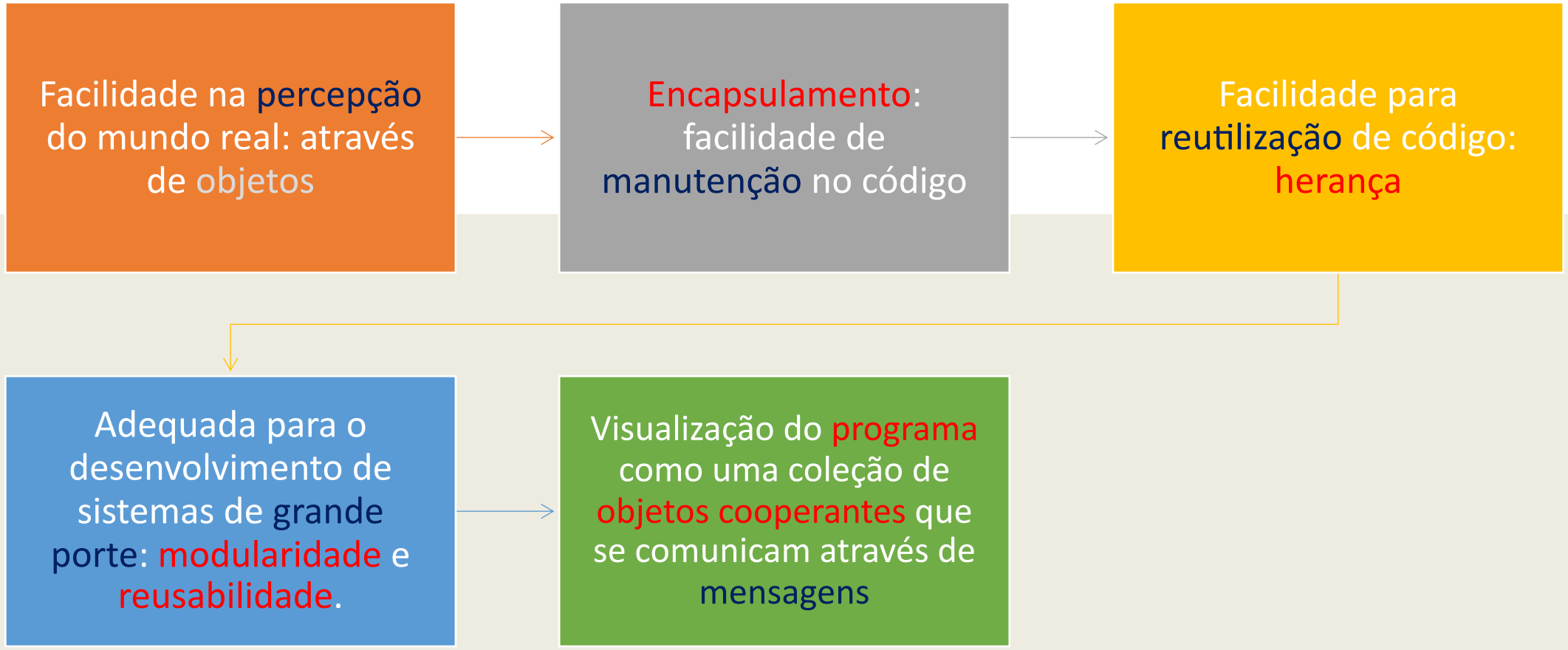
Late

Cacareja

Mia

Fonte: http://contembits.com.br/minicursos/poo/Polimorfismo.aspx

# Benefícios


Facilidade na percepção do mundo real: através de objetos
Encapsulamento: facilidade de manutenção no código
Facilidade para reutilização de código: herança
Adequada para o desenvolvimento de sistemas de grande porte: modularidade e reusabilidade.
Visualização do programa como uma coleção de objetos cooperantes que se comunicam através de mensagens

# Objeto

- **Elementos** do mundo real:
  - **Tudo** no mundo real é visto como objeto
- Podem ser **concretos** ou **abstratos**
  - **Concretos**: pessoa, carro, nota fiscal
  - **Abstratos**: conta corrente, venda
- Possuem **estados** e **comportamentos;**
- **Comunicam-se** através de **mensagens.**

# Exemplos de objetos

**Ferrari**
Placa KZE1018
Vermelho
4 portas

**Liga**
**Desliga**
**Acelera**
**Frear**

**Scooby**
Médio porte
Vira-lata
2 anos

**Anda**
**Late**
**Come**
**Dorme**
**Pega osso**

**Camila**
Cabelos negros
20 anos
Estudante

**Andar**
**Correr**
**Comer**

Fontes: figuras adaptadas extraídas de:
https://intranet.ifs.ifsuldeminas.edu.br/michelle.nery/Linguagem%20de%20Programa%C3%A7%C3%A3o%20Java%20-%20Integrado/Aula%2019%20-%20POO/Aula%2016%20-%20POO.pdf
http://fabrica.ms.senac.br/2015/03/classes-objetos-atributos-e-metodos-em-java/

# Classe

- Modelagem de um conjunto de objetos: **estados** (atributos) e **comportamentos** (métodos) comuns.
- **Conceito**: estrutura que **abstrai** um conjunto de objetos com características **similares.**
- **Molde** para a criação de objetos
- Objetos são **instâncias** de classes: representações concretas das classes
- Classes se relacionam por meio de **herança**.

**Tipo:** Porsche

**Cor:** Cinza

**Placa:** MHZ-4345

**Número de Portas:** 2

Liga
Desliga
Acelera
Frea

← OBJETOS →

**Tipo:** Ferrari

**Cor:** Vermelho

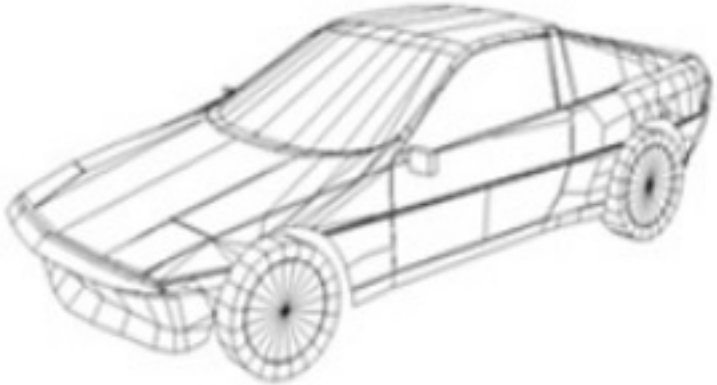
**Placa:** JKL-0001

**Número de Portas:** 4

Liga
Desliga
Acelera
Frea

Faculdade POSITIVO Londrina

CLASSE →



**Tipo:** ?

**Cor:** ?

**Placa:** ?

**Número de Portas:** ?

| Carro |
| --- |
| - Tipo<br>- Cor<br>- Placa<br>- Numero de portas |
| + Liga<br>+ Desliga<br>+ Acelera<br>+ Freia |

← OBJETOS →

**Tipo:** Porsche

**Cor:** Cinza

**Placa:** MHZ-4345

**Número de Portas:** 2

Liga
Desliga
Acelera
Frea

**Tipo:** Ferrari

**Cor:** Vermelho

**Placa:** JKL-0001

**Número de Portas:** 4

Liga
Desliga
Acelera
Frea

# Classe

Uma classe é o **modelo** para a **construção de objetos**. Ela define as **características** e **comportamentos** que os objetos irão possuir. A classe é **abstrata**: ela não existe concretamente!

Um objeto é uma **instância** de uma classe!

# Criando classes em Java

- Classe é um modelo para a criação de objetos;
- Para criarmos uma classe, precisamos definir
  - Atributos
  - Métodos;
- Exemplo: classe Carro

# Criação de classes

- Como vimos no slide anterior, a classe Carro possui os seguintes atributos:
  - Tipo
  - Cor
  - Placa
  - Número de portas

| **Carro** |
| --- |
| - Tipo<br>- Cor<br>- Placa<br>- Numero de portas |
| + Ligar<br>+ Desligar<br>+ Acelerar<br>+ Frear |

# Criação de classes

- Como vimos no slide anterior, a classe Carro possui os seguintes atributos:
  - Tipo
  - Cor
  - Placa
  - Número de portas

```
public class Carro {
    String tipo;
    String cor;
    String placa;
    int num_portas;
}
```

# Criação de classes

- Métodos definidos para o Carro:
  - Ligar
  - Desligar
  - Acelerar
  - Frear

| **Carro** |
| --- |
| - Tipo<br>- Cor<br>- Placa<br>- Numero de portas |
| + Ligar<br>+ Desligar<br>+ Acelerar<br>+ Frear |

# Criação de classes

- Métodos definidos para o Carro:
  - Ligar
  - Desligar
  - Acelerar
  - Frear

```
public class Carro {
    String tipo;
    String cor;
    String placa;
    int num_portas;

    public void ligar() {
        System.out.println("Carro ligado!");
    }

    public void desligar() {
        System.out.println("Carro desligado!");
    }

    public void acelerar() {
        System.out.println("Carro acelerando!");
    }

    public void frear() {
        System.out.println("Carro freando!");
    }
}
```

# Criação de objetos

- Vamos agora instanciar um objeto da classe Carro
- Para isso, vamos criar uma nova classe
- Vamos criar um objeto com os seguintes

atributos:

**Tipo:** Ferrari

**Cor:** Vermelho

**Placa:** JKL-0001

**Número de Portas:** 4

# Criação de objetos

```
public class ObjetoCarro {

    public static void main(String [] args) {

        Carro c1 = new Carro();

        c1.tipo = "Ferrari";
        c1.cor = "Vermelha";
        c1.placa = "JKL-0001";
        c1.num_portas = 4;

        System.out.println("Dados do carro 1: \n"
                        +  "Tipo: " + c1.tipo +"\n"
                        +  "Cor: " + c1.cor + "\n"
                        +  "Placa: " + c1.placa + "\n"
                        +  "Número de portas: " + c1.num_portas + "\n" );
        c1.ligar();
        c1.acelerar();
        c1.frear();
        c1.desligar();
    }
}
```

# Visibilidade dos atributos e métodos

- Por padrão, os atributos de uma classe são **públicos**
  - Por isso eles são acessados diretamente através de seus nomes

```
c1.tipo = "Ferrari";
c1.cor = "Vermelha";
c1.placa = "JKL-0001";
c1.num_portas = 4;
```

Associação de valores feita diretamente

```
System.out.println("Dados do carro 1: \n"
                 +  "Tipo: " + c1.tipo +"\n"
                 +  "Cor: " + c1.cor + "\n"
                 +  "Placa: " + c1.placa + "\n"
                 +  "Número de portas: " + c1.num_portas + "\n" );
```

Acesso feito diretamente

# Visibilidade dos atributos e métodos

- Public: nível de acesso irrestrito
  - Conseguimos acessar os atributos e métodos diretamente
- Private: nível de acesso se restringe apenas a classe
  - Por boas práticas, os atributos são privados.
  - Criação de métodos para acessar esses atributos (getters e setters)
    - **Get:** retorna o valor do atributo
    - **Set:** Atribui valor ao atributo

# Visibilidade dos atributos e métodos

```java
public class Carro {
    private String tipo;
    private String cor;
    private String placa;
    private int num_portas;

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String t) {
        tipo = t;
    }

    public String getCor() {
        return cor;
    }

    public void setCor(String c) {
        cor = c;
    }

    public String getPlaca() {
        return placa;
    }

    public void setPlaca(String p) {
        placa = p;
    }

    public int getNum_portas() {
        return num_portas;
    }

    public void setNum_portas(int np) {
        num_portas = np;
    }
```

# Visibilidade dos atributos e métodos

- Mudança na forma de acesso dos atributos

```
public class ObjetoCarro {

    public static void main(String [] args) {

        Carro c1 = new Carro();

        c1.setTipo("Ferrari");
        c1.setCor("Vermelha");
        c1.setPlaca("JKL-0001");
        c1.setNum_portas(4);

        System.out.println("Dados do carro 1: \n"
                        +  "Tipo: " + c1.getTipo() +"\n"
                        +  "Cor: " + c1.getCor() + "\n"
                        +  "Placa: " + c1.getPlaca()+ "\n"
                        +  "Número de portas: " + c1.getNum_portas() + "\n" );
        c1.ligar();
        c1.acelerar();
        c1.frear();
        c1.desligar();
    }
}
```

# Exercícios

1. Crie a classe Pessoa (defina os atributos que achar necessário). Para essa classe, os atributos devem ser privados. Criem também os métodos de acesso aos atributos e também um método que imprime os dados da pessoa. Crie também uma outra classe com um método principal, que instancia um objeto da classe Pessoa, atribui valores aos atributos e acessa os seus métodos.

# Exercícios

2. Crie uma classe para representar uma pseudo calculadora, que possui dois números inteiros como atributos. Nessa classe, os dois atributos devem ser privados. Por isso, criem também os métodos que irão acessar os atributos (getters e setters). Crie também métodos para realizar as seguintes operações com os dois atributos: soma, subtração, divisão e multiplicação. Crie também uma outra classe, que possui um método principal e que instancie um objeto da classe criada, e que realiza as operações disponibilizadas pelo objeto.

# Exercícios

3. Escreva em Java uma classe Contador, que encapsule um valor usado para contagem de itens ou eventos (ou seja, possui um atributo do tipo inteiro que será responsável pela contagem). A classe deve possuir os métodos de acesso (get e set) e os seguintes métodos:

    a) Zerar(): atribui zero ao atributo da classe;

    b) Incrementar(): adiciona 1 ao atributo da classe;

    c) Decrementar(): subtrai um do atributo da classe.

    d) Somar (valor): adiciona o valor passado por parâmetro ao atributo da classe;

    e) Subtrair (valor):  subtrai o valor passado por parâmetro do atributo da classe.

# Palavra reservada this

- Faz referência ao objeto em um método

```
public void setTipo(String tipo) {
    this.tipo = tipo;
}
```

# Palavra reservada this

Modificando os métodos de acesso aos atributos...

```
public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public String getCor() {
    return cor;
}

public void setCor(String cor) {
    this.cor = cor;
}

public String getPlaca() {
    return placa;
}

public void setPlaca(String placa) {
    this.placa = placa;
}

public int getNum_portas() {
    return num_portas;
}

public void setNum_portas(int num_portas) {
    this.num_portas = num_portas;
}
```

# Construtor

- No exemplo anterior, os valores dos atributos são passados:
  - Diretamente: quando os atributos são públicos
  - Indiretamente: pelos métodos set, caso os atributos sejam privados
- E se quisermos passar os valores dos atributos no momento da instanciação do objeto?

# Construtor

- Método utilizado para a construção do objeto
- Quando criamos uma classe, um construtor é feito por padrão
- Se nenhum construtor for definido, instanciamos um objeto da seguinte forma

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

# Construtor

- Método utilizado para a construção do objeto
- Quando criamos uma classe, um construtor é feito por padrão

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

- Para criarmos construtores, utilizamos o nome da própria classe:

```
public Carro(String tipo, String cor, String placa, int num_portas) {
    this.tipo = tipo;
    this.cor = cor;
    this.placa = placa;
    this.num_portas = num_portas;
}
```

# Construtor

- Implementação de um construtor

```
public Carro(String tipo, String cor, String placa, int num_portas) {
    this.tipo = tipo;
    this.cor = cor;
    this.placa = placa;
    this.num_portas = num_portas;
}
```

- Instanciando um objeto passando os valores dos atributos no construtor:

```
Carro c2 = new Carro("Porshe", "Azul", "LAM-9087", 2);
```

# Construtor

- **OBS:** quando criamos um construtor com parâmetros e queremos manter o construtor sem parâmetros, precisamos implementar os dois construtores -> **sobrecarga de métodos**

```
public Carro() {

}

public Carro(String tipo, String cor, String placa, int num_portas) {
    this.tipo = tipo;
    this.cor = cor;
    this.placa = placa;
    this.num_portas = num_portas;
}
```

# Construtor

- Podemos criar diferentes construtores passando diferentes conjuntos de parâmetros.
- **Criando objetos:** podemos criar utilizando ambos os construtores

```
public static void main(String [] args) {
    Carro carro1 = new Carro("Corolla", "Cinza", "OUG00909", 4);

    Carro carro2 = new Carro();
}
```

# Exercícios

4. Crie uma classe Retangulo em Java, que possui os seguintes atributos privados:
    - Lado1
    - Lado2

A classe deve possui os seguintes métodos:

- Construtor sem parâmetros;
- Construtor que inicializa os atributos lado1 e lado 2;
- Métodos de acesso aos atributos (getters e setters)
- calculaArea()
- calculaPerimetro()

# Exercícios

5. Escreva uma classe Java para representar as contas dos clientes de um banco. Essa classe deve conter os seguintes atributos:
    1. Tipo da conta (corrente, poupança ou salário)
    2. Número da conta
    3. Nome do Cliente
    4. Saldo

Essa conta deve conter os seguintes métodos (além de 2 construtores e os métodos de acesso aos atributos):

- Sacar (valor): verifica se a conta tem saldo e subtrai o valor a ser sacado do saldo atual;
- Depositar (valor): adiciona o valor passado por parâmetro ao saldo;
- Imprimir(): imprime todos os dados da conta .

# Exercícios

6. Crie uma classe Funcionário com atributos nome, sobrenome, horas trabalhadas e valor hora. A classe deve ter os seguintes métodos:
   - Dois construtores (sem e com parâmetros)
   - Métodos de acesso aos atributos (getter e setters)
   - nomeCompleto(): retorna uma string com os atributo nome e sobrenome concatenados.
   - calcularSalario(): calcula e retorna o salário do funcionário no mês, multiplicando o atributo horasTrabalhadas pelo atributo valorPorHora.
   - incrementarHoras(valor): adiciona um valor passado por parâmetro ao valor já existente no atributo valorPorHora.
   - decrementaHoras (valor): subtrai um valor passado por parâmetro ao valor já existente no atributo valorPorHora
   - aumentaValorHora(porcentagem): reajusta o valor da hora de acordo com a porcentagem passada por parâmetro e retorna o valor reajustado.

# Exercícios

7. Escreva uma classe em Java que representa um voo que acontece em determinada data e em determinado horário. Cada voo possui no máximo 100 passageiros, e a classe permite controlar a ocupação das vagas. A classe deve ter os seguintes métodos:

- Dois construtores (com e sem parâmetros);
- Métodos de acesso aos atributos (getters e setters);
- proximoLivre(): retorna o número da próxima poltrona livre. O método deve retornar -1 caso o avião esteja lotado;
- verificaPoltrona(numero): que recebe o número de uma cadeira como parâmetro e informa se a cadeira está livre ou ocupada;
- reservaPoltrona(): que reserva a próxima cadeira livre e retorna o número da cadeira ocupada. O método deve retornar -1 caso o avião esteja lotado;
- quantidadePoltronasVagas(): retorna a quantidade de poltronas disponíveis no voo.
- quantidadePoltronasOcupadas(): retorna a quantidade de poltronas ocupadas no voo.

# Exercício desafio

6. Escreva uma classe que represente um país, que deve conter os seguintes parâmetros: código (ex.: BRA), nome, população e a sua dimensão em Km2. Adicionalmente, cada país possui uma lista de outros países com os quais ele faz fronteira. A classe deve ter os seguintes métodos:

   - Dois construtores:  um sem e outro com parâmetros
   - Métodos de acesso (getter/setter) para os atributos;
   - Um método que permita verificar se dois objetos representam o mesmo país. Dois países são iguais se tiverem o mesmo código;
   - Um método que informe se outro país faz fronteira com o pais modelado;
   - Um método que receba um país como parâmetro e imprime o nome dos países vizinhos comuns aos país modelado.

Considere que um país tem no máximo 5 outros países com os quais ele faz fronteira.