APRESENTAÇÃO
DE APOIO

PROGRAMAÇÃO ORIENTADA A OBJETOS

Alessandro Valério – Aula 01

PUCRS online uol edtech

Professores

ALESSANDRO VALÉRIO DIAS
Professor Convidado

Mestre em Administração e Negócios pela PUCRS, possui também os títulos de especialista em duas áreas: Gerenciamento de Projetos de Tecnologia da Informação e Informática na Educação. É bacharel em Ciência da Computação e Psicologia pela Pontifícia Universidade Católica do Rio Grande do Sul. Atualmente, é analista de sistemas da Pontifícia Universidade Católica do Rio Grande do Sul e professor dos cursos de Análise e Desenvolvimento de Sistemas e de Ciência da Computação do Centro Universitário UniRitter.

EDSON IFARRAGUIRRE MORENO
Professor PUCRS

Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Atualmente, é professor adjunto pela mesma PUCRS, estando vinculado a Escola Politécnica, sendo responsável por disciplinas da área de hardware para os cursos de Ciência da Computação, e Engenharia da Computação. Adicionalmente, trabalha com a orientação de alunos no desenvolvimento de projetos do curso de Engenharia de Software. Desde 2016, coordena o laboratório iSeed Labs, uma parceria entre a academia e a iniciativa privada que tem por objetivo fomentar a inovação e o empreendedorismo. Seus principais temas de pesquisa incluem: sistemas multiprocessados em chip (Multiprocessor System on Chip, MPSoC), projeto em nível de sistema e redes em chip (Network on Chip, NoC).

Ementa da disciplina

Estudo sobre conceitos de Classes (atributos, métodos, propriedades, visibilidade, instancia ou classe). Estudo de conceitos de Herança, Polimorfismo, Interfaces, Genéricos e Arrow functions. Estudo sobre funções de filtragem, mapeamento e redução. Estudo sobre construtores de tipos.

Pai da Maria Clara

25+ anos em projetos de TI

Cientista da computação

Sempre curti programação / arquitetura

Mas também pessoas / processos

Psicólogo

Professor em cursos de TI e UX

Mestre em Administração e Negócios

Música, RPGs, video games e boardgames

Ementa

Estudo sobre conceitos de Classes (atributos, métodos, propriedades, visibilidade, instância ou classe).

Estudo de conceitos de Herança, Polimorfismo, Interfaces, Genéricos e Arrow functions.

Estudo sobre funções de filtragem, mapeamento e redução.

Estudo sobre construtores de tipos.

Introdução

PROGRAMAÇÃO ORIENTADA A OBJETOS

É UM **PARADIGMA DE PROGRAMAÇÃO**

COM FOCO EM OBJETOS

EM VEZ DE FUNÇÕES

PROGRAMAÇÃO ORIENTADA A OBJETOS

NÃO É UMA LINGUAGEM DE PROGRAMAÇÃO

NÃO É UMA FERRAMENTA OU FRAMEWORK

PROGRAMAÇÃO ORIENTADA A OBJETOS

É UM ESTILO DE PROGRAMAÇÃO

É UM PARADIGMA DE PROGRAMAÇÃO

Motivação

MUITAS LINGUAGENS IMPLEMENTAM

ORIENTAÇÃO A OBJETOS

- C++
- C#
- Java
- JavaScript
- Python
- Ruby

CONHECER PARADIGMAS

É ALGO **ESSENCIAL** PARA

QUALQUER **BOM PROFISSIONAL** DE

DESENVOLVIMENTO DE SOFTWARE

Contextualização

PROGRAMAÇÃO ESTRUTURADA

Paradigma com foco em

Sequência

Uma instrução executada após a outra

Decisão

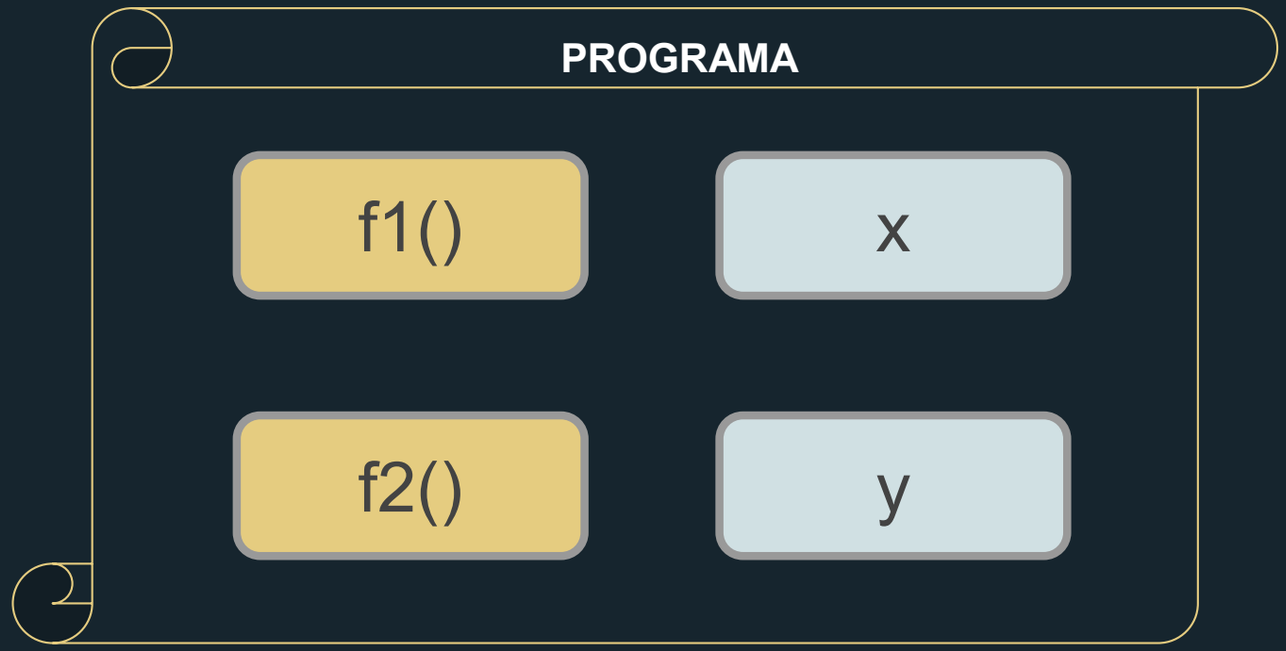
Uma instrução executada após algum teste lógico

Iteração

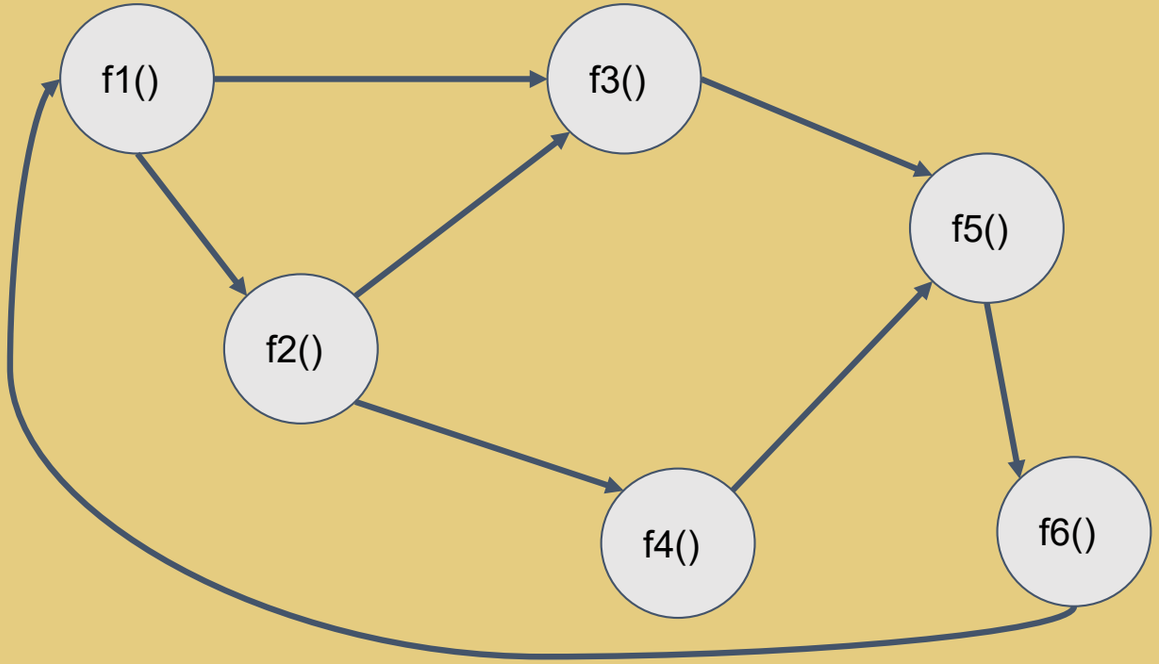
Uma trecho de código pode repetir após algum teste lógico

PROGRAMAÇÃO PROCEDURAL

Paradigma com foco no uso de procedimentos e funções para facilitar o reuso



PROGRAMAÇÃO ESTRUTURADA E PROCEDURAL



PROGRAMAÇÃO ESTRUTURADA E PROCEDURAL

Muito “copia e cola” de funções

Mudanças em uma função resultam em mudanças em outras funções

Interdependência entre funções

PROGRAMAÇÃO PROCEDURAL

```
let numero1 = 10;
let numero2 = 20;

function add(n1, n2) {
  return n1 + n2;
}

let z = add(numero1, numero2);
console.log(z);
```

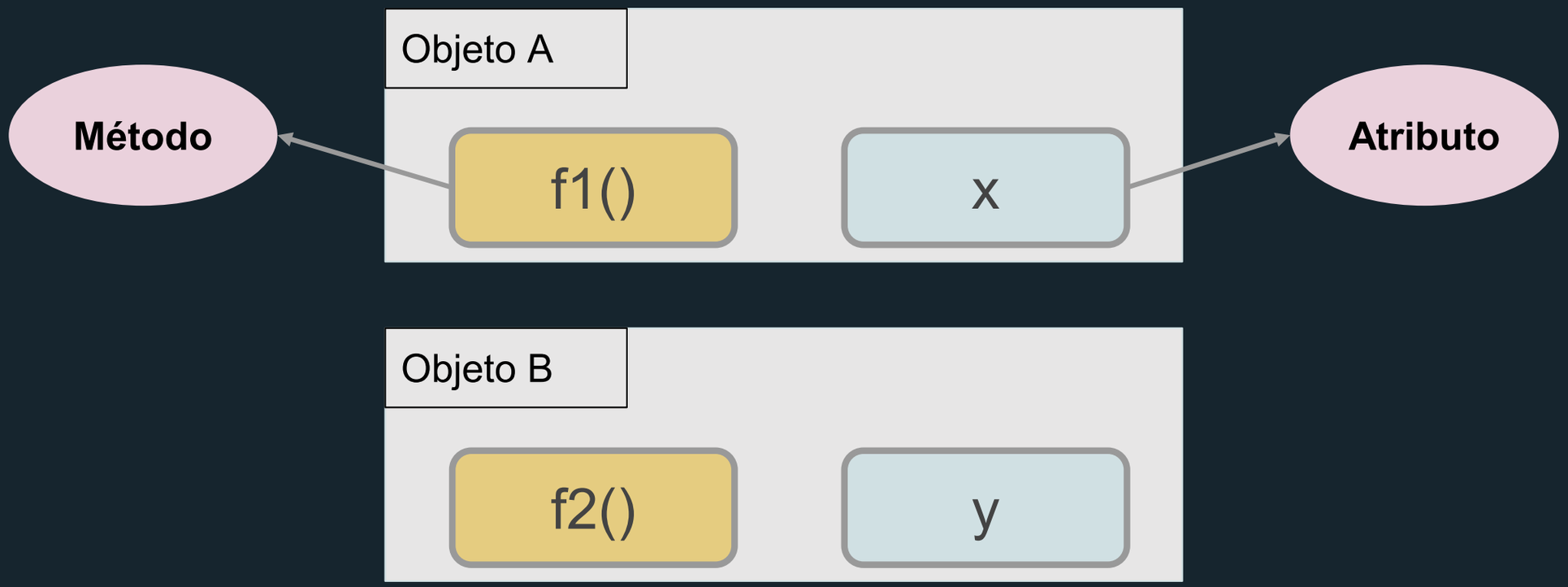
```
let horaExtra = 100;
let qtHoras = 20;
let salarioFixo = 5000;

function calculaSalario(horaExtra, qtHoras) {
  return salarioFixo + (horaExtra * qtHoras);
}

let total = calculaSalario(horaExtra, qtHoras);
console.log(total);
```

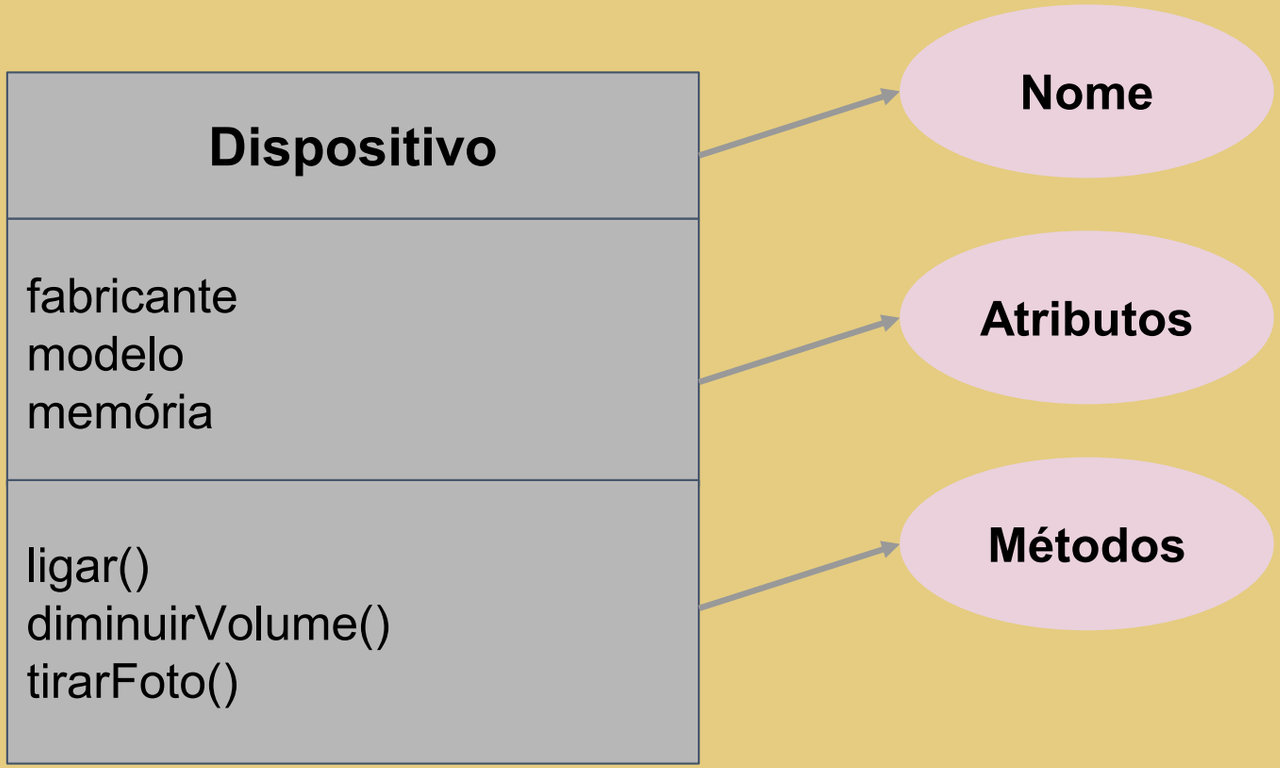
PROGRAMAÇÃO ORIENTADA A OBJETOS

Paradigma com foco no uso de objetos, onde cada um contém suas variáveis e funções



OBJETOS - COMPONENTES

PROGRAMAÇÃO ORIENTADA A OBJETOS



OBJETO

coleção de **dados** e/ou **funcionalidades** com alguma relação entre si

dados -> **variáveis** (**atributos** ou propriedades)

funcionalidades -> **funções** (**métodos**)

OBJETO

em *JavaScript*, atributos e métodos são **membros** de um objeto, cada um com um nome e um valor

```
var nomeDoObjeto = {
  nomeMembro1: valorMembro1,
  nomeMembro2: valorMembro2,
  nomeMembro3: valorMembro3
};
```

OBJETO

em *JavaScript*, atributos e métodos são **membros** de um objeto, cada um com um nome e um valor

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};
```

ATRIBUTO ou PROPRIEDADE

UM OU MAIS DADOS PRESENTES EM UM OBJETO

POSSUEM UM **NOME ÚNICO**

ARMAZENAM UM **VALOR** OU **REFERÊNCIA**

ACESSO

DUAS FORMAS

OBJETO.PROPRIEDADE

OBJETO [“PROPRIEDADE”]

OBJETO.PROPRIEDADE

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};

console.log(pessoa.nome);
```

OBJETO [“PROPRIEDADE”]

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};

console.log(pessoa[“nome”]);
```

ATRIBUIÇÃO

ACESSÍVEIS DE DUAS FORMAS

OBJETO.PROPRIEDADE = ALGO

OBJETO [“PROPRIEDADE”] = ALGO

OBJETO.PROPRIEDADE

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};

pessoa.idade = 30;
console.log(pessoa.idade);
```

OBJETO [“PROPRIEDADE”]

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};

pessoa[“idade”] = 30;
console.log(pessoa[“idade”]);
```

MÉTODO

UMA OU MAIS FUNCIONALIDADES PRESENTES EM UM OBJETO

POSSUEM UM NOME ÚNICO

REPRESENTAM UMA LÓGICA PERTINENTE AO OBJETO

ACESSO

UMA FORMA

OBJETO.METODO()

OBJETO.METODO(*parametro*)

OBJETO.METODO(*parametro1*, *parametro2*, *...*)

OBJETO.METODO()

```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function() {
      console.log(“Olá”);
      }
};

console.log(pessoa.saudar());
```

OBJETO.METODO() COM PARÂMETRO

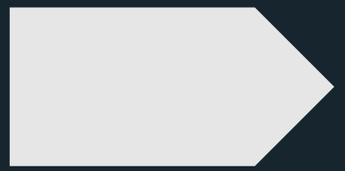
```
var pessoa = {
  nome: “Valentina”,
  idade: 60,
  saudar: function(nomeAmigo) {
      console.log(“Olá ” + nomeAmigo);
      }
};

console.log(pessoa.saudar(“Joana”));
```

CRIANDO OBJETOS - PRIMEIRA FORMA

Podemos criar objetos em JavaScript de **forma literal**:

```
var nomeDoObjeto = {
  nomeMembro1:
valorMembro1,
  nomeMembro2:
valorMembro2,
  nomeMembro3: valorMembro3
};
```



```
var pessoa = {
  nome: [‘Fulano’, ‘de Tal’],
  anoDeNascimento: 1990,
  profissao: ‘Estudante‘,
  calculaIdade: function() {
    return new Date().getFullYear() - this.anoDeNascimento;
  }
};
```

COMPARANDO POO & PE

```
let numero1 = 10;
let numero2 = 20;

function add(n1, n2) {
  return n1 + n2;
}

let z = add(numero1, numero2);
console.log(z);
```

```
const calculadora = {
  numero1: 10,
  numero2: 20,
  soma: function() {
    return this.numero1 + this.numero2;
  }
};

console.log(calculadora.soma());
```

COMPARANDO POO & PE

```
let horaExtra = 100;
let qtHoras = 20;
let salarioFixo = 5000;

function calculaSalario(horaExtra, qtHoras) {
  return salarioFixo + (horaExtra * qtHoras);
}

let total = calculaSalario(horaExtra, qtHoras);
console.log(total);
```

```
const empregado = {
  salarioFixo: 5000,
  valorHoraExtra: 100,
  horasExtras: 20,
  calculaSalario: function() {
    return this.salarioFixo +
    (this.valorHoraExtra * this.horasExtras);
  }
};

console.log(empregado.calculaSalario());
```

Há maneiras melhores e mais elegantes de criarmos objetos

“Em JavaScript **quase tudo** é objeto”

Mozilla Web Docs, 2022

FUNÇÕES SÃO OBJETOS

DADOS COMPLEXOS SÃO OBJETOS

ESTRUTURAS DE DADOS SÃO OBJETOS

E O RESTANTE?

TIPOS DE DADOS EM JAVASCRIPT

JavaScript possui dois tipos de dados

TIPOS DE VALOR

TIPOS DE REFERÊNCIA

TIPOS DE VALOR

representações de valores imutáveis

Number
String
Boolean
Symbol
undefined
null

TIPOS DE REFERÊNCIA

representações de valores mutáveis e complexos quando comparados com os Tipos de Valor

Object

(Object, Function, Array, …)

Mais exemplos de Objetos

Orientação a objetos

Conceitos principais

CONCEITOS

Encapsulamento

Abstração

Herança

Polimorfismo

CONCEITOS - ENCAPSULAMENTO

- Encapsulamento
- Abstração
- Herança
- Polimorfismo

Permitir que atributos e métodos sejam **agrupados** de certa forma em uma **interface bem definida** para manipular os dados de um objeto de forma eficiente

Isolamento entre partes de um programa

Saber **o que um objeto faz** e não como ele faz

Proteção de atributos

CONCEITOS - ENCAPSULAMENTO

```
const empregado = {
  salarioFixo: 5000,
  valorHoraExtra: 100,
  horasExtras: 20,
  calculaSalario: function() {
    return this.salarioFixo + (this.valorHoraExtra * this.horasExtras);
  }
};

console.log(empregado.calculaSalario());
```

CONCEITOS - ABSTRAÇÃO

- Encapsulamento
- Abstração
- Herança
- Polimorfismo

POO é **amplamente baseada** na abstração digital da vida real

Objetos são representações/abstrações do que queremos implementar do que observamos no mundo ao nosso redor

Buscamos o **essencial** e deixamos de lado o que não importa, focamos no que realmente precisamos representar em nossos objetos

CONCEITOS - ABSTRAÇÃO

Carro
fabricante modelo potência velocidade atual
ligar() acelerar() frear()

CONCEITOS - HERANÇA

- Encapsulamento
- Abstração
- Herança
- Polimorfismo

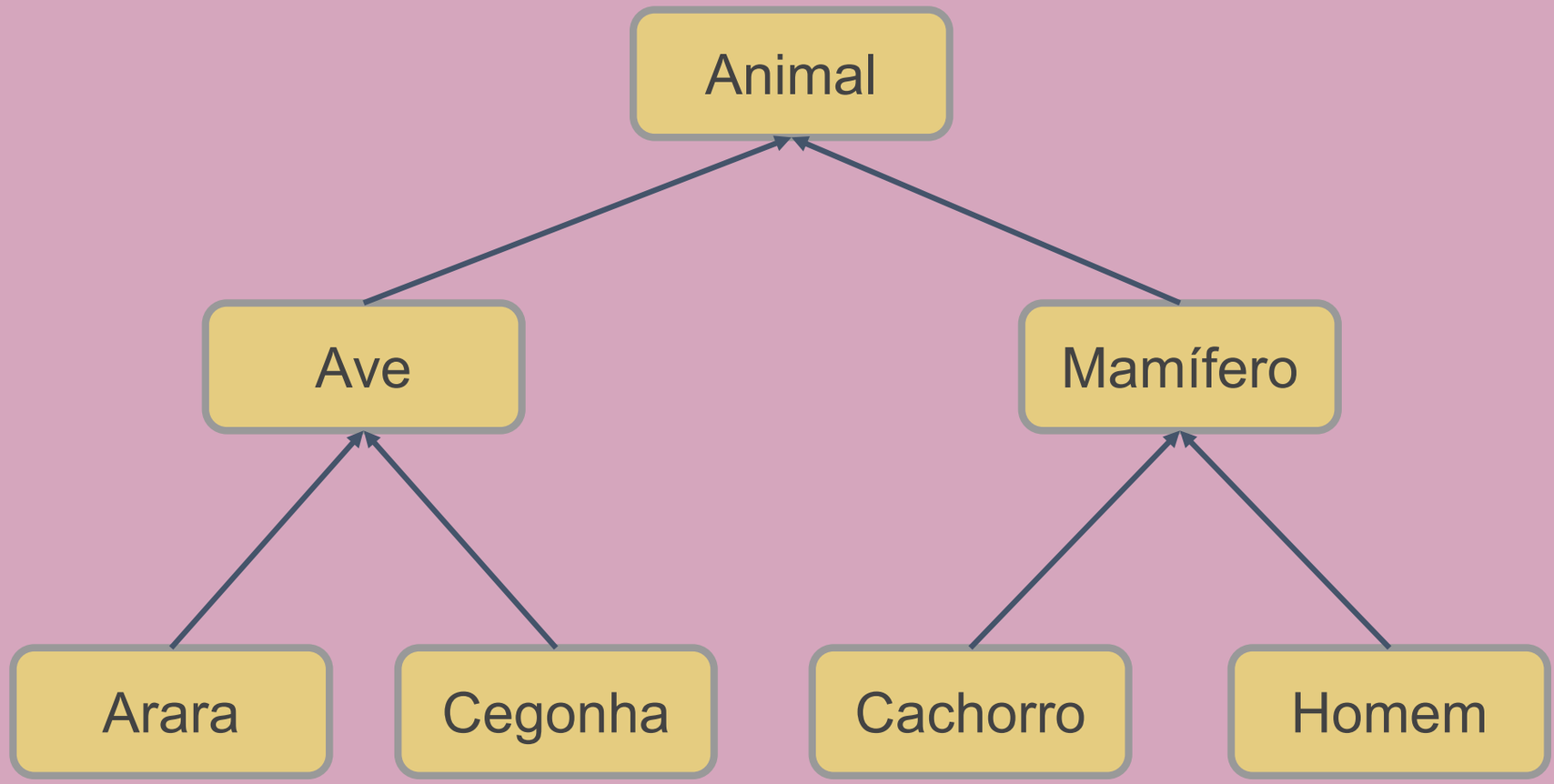
Permite o **compartilhamento** de atributos e métodos entre objetos

Reaproveita código e agrupa o que é comum a diferentes objetos

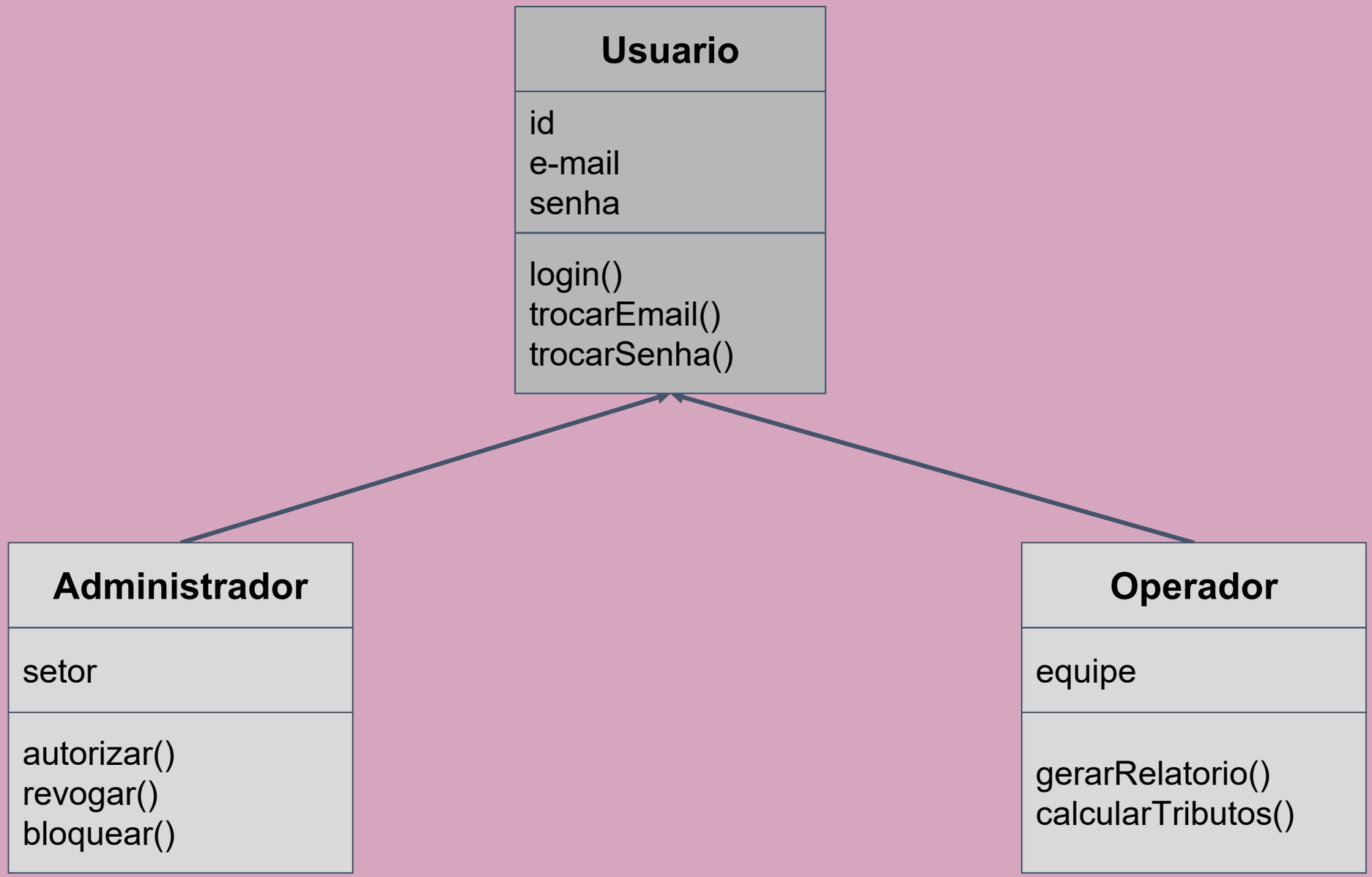
Busca identificar e agrupar **comportamentos generalizados** ou **especializados**

Ajuda a eliminar **redundâncias**

CONCEITOS - HERANÇA



CONCEITOS - HERANÇA



CONCEITOS - POLIMORFISMO

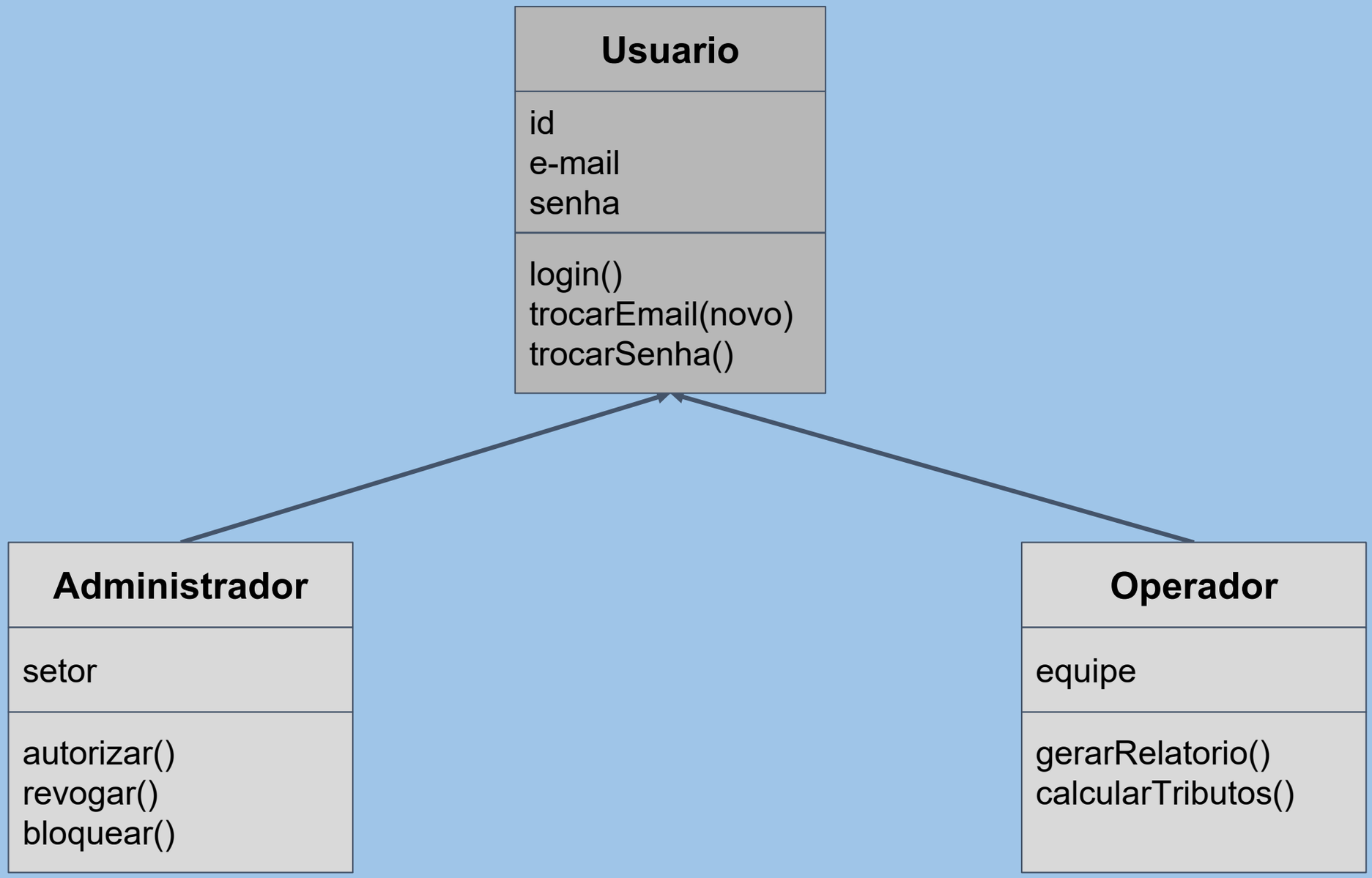
- Encapsulamento
- Abstração
- Herança
- Polimorfismo

Através da herança é possível **alterar um comportamento herdado** de um objeto-pai

Permite uma forte **separação de interesses**

Limpeza de código, removendo lógica excedente

CONCEITOS - POLIMORFISMO

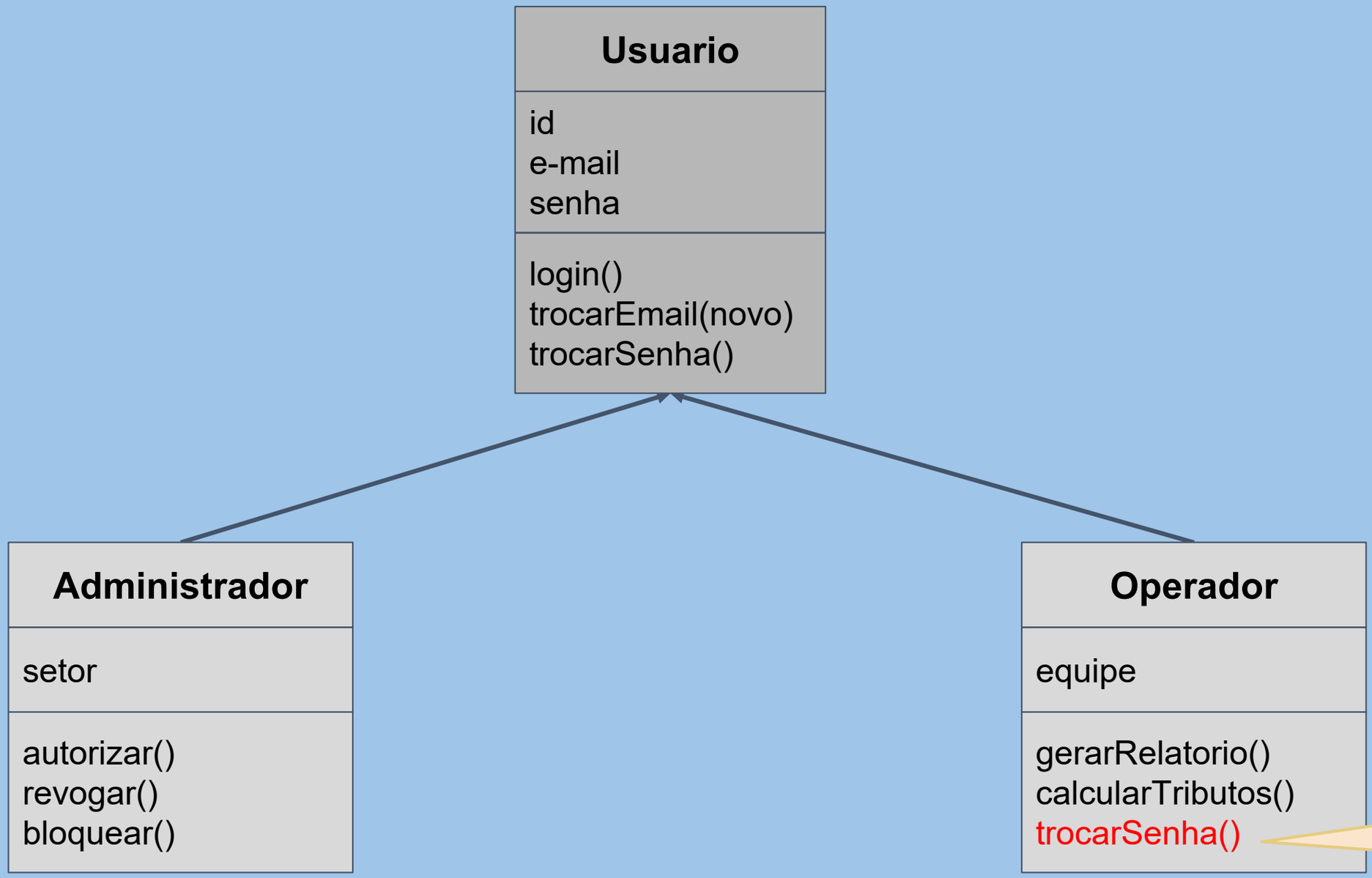


SOBRESCRITA

métodos com o mesmo nome e assinatura

em diferentes objetos relacionados por **herança**

CONCEITOS - POLIMORFISMO



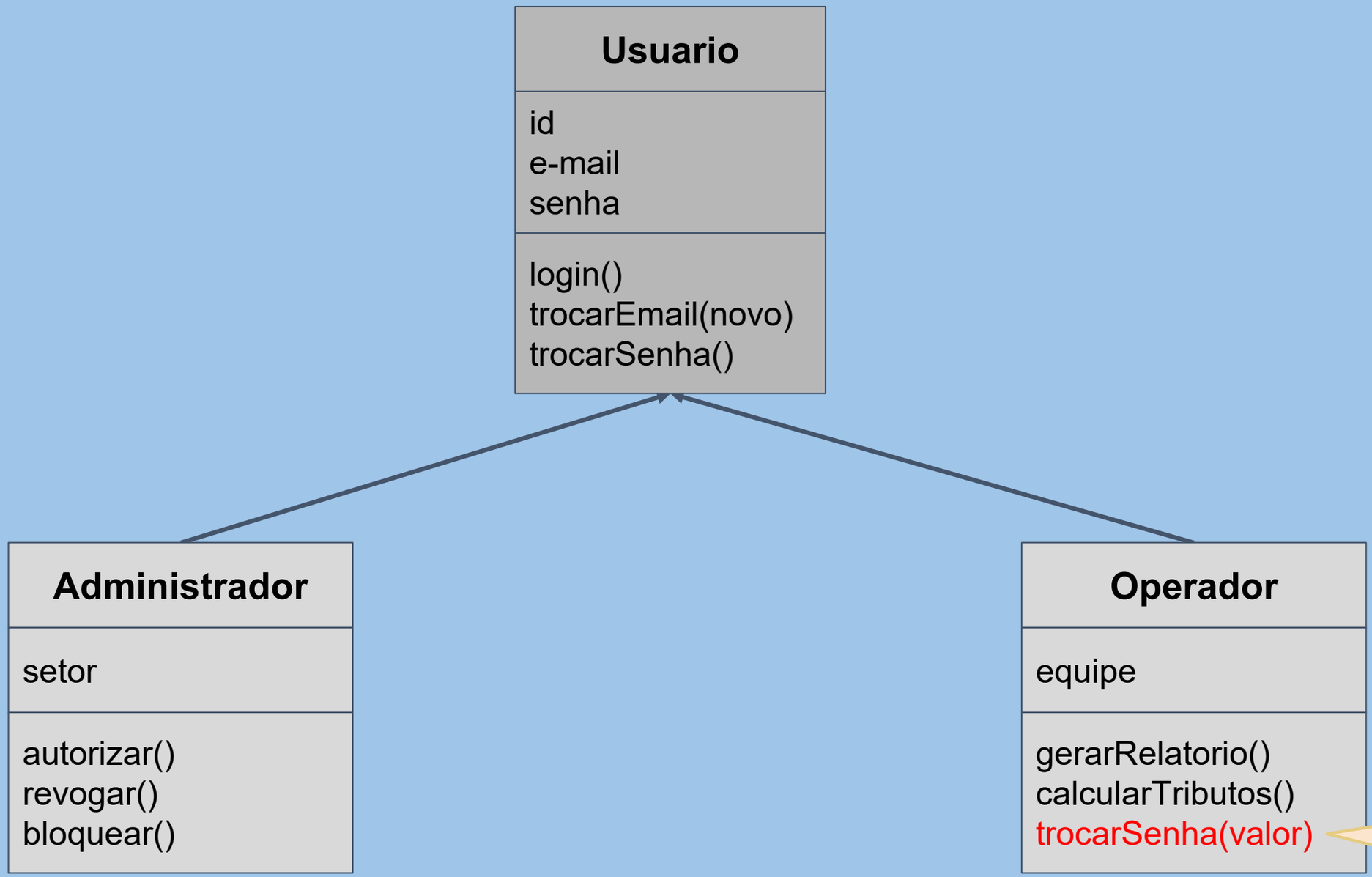
SOBRECARGA

métodos com o mesmo nome

mas **diferentes assinaturas**

em diferentes objetos relacionados por **herança**

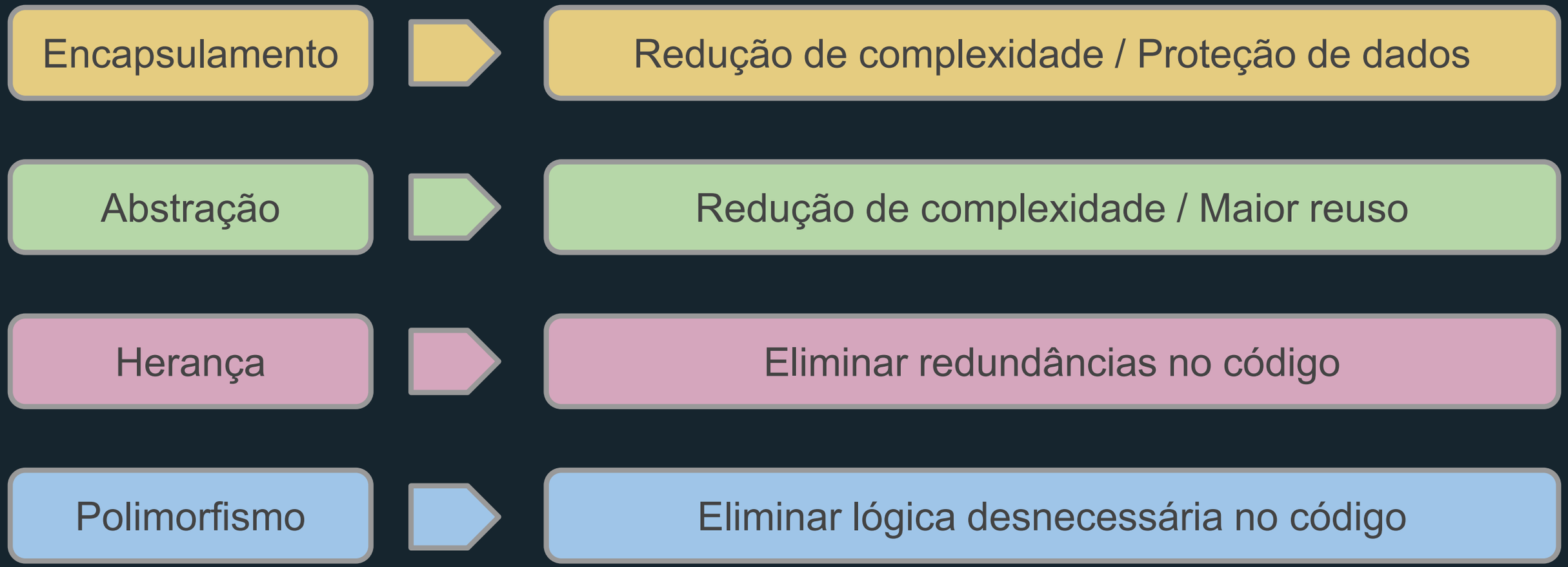
CONCEITOS - POLIMORFISMO



JAVASCRIPT

NÃO SUPORTA SOBRECARGA

CONCEITOS - VANTAGENS E GANHOS



Orientação a objetos

Trabalhando com objetos

CRIANDO OBJETOS

CRIANDO OBJETOS

LITERAIS

FÁBRICAS

CONSTRUTORES

PROTÓTIPOS

CLASSES

PUCRS online uol edtech