

# **INTRODUÇÃO À ORIENTAÇÃO A OBJETO**

**por Rodrigo Sieja Bertin**

## **Índice**

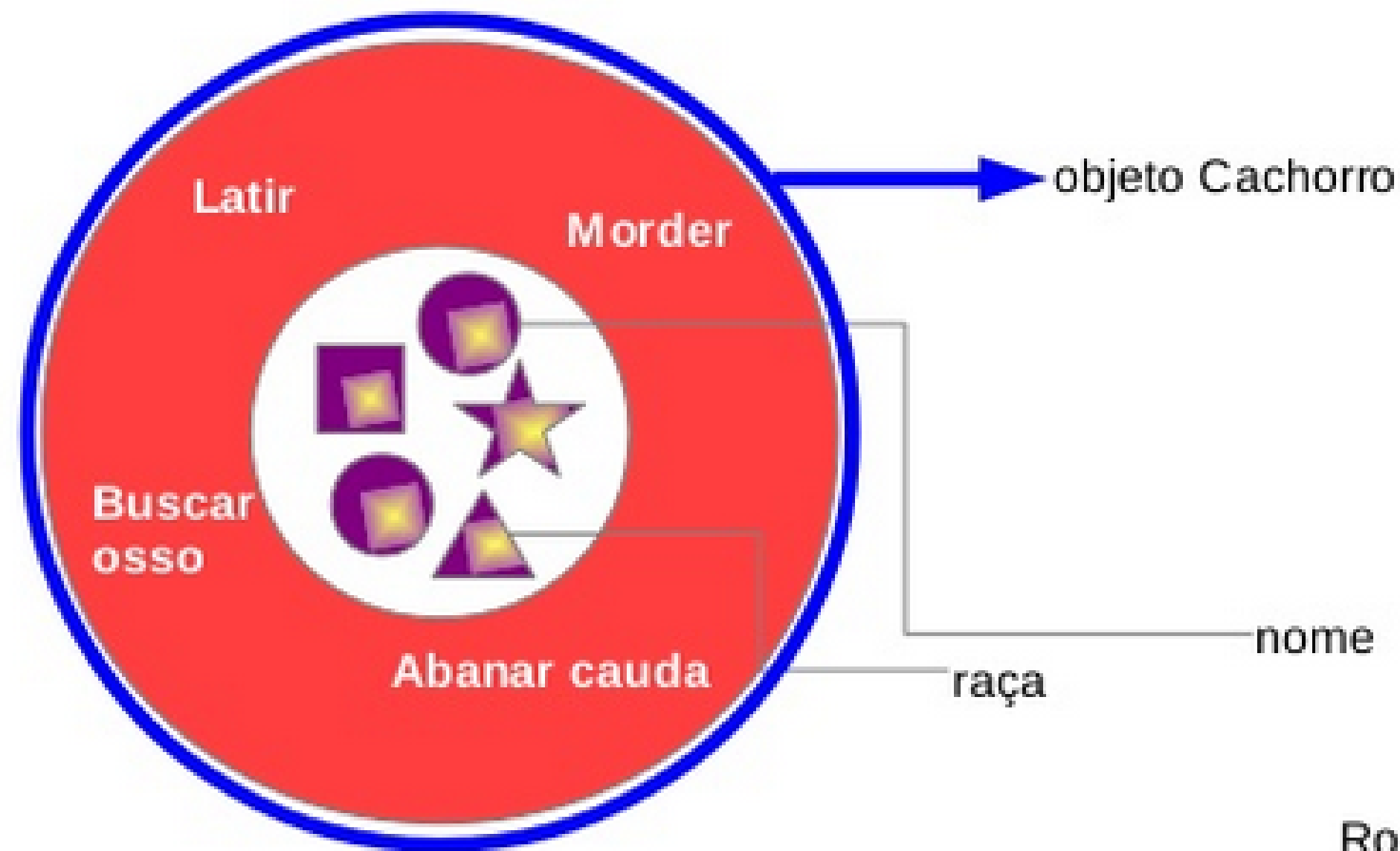
- Objeto
- Operações e métodos
- Abstração
- Classe
- Encapsulamento
- Construtor
- Herança
- Polimorfismo
- Interface
- Vantagens
- Exemplos em Java e C#

# Objeto

- Promove o entendimento do mundo real
- Todos os objetos são distinguíveis e possuem uma identidade própria
- Objetos no mundo real tem 2 características:
  - Estado (***atributos***) e comportamento (***métodos***)

# Exemplo – objeto Cachorro

- Um cachorro possui os atributos “nome”, “cor”, “raça”, “faminto”, etc e os métodos “latir”, “morder”, “abandar cauda”, etc



## Exemplo – objeto Bicicleta

- Uma bicicleta possui os atributos: marcha, cor, velocidade, etc... E os métodos: mudar marcha, frear, acelerar, etc
- Cada bicicleta contém os mesmos componentes
- Sua bicicleta é uma *instância* da classe de objetos conhecida como Bicicleta

# Persistência de um objeto

- É o tempo que este permanece armazenado na memória
- Um objeto é persistente quando os valores de seus atributos e as informações necessárias para a manutenção de suas conexões c/ outros objetos são armazenados em uma mídia
- Objetos criados e destruídos sem que sejam salvos são chamados de não-persistentes

# Operações e métodos

- Operação é uma transformação, ou função, que pode ser aplicada a objetos de uma classe ou por esses objetos
  - Emissão de NF → p/ fax, e-mail ou impressora?
- Método é uma implementação de uma operação p/ uma classe específica
- Um método possui uma assinatura constituída pelo nome de uma operação, o número, tipo e ordem de seus argumentos pelo valor de retorno

# Abstração

- Ocorre quando nos focamos nos aspectos essenciais de uma entidade e ignoramos propriedades secundárias
- Concentrar-se primeiro no que o objeto é e o que ele executa, para compreender o problema
- Isola-se o objeto do ambiente e se representa nele apenas características relevantes

# Classe – definição

- Representa um conjunto de objetos com a mesma estrutura
- Projeto do qual objetos individuais são criados
- Classes devem representar uma abstração:
  - Exemplo: o elevador fecha a sua porta e depois se move para outro andar



# Classe – instâncias

- "Animal" é uma classe, um molde, já "cachorro" é uma instância de "Animal"
- Carrega todas as características do molde
- Independe das outras instâncias
- Cada instância da classe tem valores diferentes p/ seus respectivos atributos

# Classe abstrata

- Classe a partir da qual não é possível realizar qualquer tipo de instância
- São feitas p/ servir de modelo a classes derivadas (*classes concretas*), que sobrescreverão os métodos p/ realizar a implementação dos mesmos

# Encapsulamento

- Dados e operações são *encapsulados* em uma única entidade
- A única forma de conhecer ou alterar seus atributos é por meio de seus métodos
- Não há necessidade de conhecer o funcionamento interno do objeto
- É possível modificar um objeto internamente, adicionando métodos, sem afetar os outros componentes

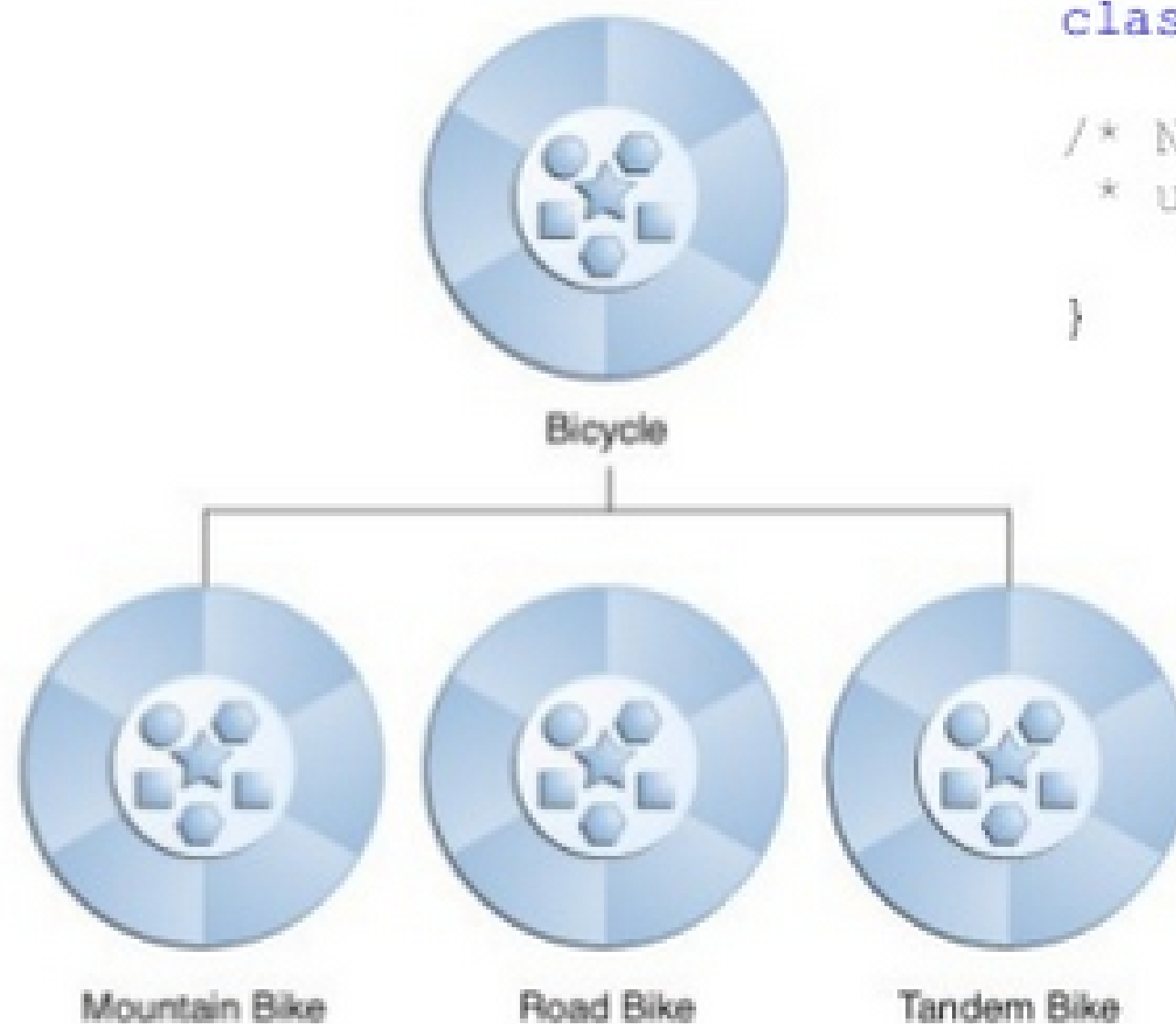
# Construtor

- É um método chamado depois que a instância de um objeto é criada
- Deve ter o mesmo nome da classe
- Não tem tipo de retorno
- Toda classe tem pelo menos um construtor
- Pode usar qualquer modificador de acesso
- O construtor padrão não tem argumentos
- Não pode ser herdado nem substituído

# Herança

- Diferentes objetos possuem algum grau de semelhança
- *As subclasses*, como MountainBike, herdam as características da *superclasse* (Bicicleta)
- Palavra-chave **extends**

# Herança



```
class MountainBike extends Bicicleta {  
    /* Novos campos e métodos que definem  
    * uma mountain bike */  
}
```

# Herança simples

- Determina que uma classe herdará características de apenas uma superclasse.

# Herança múltipla

- Quando uma classe herda características de duas ou mais superclasses
- Faz-se a prefixação do nome do membro com o nome da classe base a que ele se refere
  - Esse processo é chamado de qualificação

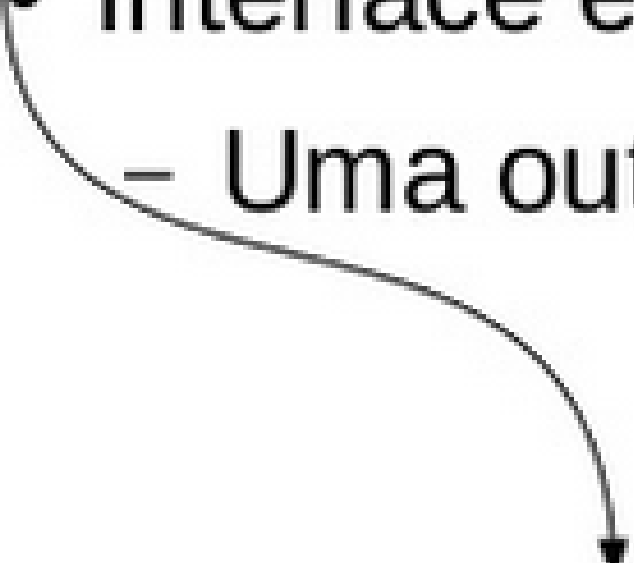


# Polimorfismo / método polimórfico

- Método que se comporta de forma específica para cada uma das classes derivadas
- Os mesmos atributos e objetos podem ser utilizados em objetos distintos, mas c/ implementação lógica diferente
  - Classe Cliente e Classe Funcionário
  - Superclasse = Pessoa
  - Método “Enviar e-mail” feito de forma diferente para cada classe derivada (Cliente, Funcionário)

# Interface

- Os métodos formam a interface do objeto c/ o mundo exterior
  - Botões do controle remoto são uma interface entre você e os fios elétricos dentro dele
- Interface é um grupo de métodos c/ corpo vazio
  - Uma outra classe implementará a interface:



```
public interface Predador {  
    boolean persegueCaça(Caça  
p);  
    void comeCaça(Caça p);  
}
```

```
public class Leão implements Predador {  
  
    public boolean persegueCaça(Caça p) {  
        /* Programação para perseguir a caça,  
        * especificamente para um leão */  
    }  
  
    public void comeCaça(Caça p) {  
        /* Programação para comer a caça,  
        * especificamente para um leão */  
    }  
}
```

# Vantagens

- Única abordagem em todas as etapas
- Maior nível de abstração
- Componentes modularizados
- Manutenção facilitada
- Reaproveitamento de código
- Desenvolvimento mais produtivo
- Redução de custo de desenvolvimento

# Exemplo prático – classe Bicicleta

(Java)

```
class Bicicleta {                                     // Classe

    /* Atributos */
    int cadencia = 0;
    int velocidade = 0;
    int marcha = 1;

    void mudaCadencia(int novoValor) {
        cadencia = novoValor;
    }
    void mudaMarcha(int novoValor) {
        marcha = novoValor;
    }
    void aumentaVelocidade(int incremento) {
        velocidade = velocidade + incremento;
    }
    void breca(int decremento) {
        velocidade = velocidade - decremento;
    }
    void exibeEstados() {
        System.out.println("Cadência: " + cadencia    // Exibe na tela
        + "\nVelocidade: " + velocidade
        + "\nMarcha: " + marcha);
    }
}
```

# Exemplo prático – classe Animal

(C#)

```
public class Animal // Classe
{
    protected string especie; // Atributo

    public Animal (string str_especie)
    {
        str_especie = especie;
    }

    static void Main(string[] args)    // Execução
    {
        /* Instâncias */
        cachorro = new Animal("Canis lupus familiaris");
        gato = new Animal("Felis catus");
        lobo = new Animal("Canis lupus");
    }
}
```