
# Programação I

# Interfaces

Jorge Roberto Trento
Bacharel em Ciências da Computação - UNOESC
Especialização em Ciências da Computação - UFSC
Formação Pedagógica - Formadores de Educação Profissional – UNISUL
Especialização em Ensino Superior – FIE
Especialização em Gestão de Tecnologia da Informação – FIE

# Introdução


- Uma interface é uma espécie de "classe totalmente abstrata", na qual todos os métodos são implicitamente públicos e abstratos e todos os atributos são implicitamente static final.
- Interfaces são criadas para fornecer serviços comuns para classes não relacionadas.
- Assim, podemos usar uma interface para amarrar elementos de várias classes em um conjunto com comportamento uniforme, mesmo que estas classes não possuam nenhum relacionamento.
- Uma interface é declarada através da palavra **interface** no lugar de **class**. Embora não represente exatamente uma classe, deve ser gravada em um arquivo com extensão .java e compilada.

# Introdução

```
public interface MinhaInterface {
    // atributos estáticos constantes
    // assinatura dos métodos
}
```

Para que uma classe possa usar uma interface, é necessário que a classe implemente a interface com o uso da palavra reservada **implements**:

```
public class MinhaClasse implements MinhaInterface {
    // atributos específicos da classe
    // métodos específicos da classe
    // métodos da interface implementados
}
```

# Interface


- Uma interface é como um contrato: se uma classe implementa uma interface, então ela deve, obrigatoriamente, implementar todos os seus métodos (pois são todos abstratos).
- Uma interface não pode ser instanciada, mas podem ser definidas referências do seu tipo:
  - MinhaInterface ref; // definindo uma referência sem instanciar objeto
- OBS: Não confundir com interface gráfica.

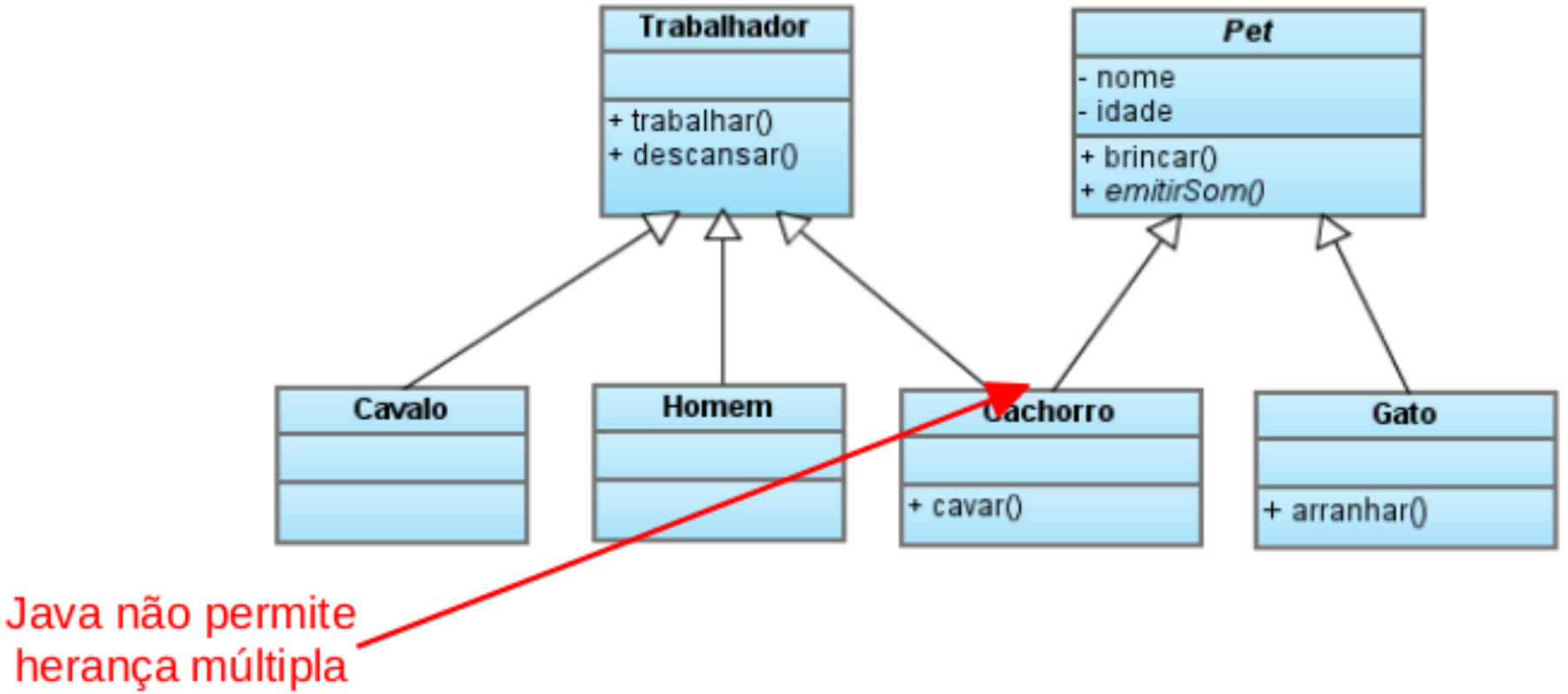
# Interface


- Interfaces podem ser herdadas (ou seja, pode-se ter uma sub-interface). Uma interface pode estender mais de uma interface, diferentemente de uma classe, que pode estender somente uma classe:
  - public interface MinhaInterface extends OutraInterfaceX, OutraInterfaceY

# Exemplo


- Suponha que queremos representar o comportamento de uma classe Trabalhador através dos métodos trabalhar() e descansar() .
- Poderíamos fazer com que todos animais que trabalham (Homem, Cachorro, Cavalo, etc) fossem subclasses de Trabalhador e herdassem estes 2 métodos.
- No entanto, se uma classe já é subclasse de outra (como Cachorro , que já é subclasse de Pet ), ela não poderá estender também outra classe. Em Java, não é permitido que uma classe herde de duas ou mais classes (herança múltipla).

# Exemplo


Trabalhador
+ trabalhar()
+ descansar()
Pet
- nome
- idade
+ brincar()
+ emitirSom()
Cavalo
Homem
Cachorro
+ cavar()
Gato
+ arranhar()
Java não permite
herança múltipla

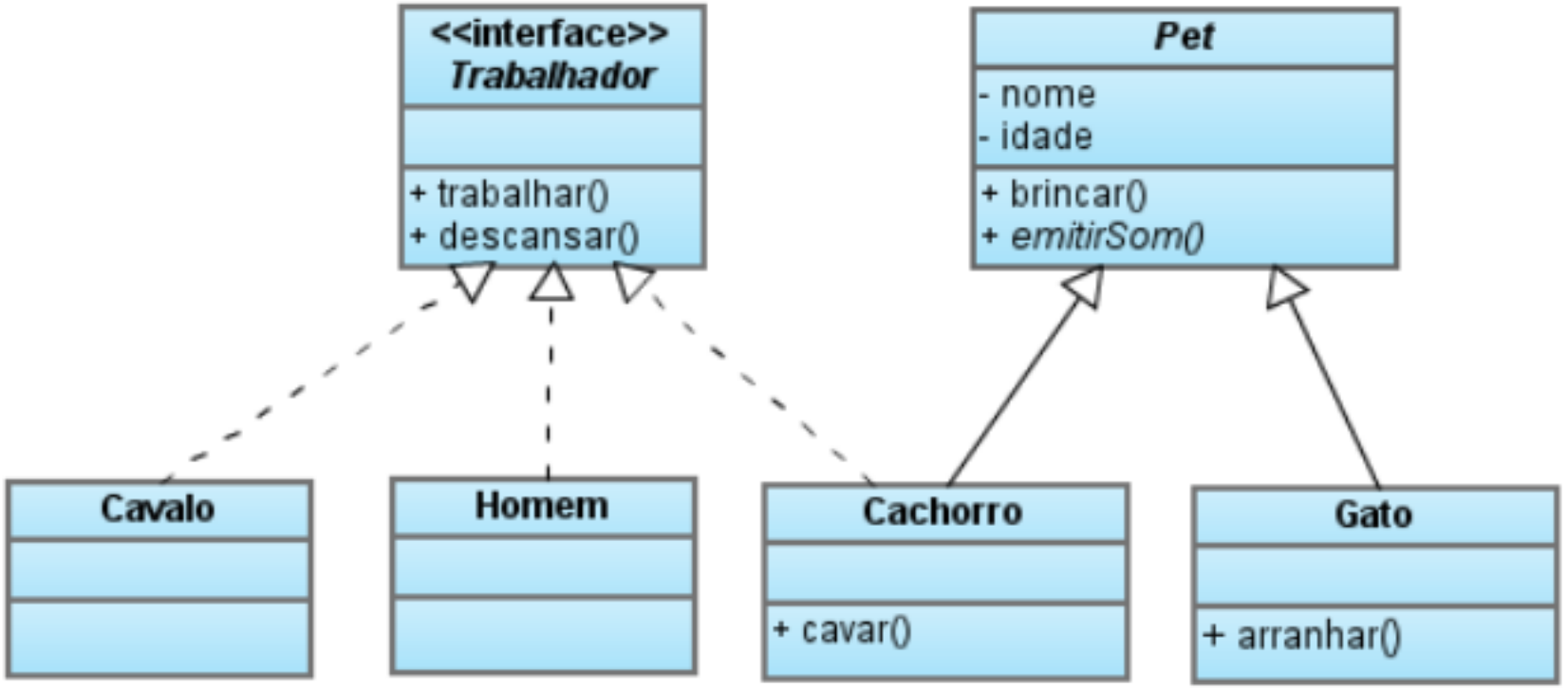
# Exemplo


- Colocar os métodos trabalhar() e descansar() na classe Pet também não é uma boa solução, uma vez que Cavalo e Homem também são trabalhadores, mas não são tipos de Pet.
- A solução é definir Trabalhador como uma Interface. Assim, qualquer classe poderá implementá-la, mesmo que não haja relacionamento nenhum entre elas. Ao implementar Trabalhador, as classes serão obrigadas a implementarem os métodos trabalhar() e descansar().
- Assim garante-se a uniformidade de comportamento, ou seja, que todo Trabalhador , independentemente de que tipo seja, tenha esses dois métodos.

# Exemplo


- Cabe ressaltar que uma classe só pode herdar de uma classe, porém, pode implementar quantas interfaces for preciso.
- Uma interface é representada num diagrama UML através do estereótipo <<interface>> antes de seu nome. Da mesma forma que uma classe abstrata, o nome deve aparecer em itálico.
- Para indicar que uma classe implementa uma interface, usa-se uma seta semelhante à usada para herança, porém com linha tracejada.

# Exemplo


<<interface>>
Trabalhador
+ trabalhar()
+ descansar()
Pet
- nome
- idade
+ brincar()
+ emitirSom()
Cavalo
Homem
Cachorro
+ cavar()
Gato
+ arranhar()

# Exemplo


- Veja que não há relação nenhuma de herança entre um cavalo, um homem e um cachorro, portanto são objetos não relacionados. Porém, ainda assim eles apresentam serviços em comum.
- As interfaces também promovem o relacionamento "é-um" com quem a implementa, ou seja, no exemplo acima, "Cavalo é um Trabalhador", "Homem é um Trabalhador" e "Cachorro é um Trabalhador".

# Exemplo – Interface Trabalhador


```
public interface Trabalhador {
    void trabalhar();
    void descansar();
}
```

# Exemplo – Classe Cavalo


```
public class Cavalo implements Trabalhador {
    public void trabalhar(){
        System.out.println("Puxando carroça");
    }
    public void descansar(){
        System.out.println("Cavalo descansando");
    }
}
```

# Exemplo – Classe Cachorro

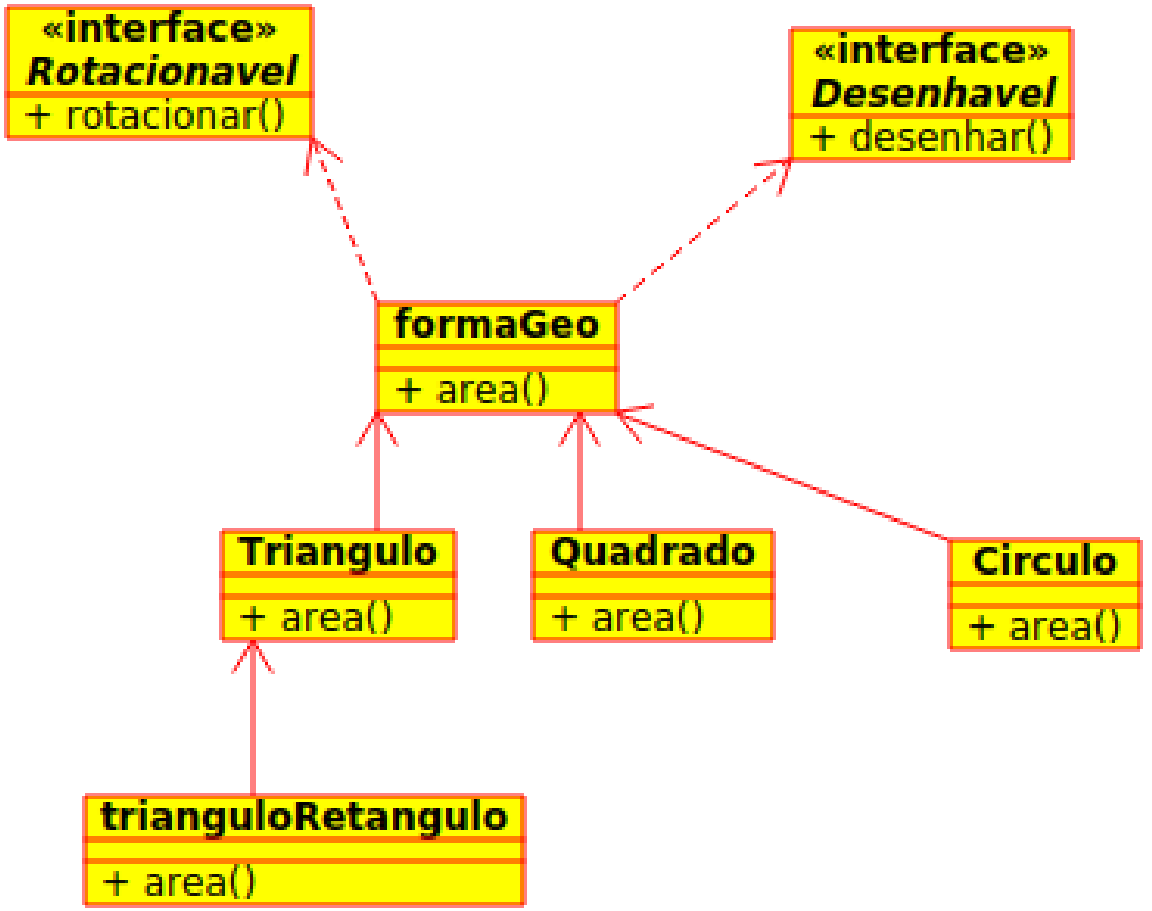

- Já a classe Cachorro deve herdar de Pet (e portanto, sobrepor o método abstrato emitirSom() ) e implementar Trabalhador (e portanto, implementar os métodos trabalhar() e descansar() );

```
public class Cachorro extends Pet implements Trabalhador {
    public void emitirSom(){
        System.out.println("Au au au");
    }
    public void cavar(){
        System.out.println("Cavando");
    }
    public void trabalhar(){
        System.out.println("Cão em serviço");
    }
    public void descansar(){
        System.out.println("Cão descansando");
    }
}
```

# Resumo de Interfaces


- Uma interface não pode ser instanciada.
- Uma interface pode conter apenas atributos constantes, que já são declarados implicitamente como public static final.
- Todos os métodos de uma interface são implicitamente public abstract, ou seja, não possuem corpo.
- Classes abstratas e interfaces podem ser usadas para definir variáveis de referência.

# Exercício


«interface»
Rotacionavel
+ rotacionar()
«interface»
Desenhavel
+ desenhar()
formaGeo
+ area()
Triangulo
+ area()
Quadrado
+ area()
Circulo
+ area()
trianguloRetangulo
+ area()

# Exercício


- Dado o diagrama UML do slide anterior, implemente as classes e interfaces, de modo que dentro do método área de cada classe tenha uma mensagem que indica que está calculando a área de determinada figura geométrica.
- Faça um arquivo de teste que cria um objeto de cada forma geométrica e chama o método área.

# Interfaces

Autor
Prof. Doglas André Finco
doglas.andref@uffs.edu.br

jorge.trento@uffs.edu.br