COTIL

UNICAMP

Linguagem de Programação III

Profa. Simone Berbert Rodrigues Dapólito

Cap. 10 – Classes Abstratas e Interfaces

Classes Abstratas

- Até este momento todas as classes que havíamos desenvolvido eram **classes concretas**, isto é, classes que podem originar objetos.

- **Classes concretas** são estruturas definidas e prontas para instanciarem objetos.

- Uma **classe abstrata** se comporta de forma diferente de uma **classe concreta**.

Classes Abstratas

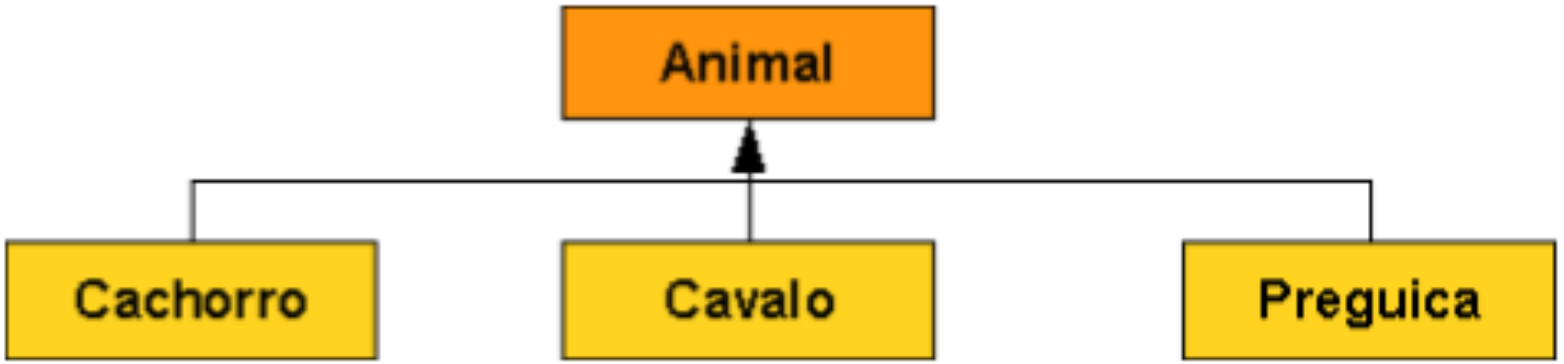
- Uma **classe abstrata** nunca pode ser instanciada de forma direta, seu maior propósito, a razão da sua existência, é ser estendida (herdada).

- Usadas quando não faz sentido termos instâncias de determinadas classes.

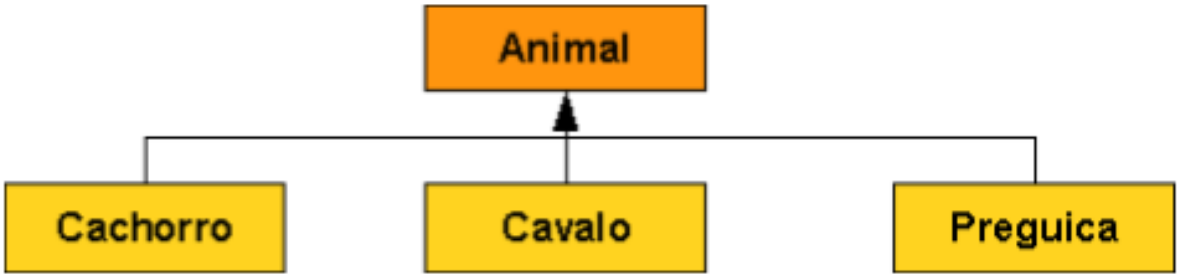
- Manter a consistência do programa.

Classes Abstratas

- Para que serve esta classe então ?
  - Imagine a seguinte situação, nós temos uma hierarquia de classe conforme abaixo:

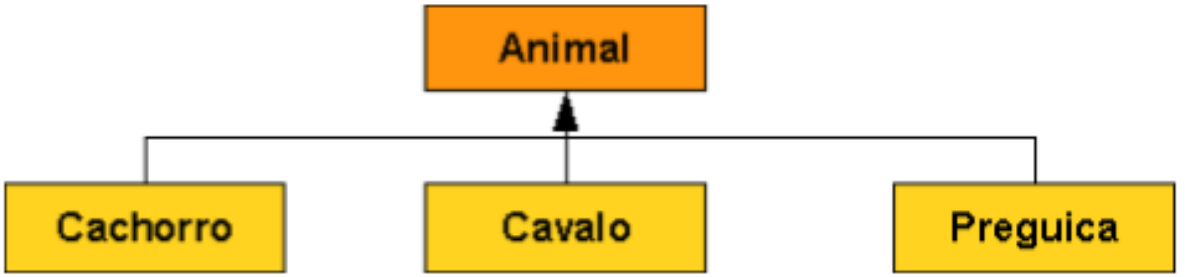


Classes Abstratas



- Todos os animais (cachorro, cavalo e preguiça) estendem (herdam) da classe Animal;
- No entanto, a classe Animal por si só não representa nenhum tipo de animal;
- Esta classe representa apenas um animal genérico e ela existe apenas para ser estendida(herdada) dando origem a um animal de fato.

Classes Abstratas



- Nesta situação é mais adequado que seja feita a mudança na classe Animal a fim de torná-la abstrata e inviabilizar a sua instanciação por qualquer parte do programa.
- Isto é realizado adicionando-se o modificador abstract.

Classes Abstratas

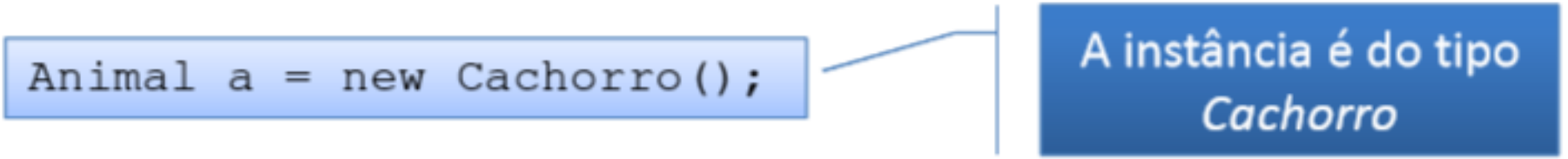
```
public abstract class Animal {
    ...
}
```

- Agora a classe Animal não pode ser instanciada indevidamente em nenhum trecho do nosso programa.

Classes Abstratas

- **Não** é permitida a existência de **objetos** da classe se ela for **abstrata:**

- É permitido criar **referências** à classe:



Classes Abstratas

- Desta forma nós temos certeza que durante a execução do programa nunca haverá um objeto da classe Animal e sim, sempre, de uma de suas subclasses.

- Uma classe abstrata pode possuir métodos concretos
  - Um método concreto é todo aquele possui corpo
  - Veja o exemplo a seguir:

Classes Abstratas

```
public abstract class Animal {
   private String nome;

  //getters e setters...

   public void falar() {
      System.out.println("Animal
  emitindo som");
   }
}
```

Classes Abstratas

- O método falar() deverá ser sobreescrito por cada uma das subclasses da classe Animal a fim de que forneçam um comportamento adequado pois, cada tipo de animal emite um som diferente dos demais.

- Porém, da forma como foi feito, esta condição não é garantida pois, podemos ter uma subclasse da classe Animal que não sobreescreve o método emitir som, no entanto existem formas de forçarmos esta condição.

Métodos Abstratos

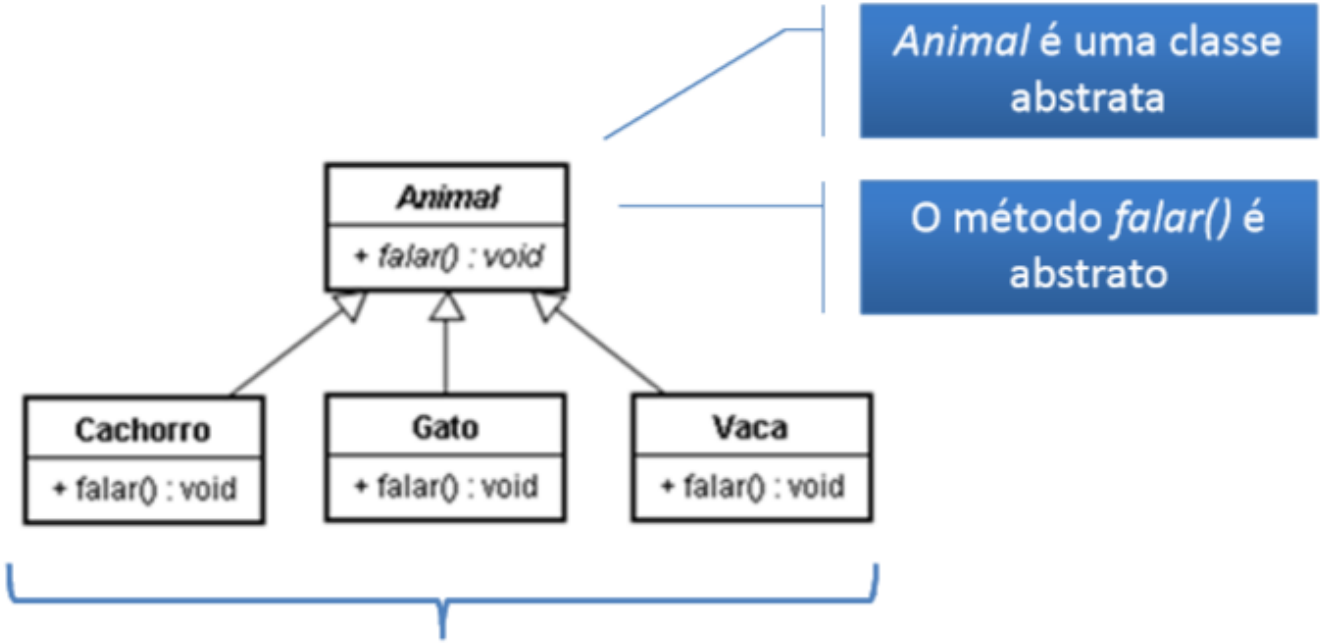
- Ao modificarmos o método com abstract impedimos que alguma das subclasses não implementem o método. Segue exemplo abaixo:

```
public abstract class Animal {
    public abstract void falar();
}
```

Métodos abstratos não são implementados

- Agora todas as subclasses da classe Animal irão obrigatoriamente ter que fornecer uma implementação para o método falar().

Métodos Abstratos



Todas as classes não-abstratas que herdam de uma classe abstrata são obrigadas a implementar os métodos abstratos

Os métodos chamados correspondem aos métodos implementados nas subclasses

Métodos Abstratos

- Uma **classe abstrata** pode conter apenas métodos abstratos, concretos ou ambos.

- No entanto a presença de um **método abstrato** torna, obrigatoriamente, a **classe abstrata** também, ou seja, métodos abstratos só podem existir em classes abstratas;

Classes Abstratas

- A extensão de uma classe Abstrata é feita da mesma forma que a extensão realizada por classes concretas:

```
public class Cachorro extends Animal {
   public void falar() {
      System.out.println("au! au!");
   }
}
```

Classes Abstratas

- Em Java não existe o conceito de herança múltipla, isto é, cada classe pode estender (extends) apenas uma superclasse de cada vez.
- Por exemplo, a seguinte situação **não** é permitida:

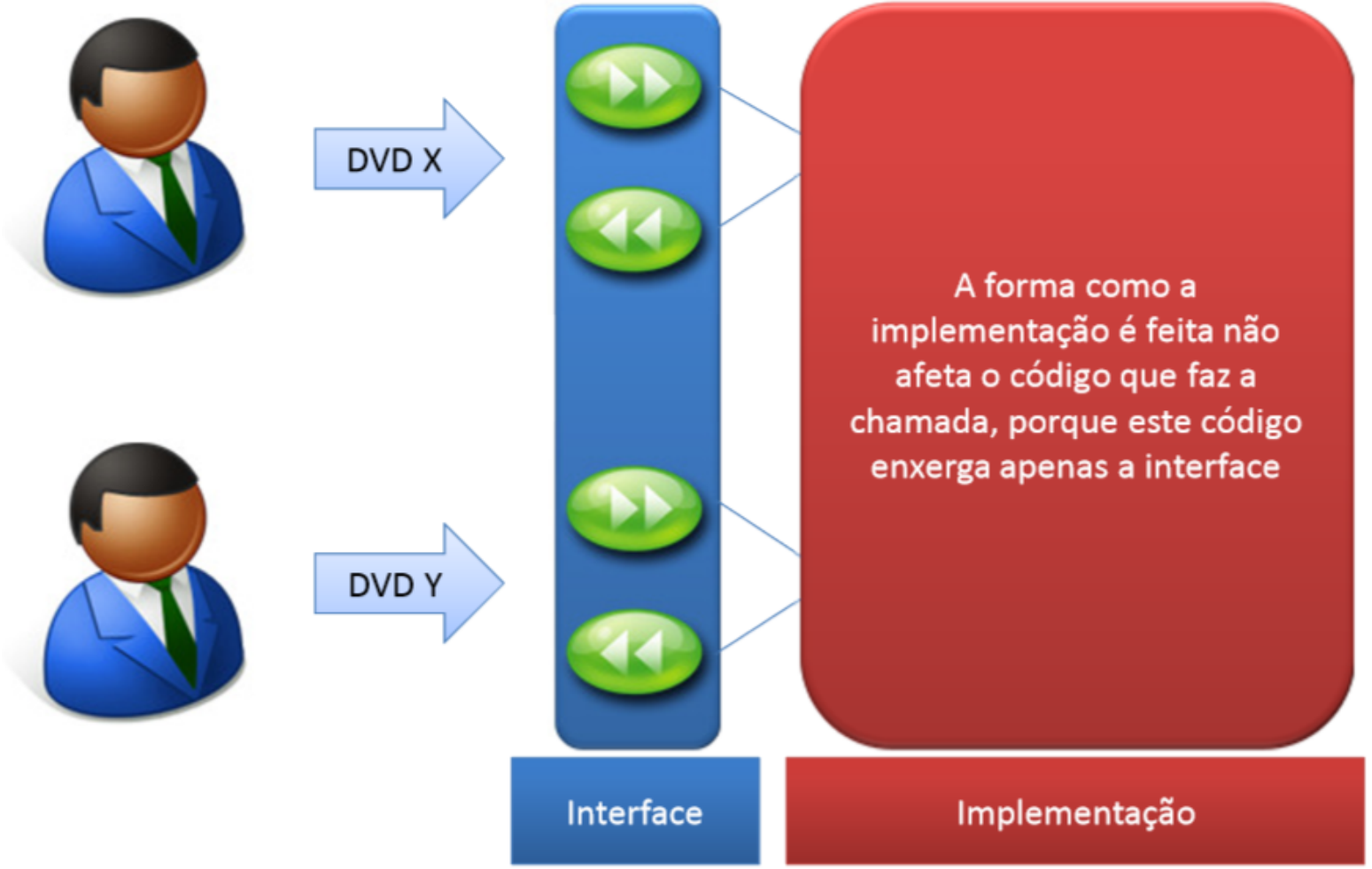
```
public class Cachorro extends Animal, Mamifero {
```

- No entanto podemos ter hierarquia de classes de qualquer tamanho o que viabilizaria a extensão de diversas classes nesta estrutura.

Classes e Métodos Abstratos

- Código Exemplo

Interfaces


DVD X
DVD Y
A forma como a implementação é feita não afeta o código que faz a chamada, porque este código enxerga apenas a interface
Interface
Implementação

Interfaces

- Pense em uma **interface** como sendo uma **classe totalmente abstrata**, isto é, que **não possui métodos concretos**.

- A interface define métodos, mas não os implementa.
  - Com exceção de métodos que usam os modificadores default e static

- Desta forma, quando uma classe implementa uma interface ela **obrigatoriamente deve implementar** o conjunto de métodos definidos pela interface.

Interfaces

- Quando uma **classe concreta** implementa uma **interface** ela está garantindo que possui os métodos especificados bem como a implementação destes, mesmo que o corpo do método seja vazio.

- Implementar uma interface permite a uma classe tornar-se **mais formal em relação ao comportamento que ela promete fornecer**.

Interfaces

- Interfaces formam um **contrato entre a classe e o mundo exterior**, e este contrato é garantido em tempo de compilação pelo compilador.

- Se a sua classe deseja implementar uma interface, **todos os métodos definidos pela interface devem aparecer no corpo antes da sua classe ser compilada**.

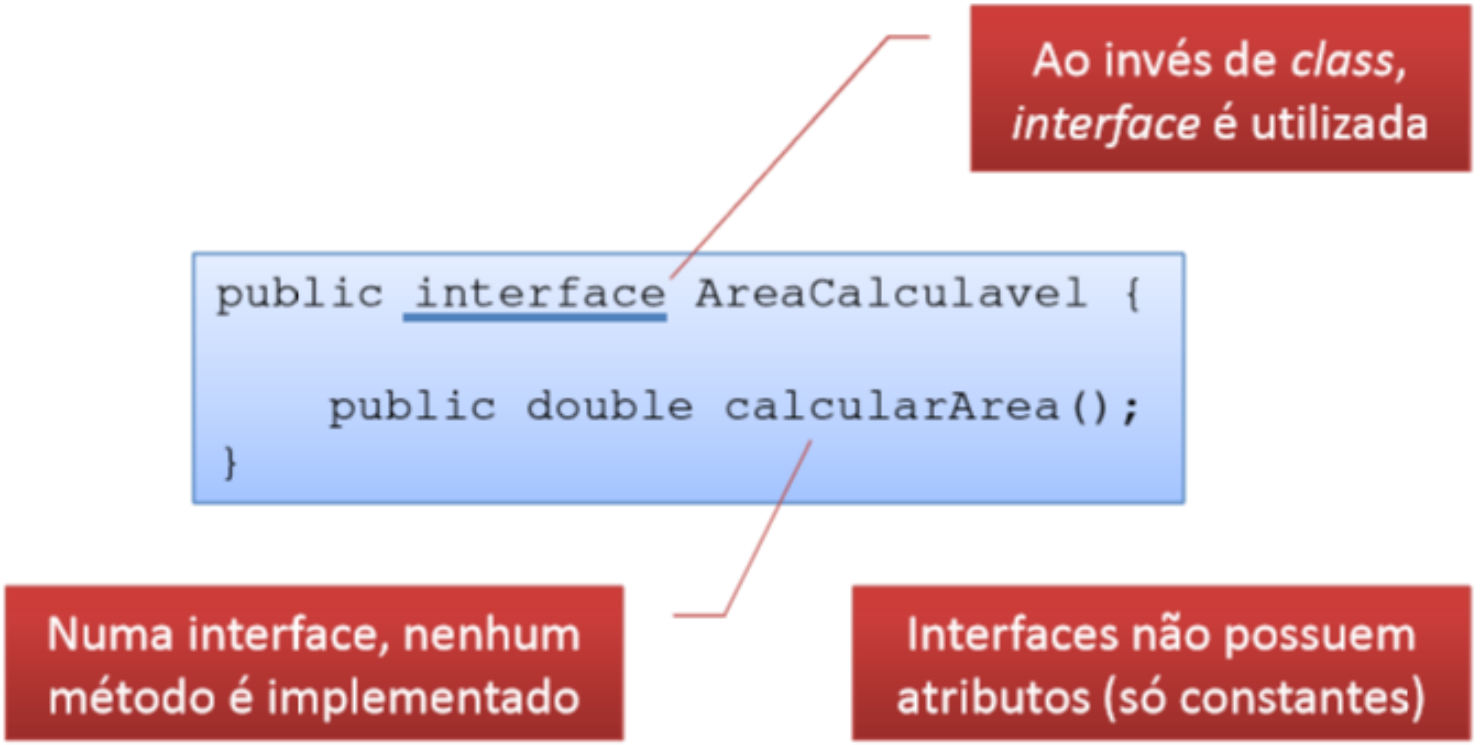
Interfaces

- Interfaces possibilitam mudanças de implementação muito mais facilmente, pois quem chama o método não conhece a sua implementação

- Alterações na interface não alteram as classes que a implementam

- O foco é no **que** o objeto faz, e não em **como** ele faz

Declarando Interfaces



Implementando Interfaces

```
public class Quadrado implements AreaCalculavel {

    private double lado;

    public double calcularArea() {
        return lado * lado;
    }
}
```

```
public class Circulo implements AreaCalculavel {

    private double raio;

    public double calcularArea() {
        return Math.PI * raio * raio;
    }
}
```

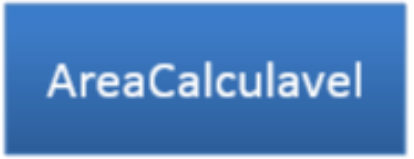
Os métodos da interface são implementados pela classe

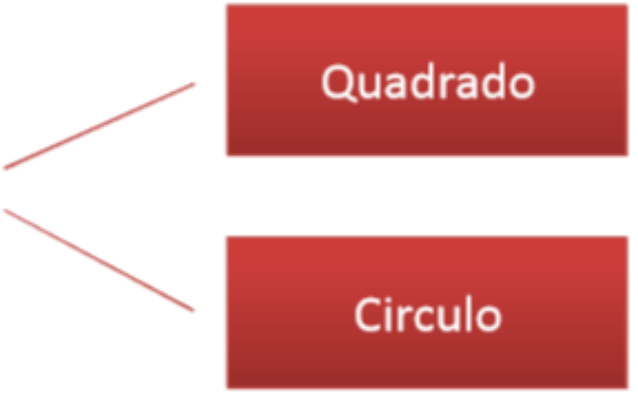
Exemplo de Interface

```
AreaCalculavel a = new Quadrado();
a.calcularArea();
```

```
AreaCalculavel a = new Circulo();
a.calcularArea();
```

Utilização de polimorfismo





Outras Considerações

- Interfaces podem estender outras interfaces;
- Classes podem estender outra classe, mas apenas podem implementar interfaces;
- Uma classe pode implementar uma ou mais interfaces;

Interfaces

- Código Exemplo

Métodos Default

- Uma interface pode definir métodos com o modificador default → suas implementações não precisam necessariamente reescrevê-lo

- Neste caso, o método é implementado diretamente na interface;

- Este recurso surgiu no Java 8 → utilizado para poder evoluir uma interface sem quebrar compatibilidade com as implementações anteriores

Definindo Métodos Default

```
public interface Calculator {

    double calculate();

    default double calculatePow(double x, int y) {
        return Math.pow(x, y);
    }
}
```

O método default é implementado na interface

```
public class MyCalculator implements Calculator {
    public double calculate() {
        //...
    }
}
```

```
MyCalculator my = new MyCalculator();
double x = my.calculatePow(10, 3);
```

Funciona como um método qualquer

Métodos Estáticos

- Interfaces também podem implementar métodos definidos com o modificador static;

- O método é acessível diretamente pela interface, sem precisar que ocorra a criação de objetos;

Definindo Métodos Estáticos

```
public interface Calculator {

    double calculate();

    static double calculatePow(double x, int y) {
        return Math.pow(x, y);
    }
}
```

O método estático é implementado na interface

```
Calculator.calculatePow(10, 3);
```

Método chamado diretamente na interface *Calculator*

Métodos Default e Static

- Código Exemplo

Classes Abstratas ou Interfaces ?

- Classe abstrata **não pode ser instanciada**, apenas **herdada**.
- Possui métodos **abstratos** e/ou métodos já **implementados**.
- Quem **herda** a classe abstrata deverá **implementar** os métodos abstratos, e consequentemente **herdará** a implementação dos métodos já implementados.
- É útil quando você precisa definir o corpo dos métodos e se ainda assim precisar, pode criar métodos abstratos para serem implementados pela classe que herda ela.

Classes Abstratas ou **Interfaces** ?

- Uma interface é um conjunto de métodos públicos, **sem implementação**, que deverão ser implementados pela classe que utiliza essa interface.
- Uma interface não pode ser instanciada, apenas implementada.
- Uma classe pode implementar diversas interfaces diferentes.
- Costuma-se dizer que uma interface é uma classe 100% abstrata.
- Todos os métodos são públicos e abstratos.
- É útil por exemplo, quando você quer que classes tenham métodos com a mesma assinatura mas implementações diferentes.
- Útil também quando existem determinados comportamentos que sejam similares entre hierarquias de classes diferentes.

Exercícios !

Bibliografia

- Site Java Starter - www.t2ti.com – acessado em 07/05/2015

- Curso Fundamentos de Java oferecido pela Softblue: www.softblue.com.br – acessado em 07/05/2015