

CLASSE ABSTRATA

- Ao subir na hierarquia de heranças, as classes se tornam mais genéricas e, provavelmente mais abstratas
- Em algum ponto, a classe ancestral se torna tão geral que acaba sendo vista mais como um modelo para outras classes do que uma classe com instâncias específicas que são usadas
- Uma classe abstrata não pode ser instanciada, ou seja, não há objetos que possam ser construídos diretamente de sua definição. Classes abstratas correspondem a especificações genéricas, que deverão ser concretizadas em classes derivadas (subclasses).
- Sintaxe:

```
abstract class NomeDaSuperclasse  
( // corpo da classe abstrata... )
```

MÉTODOS ABSTRATOS

- Cada classe filha terá um método diferente de bonificação
 - Queremos que cada pessoa que escreve a classe de um Funcionario diferente (subclasses de Funcionario) reescreva o método getBonificacao de acordo com as suas regras.
- Existe um recurso em Java que, em uma classe abstrata, podemos escrever que determinado método será **sempre** escrito pelas classes filhas. Isto é, um **método abstrato**.
- Ele indica que todas as classes filhas (concretas, isto é, que não forem abstratas) **devem reescrever esse método ou não compilarão**. É como se você herdasse a responsabilidade de ter aquele método

CLASSE E MÉTODOS ABSTRATOS

```
public abstract class ObjetoGeometrico {  
  
    abstract public double calculaArea();  
  
    public void mostraDados() {  
        System.out.println("Objeto Geométrico");  
    }  
  
}
```

EXEMPLO

Criar a classe abstrata impostos.

Esta classe terá informação do contribuinte, beneficiario e valor do imposto.

Teremos o método abstrato calcularImposto.

Serão criadas as heranças IPTU, ISS e Alvará.

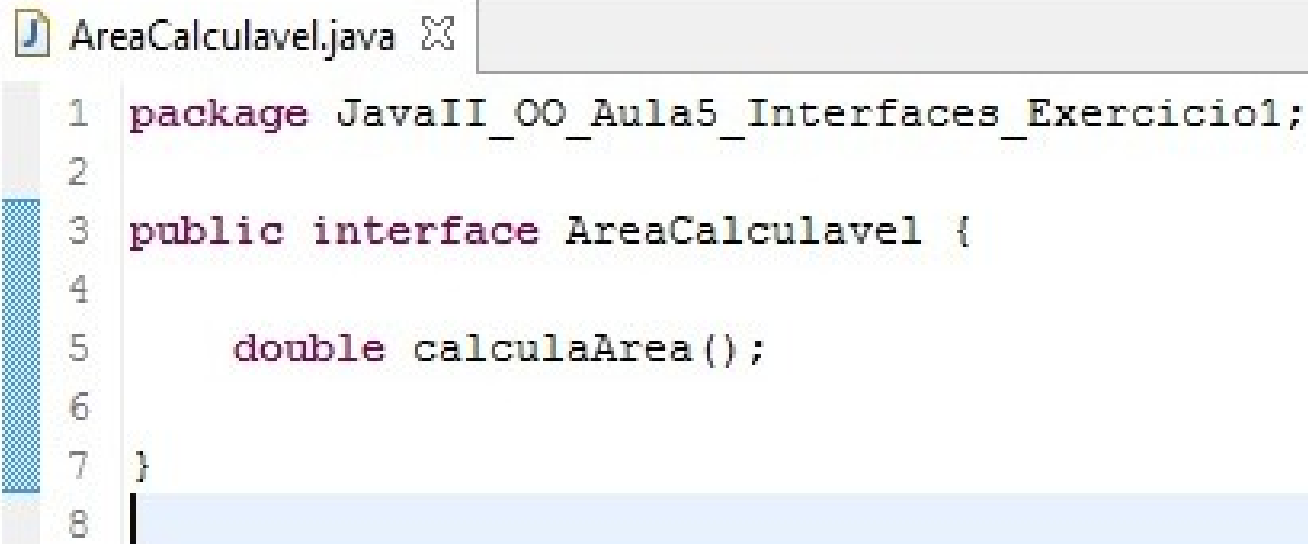
No IPTU o imposto será o valor do m2 multiplicado pela área do imóvel.

ISS será 2% do valor do faturamento da empresa.

Alvará será cobrado R\$ 200,00, concedendo R\$ 5,00 de desconto por funcionário CLT da empresa.

INTERFACE

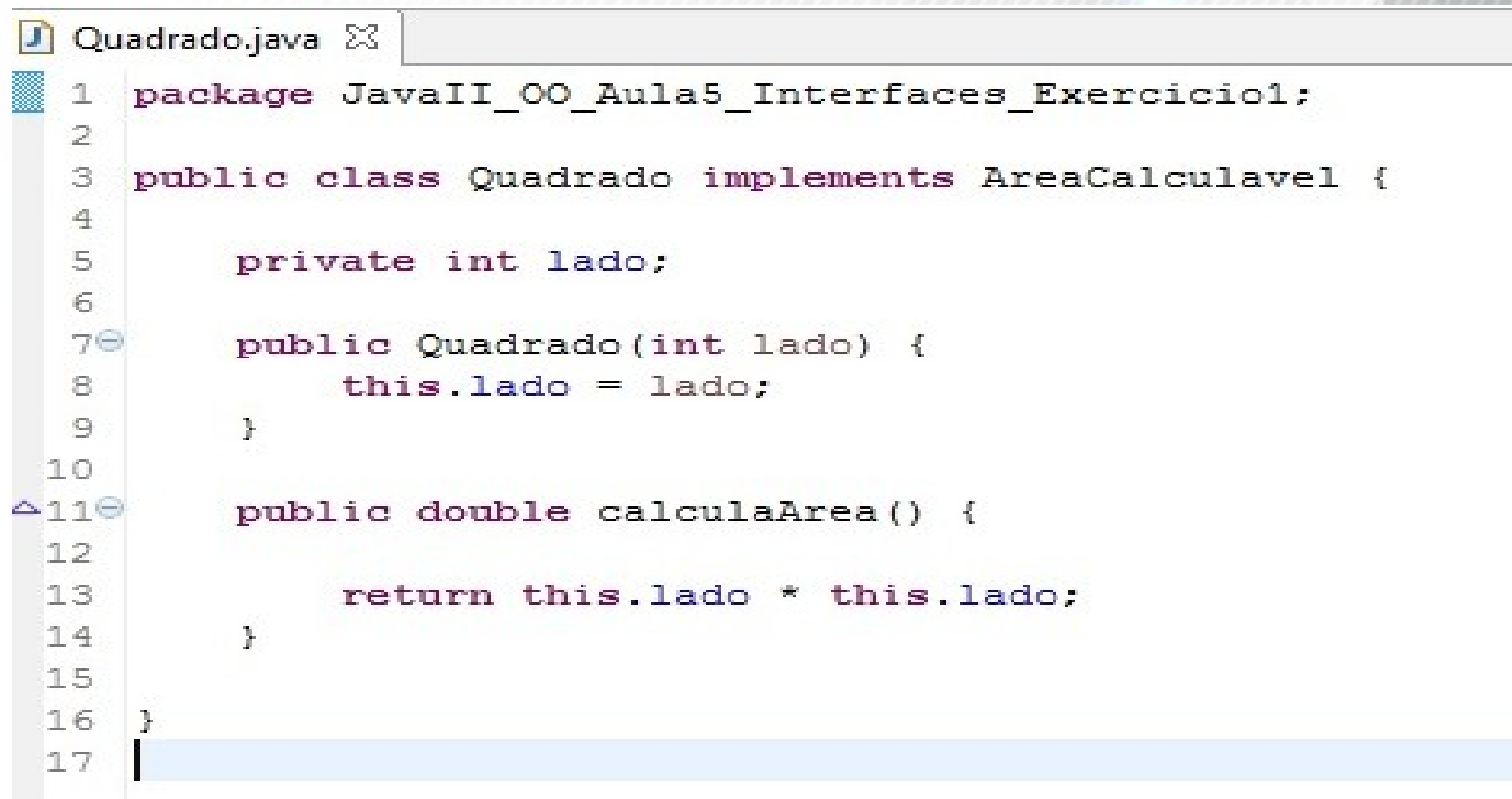
Interface não é Herança, mas permite aos programadores que determinados Objetos tenham o mesmo comportamento.



```
AreaCalculavel.java ✖
1 package JavaII_OO_Aula5_Interfaces_Exercicio1;
2
3 public interface AreaCalculavel {
4
5     double calculaArea();
6
7 }
8
```

INTERFACE

Desta forma podemos indicar que as classes estão implementados e prontas para utilização desta interface.



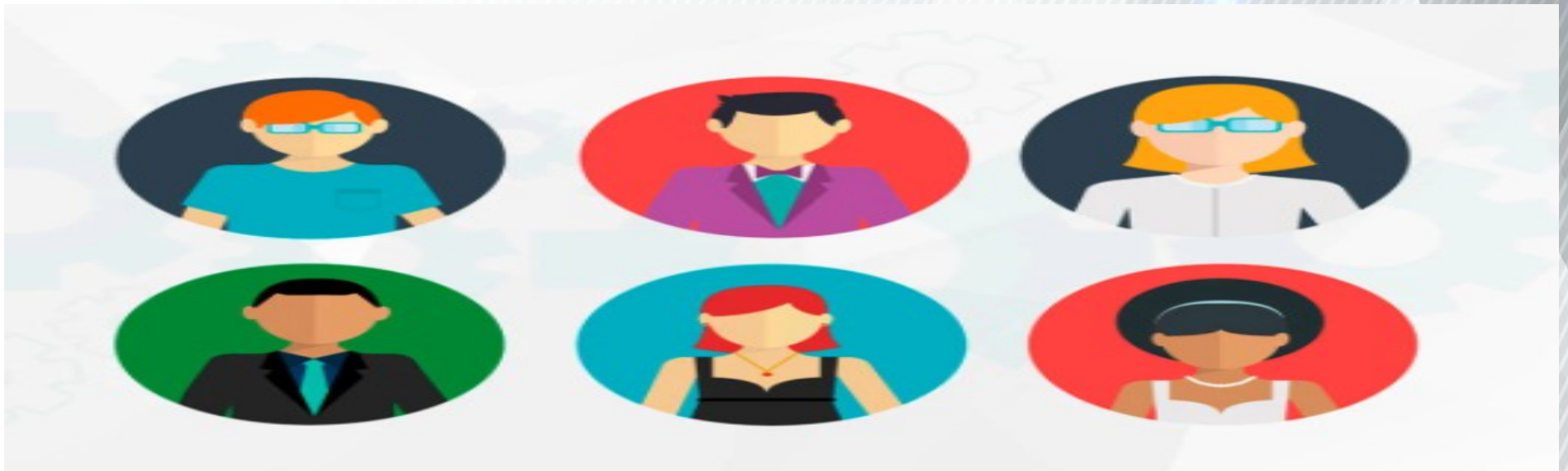
```
Quadrado.java
1 package JavaII_OO_Aula5_Interfaces_Exercicio1;
2
3 public class Quadrado implements AreaCalculavel {
4
5     private int lado;
6
7     public Quadrado(int lado) {
8         this.lado = lado;
9     }
10
11     public double calculaArea() {
12
13         return this.lado * this.lado;
14     }
15
16 }
17
```

EXEMPLO

Criar a Interface Usuarios com os métodos definirSenha, este método retorna um boolean e vai receber uma String como argumento.

Criar a classe Administrador implementando esta interface e o método definir senha vai exigir que sejam informados 8 ou mais caracteres.

Criar a classe DemaisUsuarios com a mesma interface, porém será necessário apenas 4 caracteres.



EXERCÍCIO

Criar a Interface Taxas nesta teremos o método calcularTaxa.

Criar a classe ContaCorrente, onde o método calcularTaxa vai cobrar 5% sobre o saldo.

Criar a classe ContaPoupanca, onde o método calcularTaxa vai cobrar 2% sobre o saldo.

