# Classes Abstratas

**Prof. Fábio Procópio**

**Prof. João Nascimento**

**Técnico Integrado em Informática – Programação Estruturada e Orientada a Objetos**

# Relembrando...

- Na aula passada, discutimos mais um pilar da orientação objetos: polimorfismo;

- Nesta aula, vamos falar sobre classes abstratas. Elas são utilizadas como modelos para implementação de classes derivadas. E aí? Ficou curios@? Então, vamos nessa?!

# Introdução

- São classes projetadas, especialmente, para servirem como modelo para classes que chamamos de **derivadas**;

- **Classes abstratas não permitem a instanciação de objetos**
  - Considerando Classe1 como sendo uma classe abstrata, então o comando a seguir geraria um erro: `c1 = Classe1()`

- As classes derivadas, via de regra, devem sobrescrever os métodos para realizar a implementação dos mesmos;

- As classes derivadas são conhecidas como classes concretas;

- Também podemos implementar polimorfismo usando classes abstratas.

# Métodos Abstratos – 1 de 2

- Os métodos abstratos são **incompletos**, ou seja, não possuem corpo;
- Se uma classe possuir, pelo menos, um **método abstrato**, esta **classe é abstrata**;
  - Somente classes abstratas podem possuir métodos abstratos
- Se uma classe possui um método abstrato, as **classes imediatamente descendentes deverão implementá-lo**;
- Em Python, o **módulo ABC** (*Abstract Base Classes*) fornece uma infraestrutura para a definição de classes base abstratas.

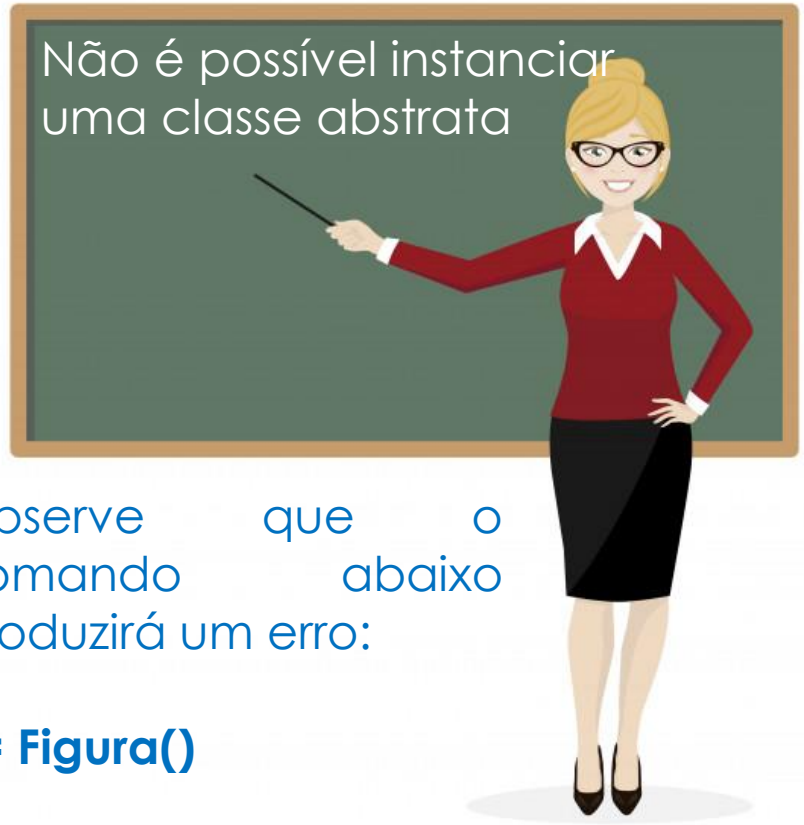
# Métodos Abstratos – 2 de 2

```
from abc import ABC, abstractmethod

class Figura(ABC):
    @abstractmethod
    def calcula_area(self):
        pass

    def mensagem(self):
        print("*** Exemplo de uma Classe Abstrata ***");
```

- Declara o método como sendo abstrato:
  Obriga as classes imediatamente descentes a implementá-lo
- Isso faz com que a classe seja considerada abstrata:
  Não permite que sejam instanciados objetos
- Não é um método abstrato:
  As subclasses de Figura não precisam implementá-lo




Observe que o comando abaixo produzirá um erro:

**f = Figura()**

# Exercício Resolvido 01 – 2 de 2

```
from Figura import *

class Quadrado(Figura):
  def __init__(self, lado):
     self.lado = lado

  def calcula_area(self):
     return self.lado ** 2
```

```
from Quadrado import *
from Triangulo import *

quad = Quadrado(2)
print(quad.calcula_area())

tri = Triangulo(2, 3)
print(tri.calcula_area())
```

```
from Figura import *

class Triangulo(Figura):
  def __init__(self, base, altura):
     self.base = base
     self.altura = altura

  def calcula_area(self):
     return (self.base * self.altura) / 2
```

# Canal no Youtube

Você pode complementar esta aula acessando o nosso canal no Youtube, Procópio na Rede, e assistindo a outras vídeo-aulas sobre este assunto. Veja:

- Classes Abstratas

# Referências

1) DEVMEDIA. **Conceitos – Classes Abstratas: Programação Orientada a Objetos – Parte 1.** Disponível em: http://www.devmedia.com.br/conceitos-classes-abstratas-programacao-orientada-a-objetos-parte-1/18812. Acessado em: 14 set. 2019.

2) PYTHON 3.7.8 Documentation. **abc - Abstract Base Classes**. Disponível em: https://docs.python.org/3.7/library/abc.ht. Acessado em: 14 set. 2019.

3) LARBACK. **Classes abstratas em Java**. Disponível em: http://www.larback.com.br/aula/119/Classes-abstratas-em-java. Acessado em: 14 set. 2019.

4) PROGRAMMINGKNOWLEDGE. **Python Tutorial for Begginers 35 – Python Abstract Classes.** Disponível em: https://www.youtube.com/watch?v=PDMe3wgAsWg. Acessado em: 17 set. 2019.