COM220
Computação Orientada a Objetos I

Aula 08: Polimorfismo

Polimorfismo

- Polimorfismo permite que referências de tipos de classes mais abstratas representem o comportamento das classes concretas que referenciam
  - Assim, é possível tratar vários tipos de maneira homogênea
- Polimorfismo significa "muitas formas"

Polimorfismo

- Exemplo
  - Suponha que eu tenha um conjunto de documentos de diferentes formatos (doc, pdf, ...)
  - Se eu tiver que implementar operações específicas para formatos específicos, vai dar um trabalhão
    - Visualizar pdf
    - Visualizar doc
    - Visualizar xyz
    - Imprimir pdf
    - Imprimir doc
    - Imprimir xyz

Polimorfismo

- Exemplo
  - Se for possível tratar os documentos genericamente fica muito mais fácil
    - Um único tipo de operação para todos os tipos de documentos
  - Isso é possível usando polimorfismo
    - Basta fazer uma superclasse abstrata *documento* e colocar nela métodos abastratos para cada uma das operações
    - Cada subclasse concreta faz a implementação desses métodos

Polimorfismo

```
from abc import ABC, abstractmethod

class Documento(ABC):
    def __init__(self, nome):
        self.__nome = nome

    def getNome(self):
        return self.__nome

    @abstractmethod
    def visualizar(self):
        pass
```

Polimorfismo

```
class Pdf(Documento):
    def visualizar(self):
        return 'Mostra no Adobe Acrobat'

class Word(Documento):
    def visualizar(self):
        return 'Mostra no Word'

documentos = [Pdf('PDF1'), Word('DOC1'), Pdf('PDF2')]
for documento in documentos:
    print('{}: {}'.format(documento.getNome(), documento.visualizar
()))
```

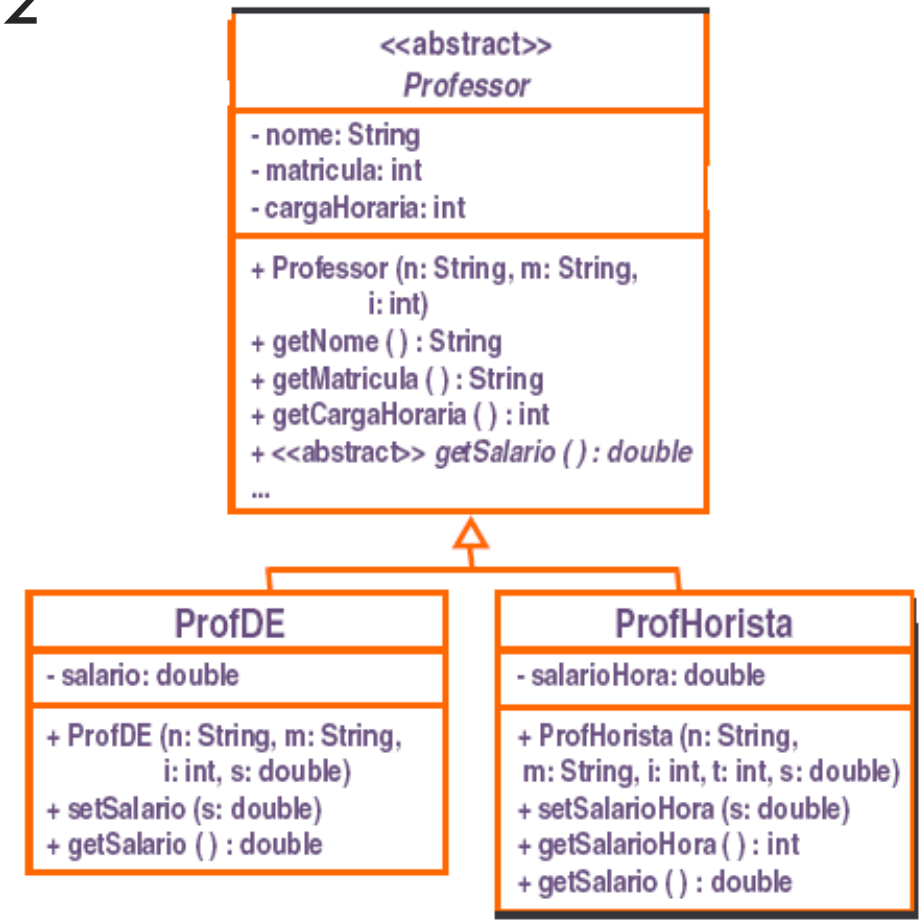
Polimorfismo

- Exemplo 2
  - A classe professor se subdivide em 2 tipos
    - ProfDE: Trabalha 40 hs e tem salário mensal fixo
    - ProfHorista: Trabalha um número de horas variável e ganha por hora trabalhada. Seu salário deve ser calculado fazendo a seguite conta:

      Nro horas x salário/hora

Polimorfismo

- Exemplo 2



Polimorfismo

```
from abc import ABC, abstractmethod

class Professor(ABC):
    def __init__(self, nome, matricula, cargaHoraria):
        self.__nome = nome
        self.__matricula = matricula
        self.__cargaHoraria = cargaHoraria

    def getNome(self):
        return self.__nome

    def getMatricula(self):
        return self.__matricula

    def getCargaHoraria(self):
        return self.__cargaHoraria

    @abstractmethod
    def getSalario(self):
        pass
```

Polimorfismo

```
class ProfDE(Professor):
    def __init__(self, nome, matricula, cargaHoraria, salario):
        super().__init__(nome, matricula, cargaHoraria)
        self.__salario = salario

    def setSalario(self, salario):
        self.__salario = salario

    def getSalario(self):
        return self.__salario
```

Polimorfismo

```
class ProfHorista(Professor):
    def __init__(self, nome, matricula, cargaHoraria, salarioHora):
        super().__init__(nome, matricula, cargaHoraria)
        self.__salarioHora = salarioHora

    def setSalarioHora(self, salarioHora):
        self.__salarioHora = salarioHora

    def getSalarioHora(self):
        return self.__salarioHora

    def getSalario(self):
        return self.__salarioHora * self.getCargaHoraria()
```

Polimorfismo

```
prof1 = ProfDE('Joao', 12345, 40, 5000)
prof2 = ProfHorista('Paulo', 54321, 30, 75)
prof3 = ProfHorista('Ana', 56789, 38, 95)
profs = [prof1, prof2, prof3]
for prof in profs:
    print ('Nome: {} - Salário: {}'.format(prof.getNome(),
                prof.getSalario()))
```

Exercício 1

- Implementar uma nova versão do exemplo 2 na qual deve-se calcular o salário líquido de um professor de acordo com as seguintes regras
  - Somente os professoresDE deverão recolher contribuição previdenciária, que corresponde a 11% do valor do salário
  - Todos os professores devem ter o desconto do imposto de renda, conforme tabela mostrada no próximo slide

Exercício 1

Até 1.903,98	isento
De 1.903,99 até 2.826,65	7,5%
De 2.826,66 até 3.751,05	15%
De 3.751,06 até 4.664,68	22,5%
Acima de 4.664,68	27,5%