PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO EM PYTHON - ARA0066

Semana Aula: 14

PARADIGMA ORIENTADO A OBJETO

Tema

6. PARADIGMAS: ESTRUTURADO, ORIENTADO A OBJETOS, FUNCIONAL E LÓGICO (CRÉDITO DIGITAL)

Objetivos

Aplicar e apresentar os conceitos de polimorfismo, herança, encapsulamento na orientação a objeto

Tópicos

6.1 PARADIGMA ORIENTADO A OBJETO

Procedimentos de Ensino-Aprendizagem

Nesta aula, estaremos conectados com o conteúdo digital. O aluno explora e estuda, previamente, o conteúdo digital disponível em seu ambiente virtual. Durante a aula, este conteúdo será discutido em sala em atividade mediada pelo professor, detalhada abaixo.

Situação-problema:
Os softwares muitas vezes possuem partes que são repetidos em várias situações. Vamos supor que você criou um módulo de impressão que é utilizado em diversos softwares. Como você poderia reutilizar esse módulo sem ter que copiar o código dele para programas diversos?

Metodologia:
Aula com um rápido debate sobre herança e polimorfismo discutindo as vantagens de utilização desses dois princípios da orientação a objetos, conectando com a situação problema apresentada. Em seguida o professor deve implementar a seguinte atividade junto com alunos para demonstrar o conceito de herança.
Criar a classe Pessoa

a. Atributos: Nome, Endereço
b. Métodos: Alterar Nome, Endereço; Retornar Nome, Endereço.
Em seguida deve criar a classe Aluno que irá herdar os atributos e métodos da classe Pessoa e a irá acrescentar
a. Atributo: Nota
b. Métodos: Alterar e Retornar Nota
Devem ser implementados na linguagem Python e na Linguagem Java.

Atividade verificadora de aprendizagem:
Considere, como subclasse da classe Pessoa a classe Fornecedor. Considere que cada instância da classe Fornecedor tem, para além dos atributos que caracterizam a classe Pessoa, os atributos valorCredito (correspondente ao crédito máximo atribuído ao fornecedor) e valorDivida (montante da dívida para com o fornecedor). Implemente na classe Fornecedor, para além dos usuais métodos seletores e modificadores, um método obterSaldo() que devolve a diferença entre os valores dos atributos valorCredito e valorDivida. Depois de implementada a classe Fornecedor, crie um programa de teste adequado que lhe permita verificar o funcionamento dos métodos implementados na classe Fornecedor e os herdados da classe Pessoa.
O professor deve apresentar a solução da atividade durante a aula.

Recursos Didáticos

Laboratório de Informática com Internet com navegador Web instalado, equipado com quadro branco, projetor multimídia, acervo bibliográfico no ambiente virtual.

Leitura Específica

[1] SEBESTA, Robert W. Conceitos de Linguagens de Programação. 11. edição. Porto Alegre: Bookman, 2018., Capítulo 12 (Suporte para programação orientada a objetos), páginas 536 a 540, Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582604694/

[2] Documentação do Python "Python e Programação Orientada a Objeto". Disponível em: https://wiki.python.org.br/ProgramacaoOrientadaObjetoPython

Aprenda +

[3] Vídeo "Programação orientada a objetos". Disponível em: https://www.youtube.com/watch?v=lbXsrHGhBAU (Ative a legenda e a tradução automática para o português)

Atividade Autônoma Aura:

Questão 1: (FGV - DPE-RJ, 2014) Considere o seguinte trecho de um programa escrito na linguagem Python.

```
class Carro(object):
def FaleComigo(self):
print "Sou um carro"

class Fusca (Carro):
def FaleComUmFusca(self):
print "Sou um Fusca"

x = Carro( )
y = Fusca( )

x.FaleComigo( )
y.FaleComigo( )
```

No primeiro bloco, o método*FaleComigo*é definido para a classe*Carro*, que simplesmente produz a mensagem "Sou um carro" ao ser invocado. Para a classe*Fusca*, definida no segundo bloco, foi definido o método*FaleComUmFusca*, que apenas produz a mensagem "Sou um Fusca". No terceiro bloco, os objetos x e y tornam-se instâncias das classes*Carro*e*Fusca*, respectivamente. No quarto bloco, o método*FaleComigo*é invocado para cada um dos dois objetos, x e y. Ao ser executado, esse programa produz duas linhas na sua tela de saída:

```
Sou um carro
Sou um carro
```

A mensagem produzida no comando y.*FaleComigo*deve-se ao mecanismo de

a) abstração

b) associação

c) interface

d) herança

e) polimorfismo

Questão 2: (COMPERVE - UFRN, 2016) (Adaptada) Sobre a implementação de conceitos de programação orientada a objetos na linguagem Python, é correto afirmar:

a) para construir uma hierarquia de classes, deve-se passar como parâmetro na definição da classe o construtor*self*.

b) ao aplicar o conceito de encapsulamento, não se deve declarar os métodos*get/set*como única estratégia de exposição dos atributos.

c) uma classe que implementa uma interface não é obrigada a implementar todos os métodos definidos na interface.

d) para definir um construtor em uma classe em Python, utiliza-se o método especial _init_, passando como parâmetro a referência da classe.

e) Pyhton não suporta herança múltipla