PROGRAMAÇÃO ORIENTADA A OBJETOS

Apresentação da Disciplina

Prof. Emanoel Barreiros

UNIVERSIDADE DE PERNAMBUCO

A Disciplina

- Componente curricular: Programação II

- Objetivos: Motivar, discutir, exercitar, e consolidar o uso de técnicas de programação que tenham um impacto considerável sobre a qualidade e produtividade no desenvolvimento de software, através do Paradigma de Programação Orientado a Objetos e do uso de uma Linguagem de Programação Orientada a Objetos

A Disciplina

- Ementa: Conceitos da programação orientada a objetos. Exploração de uma linguagem de programação orientada a objetos, através de suas construções e bibliotecas. Uso de ambiente integrado de desenvolvimento. Introdução à qualidade de software. Introdução a padrões e arquiteturas de softwares. Projeto: desenvolvimento de um sistema de pequeno porte.

- Referências:
  - CORNELL, Gary e HORSTMANN, Cay S. **Core Java V.1 – Fundamentos**. Editora Prentice Hall. 8ª Edição, 2009.
  - LEWIS, John and LOFTUS, William. ***Java Software Solutions: Foundations of Program Design***. Addison Wesley. 7th Edition, 2011.
  - DEITEL, Harvey e DEITEL, Paul. **Java – Como Programar**. Editora Prentice Hall. 8ª Edição, 2010.

Avaliação

- 1ª Unidade
  - Avaliação presencial escrita (70%)
  - Lista de Exercício/Projeto (30%)
- 2ª Unidade
  - Avaliação presencial escrita (60%)
  - Projeto (40%)

Vamos começar?

- O que é um objeto? Na vida real mesmo ☺

Objetos

- Objetos são coisas
  - Exemplos?

Objetos

- Objetos podem ter características em comum?
- Claro que podem, mas o que isso significa para nós?

Classes

- Existe o que chamamos de classe.
- Classes são como projetos para objetos, definem quais **características** os objetos devem ter.
- Classes também definem o **comportamento** que os objetos podem ter.

Comportamento de objetos

- Comportamentos são ações (funções) que podem ser executadas pelo objeto.

- Na POO as funções dos objetos são chamadas de **métodos**, e suas características são chamadas de **atributos.**

- **Vamos para um exemplo que ilustra os conceitos vistos até agora.**

Exemplo (analogia com um carro)

- Suponha que você deseja dirigir um carro que aumente a velocidade quando você pisa no acelerador e reduz sua velocidade quando você pisa no freio.

- Antes do carro poder existir, é necessário que alguém o projete.

- Neste projeto está definida a carroceria do carro, e as cores que ele pode ter, logo, cor é um _____.

Atributo!

Exemplo (cont.)

- Quando você pisa no acelerador, o carro ganha velocidade. Internamente, o pedal do acelerador envia uma mensagem para a injeção eletrônica que coloca mais combustível nos cilindros, logo, acelerar é um ____?
  - **Método**

- Muitas vezes, na OO dizemos que uma chamada de método é um **envio de mensagem**.

Classes e Objetos

- Classes são o projeto do carro, e objeto é o seu Camaro amarelo.

- Em outras palavras, classes definem uma estrutura que deve ser seguida, enquanto objetos (ou instâncias de uma classe) são uma "concretização" da classe.
  - Ex: Classe -> Marca, nome, motor, cor, quantidade de portas, tipo de pneu, etc.
  - Ex: Objeto -> O seu VW, Gol 1.0, vermelho, 4 portas, pneus pirelli, etc.

O Paradigma Orientado a Objetos

- No paradigma OO, programas são uma coleção de objetos, com atributos e métodos
- São capazes de interagir entre si, enviando e recebendo mensagens (chamando métodos de outros objetos, ou recebendo essas chamadas)
- Processam dados. Cada objeto pode ser visto como uma entidade independente, com responsabilidades e papéis distintos.

O Paradigma Orientado a Objetos

- *Features* presentes na maioria das linguagens orientadas a objetos
  - Chamada dinâmica de métodos: o método a ser executado é "decidido" em tempo de execução
  - Encapsulamento
  - Polimorfismo de subtipo
  - Herança
  - *Open recursion*: presença da variável *this* (ou *self*) para se referenciar ao objeto em execução no momento

História

- Conceito surgiu no fim dos anos 50 e início dos anos 60, com a criação da linguagem Lisp
- Grupo de inteligência artificial do MIT

- Primeira linguagem a seguir “oficialmente” o paradigma foi a Simula 67. Linguagem focada em simulação e tratamento de eventos
  - Introduziu o conceito de classes e suas instâncias (objetos), subclasses, métodos virtuais, entre outros
  - Garbage collection automático

História

- Desenvolvimento da linguagem Smalltalk pelo Xerox PARC (década de 70), liderado por Alan Kay
  - O termo *programação orientada a objetos* foi introduzido
  - Classes e objetos podem ser criadas dinamicamente, ao contrário do que Simula permitia

- Desenvolvimento da linguagem Eiffel (1985) por Bertrand Meyer
  - Linguagem puramente orientada a objetos
  - Focada em qualidade de software. Modela todo o ciclo de vida do software
  - Introduziu o conceito de design por contrato

História

- O paradigma OO se tornou “padrão” na década de 90, quando várias linguagens novas surgiram (e.g. FoxPro, C++, Delphi, Java)
- O surgimento e popularização de interfaces gráficas impulsionou a OO
- *Features* OO foram adicionadas a várias outras linguagens existentes na época (e.g. Ada, BASIC, Fortran, Pascal, COBOL), o que não foi bom…

História recente

Fonte: https://www.tiobe.com/tiobe-index/

Aug 2018	Aug 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.881%	+3.92%
2	2		C	14.966%	+8.49%
3	3		C++	7.471%	+1.92%
4	5	^	Python	6.992%	+3.30%
5	6	^	Visual Basic .NET	4.762%	+2.19%
6	4	v	C#	3.541%	-0.65%
7	7		PHP	2.925%	+0.63%
8	8		JavaScript	2.411%	+0.31%
9	-	^^	SQL	2.316%	+2.32%
10	14	^^	Assembly language	1.409%	-0.40%
11	11		Swift	1.384%	-0.44%
12	12		Delphi/Object Pascal	1.372%	-0.45%
13	17	^^	MATLAB	1.366%	-0.25%
14	18	^^	Objective-C	1.358%	-0.15%
15	10	vv	Ruby	1.182%	-0.78%
16	9	vv	Perl	1.175%	-0.82%
17	16	v	Go	0.996%	-0.65%
18	15	v	R	0.965%	-0.80%
19	13	vv	Visual Basic	0.922%	-0.89%
20	21	^	PL/SQL	0.702%	-0.51%