Programação Orientada a Objetos

INSTITUTO FEDERAL
MINAS GERAIS

Professor Leonardo Cabral - Larback

Programação Orientada a Objetos

O termo Programação Orientada a Objetos (**POO**) foi criado por Alan Kay (http://pt.wikipedia.org/wiki/Alan_Kay), um dos autores da linguagem de programação **Smalltalk**. Mesmo antes da criação do Smalltalk, algumas das ideias da **POO** já eram aplicadas, sendo a primeira linguagem a realmente utilizar estas ideias a linguagem **Simula 67**, criada por Ole Johan Dahl e Kristen Nygaard em 1967.

Alguns exemplos de linguagens modernas que adotaram essas ideias são Java, C#, C++, PHP, Ruby, Python e Lisp.

Programação Orientada a Objetos

A **POO** foi criada para tentar aproximar o mundo real do mundo virtual - O programador cria representações abstratas para os objetos do mundo real. Os objetos conversam uns com os outros através do **envio de mensagens**. O programador especifica quais serão as mensagens que cada objeto pode receber, e também qual a ação que aquele objeto deve realizar ao receber aquela mensagem em específico.

Programação Orientada a Objetos

Vantagens da POO:

- Os sistemas, em geral, possuem uma divisão de código um pouco mais lógica e melhor encapsulada do que a empregada nos sistemas não orientados a objetos. Isto torna a manutenção e extensão do código mais fácil e com menos riscos de inserção de bugs. Facilita também o reaproveitamento de código.
- É mais fácil gerenciar o desenvolvimento deste tipo de software quando temos uma equipe grande. Podemos fazer uma especificação UML antes de iniciar o desenvolvimento do software em si, e em seguida dividirmos o sistema em classes e pacotes, e cada membro da equipe pode ficar responsável por desenvolver uma parte do sistema.

POO – Classes e Objetos

Uma classe é uma abstração que define um tipo de objeto. Ela define o que objetos deste determinado tipo tem dentro deles (seus atributos) e quais ações esse tipo de objeto é capaz de realizar (métodos).

Note que uma Classe não tem vida, é só um conceito. Quando encontramos uma ocorrência específica da classe, estamos visualizando um objeto da mesma.

Por exemplo, a classe Professor contém as características de todos os professores. Mas a classe professor não pode lecionar ou fazer a chamada. Quem faz isso é o objeto (ou instância) *leonardoCabral* da classe Professor.

POO – Classes e Objetos

As classes podem possuir **atributos** e/ou **métodos**.

Os **atributos** (ou propriedades) são variáveis que estarão dentro de cada um dos objetos desta classe, e podem ser de qualquer tipo. Por exemplo, uma classe Cachorro poderá ter o atributo nome que será do tipo String. Assim, cada Objeto desta classe terá uma variável própria chamada nome, que poderá ter um valor qualquer (Rex, Totó, Ruga, Nóia etc.)

POO – Classes e Objetos

Métodos serão as ações que a Classe poderá realizar. Quando um objeto desta classe receber uma mensagem de algum outro objeto contendo o nome de um método, a ação correspondente a este método será executada. Por exemplo, caso um objeto da classe Dono envie uma mensagem para um objeto do tipo Cachorro falando "sente", o cachorro irá interpretar esta mensagem e consequentemente irá executar todas as instruções que foram especificadas na classe Cachorro dentro do método sente.

POO – Classes e Objetos

Vejamos um pequeno exemplo utilizando a linguagem Java. Criemos uma classe para representar Alunos:

```java
public class Aluno {
    // Atributos
    String nome;
    float n1;
    float n2;
    float nFinal;
    // Métodos
    public float calculaFinal(){
        return (n1+n2)/2;
    }
    public String resultado(){
        this.nFinal = this.calculaFinal();
        if (this.nFinal>=60)
            return "Aprovado";
        else
            return "Reprovado";
    }
}
```

POO – Classes e Objetos

Agora, podemos criar objetos da classe Aluno, atribuir valores para seus atributos e executar seus métodos.

```java
    public static void main(String[] args) {
        Aluno aluno01 = new Aluno();
        Aluno aluno02 = new Aluno();
        aluno01.nome = "Lestat de Licontour";
        aluno01.n1 = 30;
        aluno01.n2 = 50;

        aluno02.nome = "Pandora de Ìsis";
        aluno02.n1 = 70;
        aluno02.n2 = 60;
        System.out.println("O aluno "+aluno01.nome+
                " foi " + aluno01.resultado());
         System.out.println("O aluno "+aluno02.nome+
                " foi " + aluno02.resultado());
    }
```

POO – Classes e Objetos

Ao executarmos o código anterior, teríamos uma saída como:

```
run:
O aluno Lestat de Licontour foi Reprovado
O aluno Pandora de Ìsis foi Aprovado
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Como já foi dito, o paradigma de programação orientado a objetos pode ser implementado em diversas linguagens de programação. Tente refazer o exemplo anterior em outra linguagem, **PHP**, por exemplo.

POO – Classes e Objetos

Primeiro, criamos a classe. Observe o código PHP abaixo e compare-o com o código Java:

```php
class Aluno {
    // Atributos
    public $nome;
    public $n1;
    public $n2;
    public $nFinal;

    // Métodos
    public function calculaFinal() {
        $this->nFinal = ($this->n1+$this->n2)/2;
    }
    public function resultado() {
        if ($this->nFinal>=60)
            return "Aprovado";
        else
            return "Reprovado";
    }
}
```

POO – Classes e Objetos

Agora vamos criar os objetos e manipulá-los:

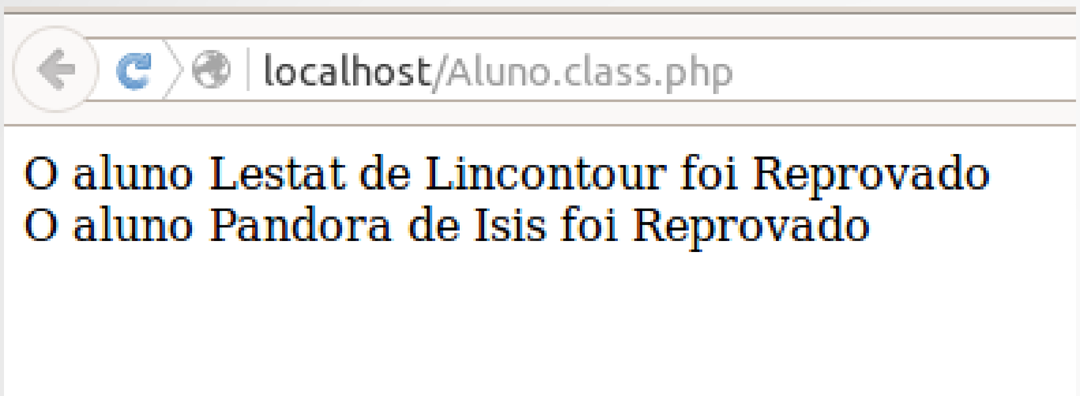
```
$aluno01 = new Aluno();
$aluno02 = new Aluno();
$aluno01->nome = "Lestat de Lincontour";
$aluno01->n1 = 30;
$aluno01->n2 = 50;

$aluno02->nome = "Pandora de Isis";
$aluno02->n1 = 70;
$aluno02->n2 = 60;

echo "O aluno " .$aluno01->nome. " foi " .$aluno01->resultado();
echo "<br>";
echo "O aluno " .$aluno02->nome. " foi " .$aluno02->resultado();
```

POO – Classes e Objetos

Ao executarmos o código, obteremos a saída:



Experimente um pouco mais, tente implementar o exemplo em outras linguagens e note as semelhanças e diferenças. Qual implementação da POO mais te agrada?

Bons estudos.

Bibliografia

DALL'OGLIO, P.; *PHP Programando com Orientação a Objetos. 2ª Edição*. São Paulo: NOVATEC, 2009.

DAVID, Marcio Frayze. *Programação Orientada a Objetos: Uma Introdução*. 2011. em: < http://www.hardware.com.br/artigos/programacao-orientada-objetos/ > Acesso em 1 fevereiro 2016.

Maia, Renato Dourado, and Edurado MORELLI. *Programação Orientada a Objetos*. (2007).