Gente,
Saúde e
Bem-estar.

RaiaDrogasil S.A.

Linguagem Java
-
Orientação a Objetos

Orientação a Objetos

Orientação a Objetos
Paradigmas de Programação

Um paradigma é um estilo de programação, um modelo, uma metodologia. Não se trata de uma linguagem, mas a forma como você soluciona problemas usando uma determinada linguagem de programação.

Existem muitas linguagens de programação conhecidas, mas todas elas precisam seguir algumas regras quando implementadas. E essas regras são os paradigmas.

Desta forma, quando uma nova linguagem de programação é desenvolvida, conforme suas peculiaridades, ela tende a se enquadrar em um paradigma ou até mesmo multiparadigma, como o JavaScript, por exemplo.

Principais Paradigmas de Programação

- Não Estruturado
- Procedural Estruturado
- Funcional
- Orientado a Objetos

Orientação a Objetos - Paradigmas

Não Estruturado

Nas linguagens de programação, o comando que indica se o Paradigma Não-Estruturado é suportado é o goto (“go to” no Inglês ou “vá para” no Português). Com esse comando, você pode desviar o fluxo de execução para qualquer outra parte.

No programa fictício abaixo, é mostrada a mesma situação onde se usa o go to:

```
p1();
if (c1) {
   p2();
} eles {
   p3:
   p3();
}
p4();
if (c2){
  p5();
} eles {
  goto p3;
}
```

Orientação a Objetos - Paradigmas

Não Estruturado

Apesar das críticas (que, no meu ponto de vista, estão corretas), não há nada de errado com esse paradigma. A questão toda é que, quando os algoritmos começam a ficar muito grandes, seguir o fluxo de comandos passa a ficar muito complicado. E, com isso, qualquer alteração ou manutenção passa a ficar inviável. É aí que entram algumas restrições que colocamos na possibilidade de direcionar o fluxo para qualquer lugar.

Orientação a Objetos - Paradigmas

Procedural
A programação procedural é excelente para programação de uso geral e consiste numa lista de instruções para informar ao computador o que fazer passo a passo.
A maioria das linguagens de programação ensinadas na faculdade são procedurais, exemplos:

- C
- C++
- Java
- Pascal

Quando é recomendado usar programação procedural:

- Quando existir uma operação complexa que inclui dependências entre operações e quando há necessidade de visibilidade clara dos diferentes estados do aplicativo.
- O programa é muito único e poucos elementos foram compartilhados.
- O programa é estático e não se espera que mude muito ao longo do tempo.
- Espera-se que nenhum ou apenas alguns recursos sejam adicionados ao projeto ao longo do tempo.

Orientação a Objetos - Paradigmas

Funcional
O paradigma de programação funcional tem suas raízes na matemática e é independente da linguagem.
A base desse paradigma é a execução de uma série de funções matemáticas. Você compõe seu programa de funções curtas. Todo o código está dentro de uma função.
Todas as variáveis têm escopo definido para a função.
No paradigma de programação funcional, as funções não modificam nenhum valor fora do escopo dessa função e as próprias funções não são afetadas por nenhum valor fora do escopo.
Linguagens que usam este paradigma:

- Haskell
- Scala
- Racket
- JavaScript

Ideal para usar quando:

- Tem matemática envolvida diretamente na programação.

Orientação a Objetos - Paradigmas

Orientada a Objetos
A programação orientada ao objeto (OOP) é o paradigma de programação mais popular devido aos seus benefícios, como a modularidade do código e a capacidade de associar diretamente problemas reais em termos de código.
Neste caso, o programa é escrito como uma coleção de classes e objetos para uma boa comunicação. A entidade menor e básica é objeto e todo tipo de cálculo é realizado apenas nos objetos.
É o paradigma mais popular e requisitado pelas empresas e as principais linguagens que o implementam são:

- PHP
- Java
- Ruby
- C#
- Python

Vale a pena utilizá-la quando:

- Vários programadores atuam juntos e não precisam entender tudo sobre cada componente.
- Existe muito código a ser compartilhado e reutilizado.
- São previstas muitas mudanças no projeto.

Orientação a Objetos - Pilares

Os pilares da orientação a objetos são os princípios sobre os quais a programação Orientada a Objetos foi fundamentada.

São quatro os pilares da Orientação a Objetos:

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Orientação a Objetos – Pilares
Abstração

Abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

Por exemplo, imaginamos a abstração referente a classe Animais. Há várias entidades na classe Animais como Anfíbios, Répteis e Mamíferos que são também sub-classes da classe Animais, onde há objetos que contêm cada sub-classe como Ser-humano, Jacaré e outros.

Orientação a Objetos – Pilares
Encapsulamento

O encapsulamento é uma técnica que adiciona segurança à aplicação em uma programação orientada a objetos, pois esconde as propriedades, criando uma espécie de caixa preta.

Muitas das linguagens orientadas a objetos implementam o encapsulamento baseado em propriedades privadas, por métodos chamados getters e setters, responsáveis por retornar e setar o valor da propriedade, respectivamente. Assim, se evita o acesso direto à propriedade do objeto, adicionando outra camada de segurança à aplicação.

Orientação a Objetos – Pilares
Encapsulamento

Modificadores de Acesso

public

- O modificador de acesso public é o menos restritivo de todos. Ele permite que qualquer outra parte da sua aplicação tenha acesso ao componente marcado como public.

protected

- Os membros das classes marcados com o modificador de acesso protected serão acessíveis por classes e interfaces dentro do mesmo pacote e por classes derivadas mesmo que estejam em pacotes diferentes.

default

- O modificador de acesso padrão, também conhecido como acessibilidade de pacote, é o modificador atribuído aos membros da classe que não foram marcados explicitamente com um outro modificador de acesso. Membros com acessibilidade de pacote só podem ser acessados por outras classes ou interfaces definidas dentro do mesmo pacote.

private

- O modificador de acesso private é o mais restritivo modificador de acesso. Todo membro de uma classe definido com o modificador private só é acessível para a própria classe. Não importa a localização dentro de pacotes ou se a classe foi herdada ou não, um membro private só é acessível dentro da mesma classe em que ele foi declarado.

Orientação a Objetos – Pilares
Herança

Na programação orientada por dados, o reuso de código é uma de suas vantagens de destaque e ela se dá por herança. Essa característica otimiza a produção da aplicação em tempo e linhas de código.

Para fazer uma analogia próxima à realidade não virtual, em uma família, por exemplo, a criança herda diretamente do pai e indiretamente do avô e do bisavô. Em programação, a lógica é similar. Assim, os objetos filhos herdam as características e ações de seus ancestrais”.

Orientação a Objetos
Composição

Em ciência da computação, composição de objetos (não confundir com composição de funções) é uma maneira de se combinar objetos simples ou tipos de dados em objetos mais complexos. Composições são blocos de construção críticos de muitas estruturas de dados básicas, incluindo lista ligada(LinkedLists) e árvore binária, bem como o objeto utilizado em programação orientada a objetos.

Objetos compostos são frequentemente referidos como tendo um relacionamento "tem um". Um exemplo de composição do mundo real pode ser visto em um automóvel: os objetos roda, volante, banco, transmissão e motor podem não ter utilidade funcionando isoladamente, mas um objeto chamado automóvel contendo todos estes objetos teria uma função muito útil, maior que a soma de todas as suas partes.

Quando, em uma linguagem de programação, os objetos são tipados, os tipos geralmente podem ser divididos em tipos compostos e não-compostos e a composição pode ser considerada uma relação entre os tipos: um objeto de um tipo composto (e.g. carro) "tem um" objeto de um tipo mais simples (e.g. roda).

Orientação a Objetos
Enum

São tipos de campos que consistem em um conjunto fixo de constantes (static final), sendo como uma lista de valores pré-definidos. Na linguagem de programação Java, pode ser definido um tipo de enumeração usando a palavra chave enum. Todos os tipos enums implicitamente estendem a classe java.

Orientação a Objetos – Pilares
Polimorfismo

Na natureza, existem animais que são capazes de alterar sua forma conforme a necessidade. Na orientação a objetos a ideia é a mesma.

O polimorfismo permite herdar um método de classe pai e atribuir uma nova implementação para o método pré-definido.

Ou definir um método que possua uma entrada padrão funcionando para vários tipos de objetos que tenham a mesma classe Pai no seu processo de construção.

Orientação a Objetos – Pilares
Polimorfismo

Estático
Ocorre quando temos sobrecarga de métodos.
O comportamento é alterado de forma explícita pelo desenvolvedor.
Por exemplo:

```
void calcularSalario(float horasTrabalhadas, float salarioBase){
        return horasTrabalhadas * salarioBase;
}

void calcularSalario(float horasTrabalhadas, float salarioBase, float bonus){
        return(horasTrabalhadas * horasTrabalhadas) *(1 + bonus);
 }
```

Orientação a Objetos – Pilares
Polimorfismo

Dinâmico
Ocorre quando temos herança entre classes.
Um exemplo de como isso ocorre:

```
List lista = new ArrayList();
```

Aqui definimos um objeto ArrayList que ao mesmo tempo é do tipo List.
O polimorfismo ocorre porque podemos utilizar esta mesma variável lista para fazer a seguinte atribuição:

```
lista = new LinkedList();
```

Um outro exemplo:
Temos um método estacionar abaixo:

```
void processarLista(LinkedList lista){
}
```

Nesse caso não temos polimorfismo pelo fato do método exigir um tipo específico de carro. Mas podemos torná-lo polimórfico da seguinte maneira:

```
void estacionar(List lista) {
}
```