Paradigma de Orientação a Objeto

Resumo

Por: Lucivaldo Araujo

Linguagem de Programação

- De uma forma bem resumida, uma linguagem de programação É a arte que usamos para orientar uma máquina a executar determinada tarefa. É o "idioma" que devemos falar quando nos comunicamos com uma máquina de forma que essa máquina entenda e execute funções a ela determinadas.

  - Já o paradigma de programação é a forma que

Paradigmas de Desenvolvimento

- Para programar (desenvolver um software), Temos que estipular uma linha de raciocinio que conduza nossas ações a uma padronização do programa. O Paradigma de Programação é a forma de estrutura que o programador irá usar para programar e executar o software de forma padronizada.
- Escolher entre uma Programação Estruturada ou Programação Orientada a Objetos, é algo que vai depender da desenvoltura, da formação e envolvimento de cada programador com o tipo de paradigma escolhido.
- O certo é que tanto com um quanto com outro padrão de Desenvolvimento escolhido, o problema poderá ser resolvido a contento, cabendo aí apenas a escolha do melhor meio para chegar a solução.
- Para basearmos o tema que por hora decidimos explanar, o Paradigma de Orientação a Objetos, vamos resumir brevemente os dois paradigmas mais utilizados na atualidade, o da Programação estruturada e o da Programação orientada a objeto.

Programação Estruturada

- A programação estruturada, surgiu da necessidade de levar ao programador um maior controle sobre o fluxo do programa. É um tipo de programação onde a ideia é dividir um programa em blocos conhecidas como procedimentos ou funções. Os procedimentos ou funções se interligam no programa através do uso dos seguintes mecanismos ou estruturas: sequência, decisão e iteração.

  - Sequência: É onde uma tarefa ou rotina é executada em um programa.

  - Decisão: É aqui que ocorre a seleção de determinado fluxo para a execução de uma rotina dentro do programa.

  - Iteração: que a partir do teste lógico algum trecho do código , bloco de comandos, pode ser repetido finitas vezes.

Característica do paradigma estruturado

- **Vantagens**
  1- Um melhor controle sobre o desenvolvimento do código.

  2- Ser fácil de entender.

  3- Permite tratar um grande problema. Dividindo-o em vários problemas menores.

- **Desvantagens**
  1- falta de foco do que deve ser feito.

  2- Tendência a gerar códigos confusos.

  3- GAP Semântico Maior (Distância entre o problema do mundo real e o modelo abstrato construído)

- **Linguagens que usam este tipo de paradigma:**

  Basic, C, Pascal, Cobol, PHP e Perl .

Paradigma de Orientação a Objeto

- O Paradigma de Orientação a Objeto busca meios de diminuir o GAP- Semântico, tornando a abstração mais limpa, mais fácil de se entender e modificar.
- Trata todos os componentes de um programa como objetos, cada um com sua identidade, comportamento e estado.
- Faz a modelação dos problemas do mundo real como objetos, que interagem e se associam entre si.
- Os objetos são instâncias de classes, que por sua vez, representam um conjunto de objetos com características em comum e estão alocadas em pacotes.

- **Linguagens que usam este tipo de paradigma:**

C++, C♯, VB.NET, Java, Object Pascal, Objective-C, Python, SuperCollider, Ruby e Smalltalk

Conceitos

- **PACOTE** - É um espaço organizador em um sistema, que une um conjunto de classes, arquivos, diagramas ou até mesmo outros pacotes.

- **Classe -** Representa um conjunto de objetos com características em comum. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.
  - ✓ **Métodos**: Definem as habilidades dos objetos.
  - ✓ **Atributos:** são características de um objeto.

- Subclasse é uma nova classe que herda características de sua(s) classe(s) ancestral(is) (Classe filha herda características da classe mãe).

Conceitos(continuação)

- **OBJETO** - Um objeto recebe informações de suas características através de seus atributos e armazena seus estados, modificando-os de acordo com as mensagens que recebe.
- ✓ Os Objetos são também chamados de Instâncias de uma Classe.
- ✓ Cabe realçar que os objetos encontram-se organizados em classes e essas determinam dados comuns a esses objetos mas não impedindo que cada objeto tenha seus atributos particulares.

❖ Exemplo: Cada objeto pertencente a classe ao lado, passará a ter Os atributos: cor, marca e velocidade, mas poderá ter também: motor, porta combustível, etc.

CARRO	←	NOME DA CLASSE
COR	←	ATRIBUTOS
MARCA		
VELOCIDADE		
ACELERAR	←	MÉTODOS
FREAR		
PARAR		

Conceitos(continuação)

- Vários outros conceitos fazem parte do paradigma de programação orientada a objetos, termos que nos remetem a conceitos de interação, dependências, associação, proteção e etc.

Exemplos:

- **Mensagem:** Utilizada para invocar um dos métodos de certo objeto. pode ainda, ser direcionada a uma classe, (invocando um método estático).
- **Herança:** (ou generalização) Através do conceito de herança, uma subclasse, poderá agregar valores advindos dos atributos e métodos da superclasse.
- **Associação:** É a utilização por um objeto de recursos pertencentes a outro. Por exemplo: o caso de um objeto "homem" utilizando os recursos de teclas de um objeto "telefone". As teclas usadas na digitação são parte do objeto telefone.
- **Encapsulamento:** É um conceito utilizado na proteção dos atributos de um objeto, impedindo o acesso direto. Os atributos poderão ser acessados e/ou modificados através dos métodos como os GET e SET (na linguagem de programação JAVA).

Conceitos(continuação)

- **Polimorfismo:** São propriedades relativas a linguagem. Temos 4 tipos de polimorfismo divididos em: Universal (Inclusão e Paramétrico) e Ad-Hoc (Sobrecarga e Coerção).
- ✓ Nem todas as linguagem "O.O" tem todos os tipos de polimorfismo implementados.
- Universal:
  - Inclusão: É o tipo de polimorfismo mais básico que existe, consiste em um ponteiro para a classe mãe poder apontar para um objeto de uma classe filha.
  - Paramétrico: Se restringe ao uso de *templates* (C++, por exemplo) e *generics* (Java/C#)
- Ad-Hoc:
  - Sobrecarga: Duas funções/métodos com o mesmo nome mas assinaturas diferentes
  - Coerção: A linguagem que faz as conversões implicitamente (como por exemplo atribuir um int a um float em C++, isto é aceito mesmo sendo tipos diferentes pois a conversão é feita implicitamente)

Conceitos(continuação)

- **Abstração:** É o conceito de focar nas características essenciais do contexto, deixando características menos importantes em um segundo plano.

Se tomarmos por base que determinados objetos possuem seus próprios atributos e todos pertencem a uma mesma classe, a abstração seria a habilidade de concentrar nas características relativas a classe, generalista a todos os objetos dela e não em se preocupar com as características individuais de cada objet**o.**

- **Interface:** é um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface.

Referências

- http://www.devmedia.com.br/conceitos-dos-generics-e-seus-tipos-delegates-events-e-generics-estrutura-da-linguagem-parte-3/20063
- http://www.dicionarioinformal.com.br
- http://www.inf.ufes.br/~vitorsouza
- http://pt.wikipedia.org/wiki/