Estratégia Concursos

Aula 00

Passo Estratégico de Conhecimentos de TI p/ TCM-SP (Auxiliar Técnico - TI) - Pós-Edital

Autor:
Thiago Rodrigues Cavalcanti

10 de Março de 2020

CONSTRUÇÃO DE ALGORITMOS (TIPOS DE DADOS, VARIÁVEIS E CONSTANTES, COMANDOS DE ATRIBUIÇÃO, AVALIAÇÃO DE EXPRESSÕES, COMANDOS DE ENTRADA E SAÍDA, FUNÇÕES PRÉ-DEFINIDAS, ESTRUTURAS DE CONTROLE, PASSAGEM DE PARÂMETROS, RECURSIVIDADE, PROGRAMAÇÃO ESTRUTURADA)

Sumário

APRESENTAÇÃO

Olá Senhoras e Senhores,

Eu me chamo Thiago Cavalcanti. Sou funcionário do Banco Central do Brasil, passei no concurso em 2010 para Analista de Tecnologia da Informação (TI). Atualmente estou de licença, cursando doutorado em economia na UnB. Também trabalho como professor de TI no Estratégia e sou o analista do Passo Estratégico de Informática.

Tenho graduação em Ciência da Computação pela UFPE e mestrado em Engenharia de Software. Já fui aprovado em diversos concursos tais como ANAC, BNDES, TCE-RN, INFRAERO e, claro, Banco Central. A

minha trajetória como concurseiro durou pouco mais de dois anos. Neste intervalo, aprendi muito e vou tentar passar um pouco desta minha experiência ao longo deste curso.

O QUE É O PASSO ESTRATÉGICO?

O Passo Estratégico é um material escrito e enxuto que possui dois objetivos principais:

a) orientar revisões eficientes;

b) destacar os pontos mais importantes e prováveis de serem cobrados em prova.

Assim, o Passo Estratégico pode ser utilizado tanto para **turbinar as revisões dos alunos mais adiantados nas matérias, quanto para maximizar o resultado na reta final de estudos por parte dos alunos que não conseguirão estudar todo o conteúdo do curso regular.**

Em ambas as formas de utilização, como regra, **o aluno precisa utilizar o Passo Estratégico em conjunto com um curso regular completo.**

Isso porque nossa didática é direcionada ao aluno que já possui uma base do conteúdo.

Assim, se você vai utilizar o Passo Estratégico:

a) **como método de revisão,** você precisará de seu curso completo para realizar as leituras indicadas no próprio Passo Estratégico, em complemento ao conteúdo entregue diretamente em nossos relatórios;

b) **como material de reta final,** você precisará de seu curso completo para buscar maiores esclarecimentos sobre alguns pontos do conteúdo que, em nosso relatório, foram eventualmente expostos utilizando uma didática mais avançada que a sua capacidade de compreensão, em razão do seu nível de conhecimento do assunto.

Seu cantinho de estudos famoso!

Poste uma foto do seu cantinho de estudos nos stories do Instagram e nos marque:

@passoestrategico

Vamos repostar sua foto no nosso perfil para que ele fique famoso entre milhares de concurseiros!

ANÁLISE ESTATÍSTICA

A análise estatística estará disponível a partir da próxima aula.

ROTEIRO DE REVISÃO E PONTOS DO ASSUNTO QUE MERECEM DESTAQUE

A ideia desta seção é apresentar um roteiro para que você realize uma revisão completa do assunto e, ao mesmo tempo, destacar aspectos do conteúdo que merecem atenção.

Para revisar e ficar bem preparado no assunto, você precisa, basicamente, seguir os passos a seguir:

Lógica de Programação

De forma simples e direta, **lógica de programação** é o modo como se escreve um programa de computador, um algoritmo. Podemos definir "lógica de programação" como a elaboração de sequências de ações para atingir um determinado objetivo. O processo envolve o uso de dispositivos lógicos, como estruturas condicionais (if/else) e de repetição (for/while). Já **um algoritmo** é uma sequência de passos para se executar uma função. Para entendermos melhor, podemos fazer um paralelo fora da computação, com uma receita de bolo.

Na receita, devem-se seguir os passos para o bolo ficar pronto e sem nenhum problema. Na informática, os programadores escrevem as "receitas de bolo" (algoritmos) de modo que o computador leia e entenda o que deve ser feito, ao executar o algoritmo. Para isto é necessária uma **linguagem de programação**.

Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano. Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros. Tal diferença pode ser reflexo da complexidade computacional aplicada, que depende de estruturas de dados adequadas ao algoritmo. Por exemplo, um algoritmo para se vestir pode especificar que você vista primeiro as meias e os sapatos antes de vestir a calça enquanto outro algoritmo especifica que você deve primeiro vestir a calça e depois as meias e os sapatos. Fica claro que o primeiro algoritmo é mais difícil de executar que o segundo apesar de ambos levarem ao mesmo resultado.

A linguagem de programação é como uma língua normal, um grupo de palavras com significados. No caso da programação, a maioria das linguagens é escrita em Inglês. Estas linguagens fazem o computador assimilar cada comando e função de um algoritmo, depois executar cada função.

Na hora de programar alguns passos são indispensáveis, como Declarar Variáveis. Variáveis podem ser escritas por letras ou números, que representam um valor que pode ser mudado a qualquer momento. Cada variável tem um espaço na memória para armazenar seus dados. Porem existem vários tipos de dados, sendo os mais comuns:

- Numérico: todo e qualquer tipo número, positivo ou negativo
- Reais: podem ser positivos ou negativos e decimais.
- Caractere: São os textos. Qualquer número pode entrar aqui, porém não terá função matemática.

Diante de todo o exposto, podemos definir lógica de programação como a capacidade de dividir o problema em partes menores é uma etapa essencial da lógica de programação e precisa ser levada em consideração quando nos deparamos com qualquer exercício/desafio.

Linguagem de Programação

A linguagem de programação é um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.

Não entendeu nada? Vamos a outra definição.

Linguagem de Programação é uma linguagem escrita e formal que especifica um conjunto de instruções e regras usadas para gerar programas (software).

A definição ainda está confusa?

Vamos imaginar o computador como uma super calculadora, capaz de fazer cálculos muito mais rápido que nós, mas para isso devemos dizer para o computador o que deve ser calculado e como deve ser calculado. A função das linguagens de programação é exatamente essa, ou seja, servir de um meio de comunicação entre nós e os computadores.

Desde os primórdios da computação foram sendo desenvolvidas várias linguagens e adaptadas conforme os computadores evoluíram. Com isso, as linguagens de programação foram divididas em **quatro gerações** desde o início da década de 50 até os dias atuais.

Na **primeira geração**, a programação era feita através de linguagem de máquina e a Assembly, que dependiam de um hardware/software para que as tarefas pudessem ser executadas.

Na **segunda geração**, começaram a ser desenvolvidas linguagens mais modernas que fizeram sucesso no mercado e ampliaram o uso de bibliotecas de software. As linguagens que marcaram essa geração foram Fortran (usada por engenheiros e cientistas), Cobol (usada em aplicações comerciais), Algol (usada para suporte às estruturas de controle e tipos de dados) e Basic (usada para propósitos acadêmicos).

Na **terceira geração**, as linguagens de programação também são chamadas de linguagens estruturadas, caracterizada pela sua enorme clareza e estruturação na organização dos dados. Através delas foi possível atribuir recursos inteligentes, criar sistemas distribuídos, etc. São classificadas em:

- linguagens de propósito geral - baseadas na linguagem Algol e podem ser utilizadas para várias aplicações, desde propósitos científicos à comerciais. Ex.: C, Pascal, Modula-2;
- linguagens especializadas - desenvolvidas apenas para uma determinada aplicação. Ex.: linguagem Lisp, usada em engenharia de software para a manipulação de listas e símbolos;

APL criada para manipulação de vetores; Forth criada para o desenvolvimento de softwares para microprocessadores, etc.

- linguagens orientadas a objetos - com mecanismos baseados na semântica e sintática dando apoio à programação orientada a objetos. Ex.: Smalltalk, C++, Delphi, etc.

Na **quarta geração**, também chamadas de linguagens artificiais, houve a criação de sistemas baseados em inteligência artificial. Elas foram divididas em:

- linguagem de consulta - criadas para a manipulação de base de dados gerenciando uma grande quantidade de informações armazenadas em arquivos;
- linguagens geradoras de programas - são linguagens 4GL, que possibilitam a criação de programas complexos, da terceira geração e possuem um nível mais alto que as linguagens de quarta geração;
- outras linguagens - são aquelas usadas em sistemas de suporte à decisão, modelagem de sistemas, linguagens para protótipos, etc.

Na década de 90 e nos anos 2000 surgiram linguagens populares como: Python, Java, Ruby, Javascript, PHP, Delphi, C#, etc. Dando início a uma quinta geração de linguagens de programação.

Existem dois tipos de linguagens de programação: as de baixo nível e as de alto nível. Os computadores interpretam tudo como números em base binária, ou seja, só entendem zero e um. As linguagens de baixo nível são interpretadas diretamente pelo computador, tendo um resultado rápido, porém é muito difícil e incômodo se trabalhar com elas.

Já as linguagens de alto nível são mais fáceis de se trabalhar e de entender, as ações são representadas por palavras de ordem (exemplo: faça, imprima, etc.) geralmente em inglês. Elas foram feitas assim para facilitar a memorização e a lógica. As linguagens de alto nível não são interpretadas diretamente pelo computador, sendo necessário traduzi-las para linguagem binária. A partir daqui entra em ação um programa chamado **compilador**.

Um compilador é um programa de sistema que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em código de máquina para um processador. Em geral, um compilador não produz diretamente o código de máquina, mas sim um programa em linguagem simbólica semanticamente equivalente ao programa em linguagem de alto nível. O programa em linguagem simbólica é então traduzido para o programa em linguagem de máquina através de montadores.

Quando programamos em uma linguagem de programação de alto nível primeiramente criamos um arquivo de texto comum contendo a lógica do programa, ou seja, é onde falamos ao computador como deve ser feito o que queremos. Este arquivo de texto é chamado de código-fonte, cada palavra de ordem dentro do código-fonte é chamada de instrução. Após criarmos o código-fonte devemos traduzir este arquivo para linguagem binária usando o compilador correspondente com a linguagem na qual estamos programando. O compilador irá gerar um segundo arquivo que chamamos de executável ou programa, este arquivo gerado é interpretado diretamente pelo computador.

Para desempenhar suas tarefas, um compilador deve executar dois tipos de atividade. A primeira atividade é a análise do código fonte, onde a estrutura e significado do programa de alto nível são reconhecidos. A segunda atividade é a síntese do programa equivalente em linguagem simbólica. Embora conceitualmente

seja possível executar toda a análise e apenas então iniciar a síntese, em geral estas duas atividades ocorrem praticamente em paralelo.

Linguagens de Script

Inicialmente havia linguagens de programação para construir programas como Excel, Word, navegadores, jogos, etc. Esses programas foram construídos com linguagens de programação como C ++ e Java. Com o tempo, esses programas precisavam de uma opção que permitisse aos usuários criar novas funcionalidades. Então eles tinham que oferecer uma interface para suas funções básicas e assim as linguagens de script nasceram.

Características

- Ausência de declaração / regras de escopo simples - A maioria das linguagens de script dispensam a declaração de variáveis. Quanto ao escopo, são adotadas regras simples. Em Perl, por exemplo, por padrão tudo é global. Em PHP, é exatamente ao contrário. Em Python, as variáveis são locais ao bloco onde são atribuídas.
- Tipagem - A maioria possui tipagem dinâmica flexível. Em algumas linguagens o tipo de uma variável é checado imediatamente antes do seu uso. Em outras, a variável será interpretada dependendo do contexto onde aparece.
- Facilidade de acesso à funcionalidade do sistema - Quando comparamos as linguagens de script as linguagens de programação, as funcionalidades de sistema são muito mais facilmente acessadas por linguagens de script (entrada, saída, manipulação de arquivos, gerenciamento de processos, proteção e autorização, relógio, acesso à rede...).
- Manipulação sofisticada de strings - Normalmente baseada em expressões regulares, herança das linguagens de processamento de texto.

- Tipos de dados de alto-nível - Estruturas como conjuntos e dicionários são extremamente comuns. Várias são definidas usando a própria linguagem. Em todas há implementação de coleta automática de lixo.

Vamos ver dois exemplos de linguagens de scripts:

PHP: Hypertext Preprocessor

Numa explicação de poucas palavras, PHP (acrônimo recursivo para *PHP: Hypertext Preprocessor*) é uma linguagem de programação utilizada por programadores e desenvolvedores para construir sites dinâmicos, extensões de integração de aplicações e agilizar no desenvolvimento de um sistema.

O PHP é uma linguagem de programação de código aberto, criada para o desenvolvimento web. Ele também é um subconjunto de linguagens de scripts como *JavaScript* e *Python*. A diferença é que PHP costuma ser mais usado para comunicação do lado do servidor (*frontend*).

> Uma linguagem de script serve para automatizar a execução de tarefas num ambiente de tempo de execução especial. Isso inclui dizer para uma página estática (construída com HTML e CSS) para executar ações específicas com regras definidas anteriormente.

Linguagens de script podem rodar tanto do lado do servidor (*backend*) quanto do lado do cliente (*frontend*). Scripts do tipo *frontend* são processados pelos navegadores. Quando o seu browser – também conhecido como o cliente – solicita uma página contendo scripts do lado do cliente, o servidor responde enviando os códigos-fontes que são executáveis pelo navegador.

Por outro lado, linguagem de scripts do tipo *backend* significam que esses scripts são executados nos servidores antes de serem enviados ao navegador. Então ao invés de mandar o código-fonte, os servidores da web processam (parse) os códigos antes ao transformá-los num formato HTML puro.

Apesar do PHP ser considerado uma linguagem de scripts de propósito geral, ela é mais usada para desenvolvimento na web. Isso acontece por causa de um de seus recursos mais notáveis: a habilidade de ser integrado num arquivo HTML.

Você pode estar pensando: "Certo professor, mas o que isso significa?". Vamos a um exemplo:

```
<!DOCTYPE                                                    HTML>
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>
```

```
        <?php
            echo "Olá, eu sou um script PHP!";
        ?>

    </body>
</html>
```

Em vez de muitos comandos para mostrar HTML (como acontece com C ou Perl), as páginas PHP contêm HTML em código mesclado que faz "alguma coisa" (neste caso, mostra "Olá, eu sou um script PHP!"). O código PHP é delimitado pelas instruções de processamento (*tags*) de início e fim <?php e ?> que permitem que você entre e saia do "modo PHP".

O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. O navegador recebe os resultados da execução desse script, mas não sabe qual era o código fonte. Também é possível configurar o servidor web para processar todos os seus arquivos HTML com o PHP.

Características e Recursos

Uma das características mais fortes e mais significativas do PHP é seu suporte a uma ampla variedade de banco de dados. Escrever uma página web consultando um banco de dados é incrivelmente simples usando uma das extensões específicas de banco de dados (por exemplo, mysql), ou usando uma camada de abstração como o PDO ou conectar a qualquer banco de dados que suporte o padrão "*Open Database Connection*" usando a extensão ODBC. Outros bancos de dados podem utilizar cURL ou sockets, como o CouchDB.

O PHP também tem suporte para comunicação com outros serviços utilizando protocolos como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (no Windows) e incontáveis outros. É possível abrir sockets de rede e interagir diretamente usando qualquer outro protocolo. O PHP também suporta o intercâmbio de dados complexos WDDX, utilizado em virtualmente todas as linguagens de programação para web. Falando de comunicação, o PHP implementa a instanciação de objetos Java e os utiliza transparentemente como objetos PHP.

O PHP tem recursos úteis para processamento de texto, incluindo expressões regulares compatíveis com Perl (PCRE), e muitas outras extensões e ferramentas para analisar e acessar documentos XML. Ele padroniza todas as extensões XML a partir da base sólida da libxml2, além de estender o conjunto de recursos adicionando suporte a SimpleXML, XMLReader e XMLWriter.

Possibilidades de Uso

Existem três áreas principais onde os scripts PHP são usados:

- **Scripts no lado do servidor (*server-side*).** Este é o mais tradicional e principal campo de atuação do PHP. São necessárias três coisas para isto funcionar: o interpretador do PHP (CGI ou módulo do servidor), um servidor web e um navegador web. É necessário rodar o servidor web conectado a uma

instalação do PHP. É possível acessar os resultados do programa PHP com um navegador web, visualizando a página PHP através do servidor web. Tudo isso pode rodar em uma máquina pessoal, caso o desenvolvedor esteja apenas experimentando programar com o PHP.

- **Scripts de linha de comando.** É possível fazer um script PHP para executá-lo sem um servidor ou navegador. A única coisa necessária é o interpretador PHP. Esse tipo de uso é ideal para script executados usando o cron (Unix, Linux) ou o Agendador de Tarefas (no Windows). Esses scripts podem ser usados também para rotinas de processamento de texto simples.
- **Escrever aplicações desktop.** O PHP provavelmente não é a melhor linguagem para criação de aplicações desktop com interfaces gráficas, mas se o desenvolvedor conhece bem o PHP, e gostaria de usar alguns dos seus recursos avançados nas suas aplicações do lado do cliente, é possível usar o PHP-GTK para escrever programas assim. Existe também tem a possibilidade de escrever aplicações multi-plataformas desse jeito.

JavaScript

A primeira coisa que você precisa saber: JavaScript não tem nada a ver com Java. Java é uma linguagem server-side, como PHP, Ruby, Python e tantas outras. A única coisa parecida entre eles é o nome.

JavaScript (às vezes abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional).

```
<!DOCTYPE html>
<html>
<body>

<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button"
onclick="document.getElementById('demo').innerHTML = 'Hello JavaScript!'">
Click Me!</button>

</body>
</html>
```

Ao invés de rodar remotamente em servidores na Internet, o JavaScript tem como característica rodar programas localmente - do lado do cliente. Dessa forma, o JavaScript fornece às páginas web a possibilidade de programação, transformação e processamento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HTML e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas.

Scripts de código escritos nessa linguagem e executados em um navegador permitem, por exemplo, atualizar parte do conteúdo de uma página web sem carregá-la totalmente após preencher um formulário, através de técnicas de programação como o AJAX. Isso permite a criação de uma infinidade de softwares

completos e totalmente funcionais para diversas finalidades. O Google Docs jamais funcionaria sem a existência do JavaScript, por exemplo.

Desenvolvimento por camadas

Separando a informação, formatação e comportamento

Basicamente existem três separações no desenvolvimento web: Informação, que fica com o código HTML, onde você vai dar significado à informação. Formatação, que fica com o CSS, que é como você vai dar estilo a toda informação marcada com HTML. E por fim a camada de Comportamento, que é como você vai controlar elementos do HTML com Javascript e seus frameworks.

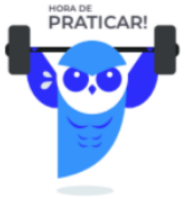
Camada de comportamento

O JavaScript é a terceira camada de desenvolvimento por que ele manipula as duas primeiras camadas, isto é: HTML e CSS. Imagine que você precise de um Slider de imagens. Toda a movimentação, ações de cliques nas setinhas e etc, é o JavaScript que vai cuidar. É isso que chamamos de comportamento.

QUESTÕES ESTRATÉGICAS

Nesta seção, apresentamos e comentamos uma amostra de questões objetivas selecionadas estrategicamente: são questões com nível de dificuldade semelhante ao que você deve esperar para a sua prova e que, em conjunto, abordam os principais pontos do assunto.

A ideia, aqui, não é que você fixe o conteúdo por meio de uma bateria extensa de questões, mas que você faça uma boa revisão global do assunto a partir de, relativamente, poucas questões.



1.

Considere, hipoteticamente, que um Técnico do TRE-SP tem, em seu computador, a seguinte organização de um diretório:

Principal: Dados

Dentro de Dados: Técnicos Práticos

Dentro de Técnicos: Árvores Hash Recursão Filas Pilhas

Dentro de Práticos: Programas AFazer Prontos

Dentro de Prontos: Eleições Urnas

Dentro de Programas: Corretos ComErro

Dentro de ComErro: Urgentes Pendentes Antigos

A estrutura de dados

a) fila é a mais adequada para representar este diretório.

b) pilha é a mais adequada para representar este diretório.

c) árvore binária, ao armazenar este diretório, terá Dados na raiz e nós com grau 2, 3, 5 e folhas.

d) árvore, que consegue armazenar este diretório, é de ordem 5.

e) hashing, ao armazenar este diretório, não terá colisões na tabela de dispersão

Comentários

Uma árvore é uma estrutura de dados usada principalmente para representar dados com uma relação hierárquica entre seus elementos. Os elementos de uma árvore são denominados "nós" e existe um nó especial chamado "Raiz da árvore" (não é filho de nenhum elemento).

O número máximo de filhos de um elemento é chamado ordem da árvore. Vamos analisar a organização do diretório:

- Dados
  - Técnicos
    - Árvores
    - Hash
    - Recursão
    - Filas
    - Pilhas
  - Práticos
    - Programas
      - Corretos
      - ComErro
        - Urgentes
        - Pendentes
        - Antigos
- AFazer
- Prontos
  - Eleições
  - Urnas

Podemos ver que a pasta "Técnicos" possui cinco subpastas. Dessa forma, a ordem dessa árvore será 5.

Gabarito: alternativa D.

2.

Estruturas de dados básicas, como as pilhas e filas, são usadas em uma gama variada de aplicações. As filas, por exemplo, suportam alguns métodos essenciais, como o

a) enqueue(x), que insere o elemento x no fim da fila, sobrepondo o último elemento.

b) dequeue(), que remove e retorna o elemento do começo da fila; um erro ocorrerá se a fila estiver vazia.

c) push(x), que insere o elemento x no topo da fila, sem sobrepor nenhum elemento.

d) pop(), que remove o elemento do início da fila e o retorna, ou seja, devolve o último elemento inserido.

e) top(), que retorna o elemento do fim da fila sem removê-lo; um erro ocorrerá se a fila estiver vazia.

Comentários

Fila (queue) é uma estrutura de dados baseada na política FIFO (first in, first out), ou seja, o primeiro objeto inserido na fila é também o primeiro a ser removido. Nessa estrutura, o elemento a ser removido é o que foi inserido a mais tempo. Segundo Cormen, a operação INSERT, chamada ENQUEUE, adiciona um elemento ao final da fila, e a operação DELETE, chamada de DEQUEUE, remove o elemento no início da fila.

A operação DEQUEUE só pode ser aplicada se a fila não estiver vazia, para evitar o erro de underflow ou fila vazia.

Gabarito: alternativa B.

3.

Durante o desenvolvimento de uma aplicação orientada a objetos com Java, um Técnico criou uma interface para obrigar um conjunto de classes de diferentes origens a implementar certos métodos de maneiras diferentes, viabilizando a obtenção de polimorfismo. A interface criada pelo Técnico pode

a) conter métodos implementados.

b) ser instanciada diretamente.

c) possuir um único construtor vazio.

d) possuir métodos abstratos.

e) conter variáveis e métodos privados.

Comentários

A interface é uma espécie de "contrato" que descreve tudo que uma classe deve fazer. Quem assina esse "contrato" fica responsável por "colocar em prática" o que está especificado. Ou seja, a interface define métodos (que são abstratos) e quem implementa a interface escreve os métodos.

Gabarito: alternativa D.

4.

São, dentre outros, recursos essenciais em uma aplicação orientada a objetos para se obter polimorfismo:

a) Herança e sobrescrita de métodos.

b) Classes estáticas, com métodos protegidos.

c) Interfaces, contendo métodos não abstratos e implementados.

d) Classes abstratas, sem subclasses.

e) Arrays unidimensionais ou multidimensionais.

Comentários

Polimorfismo significa que um objeto pode ser tratado como se fosse um tipo diferente de objeto, desde que seja com bom senso. Isso está muito relacionado com o conceito de herança. A herança é usada para criar objetos que tem "tudo que outro objeto tinha, mas também possui alguns detalhes próprios". A herança nos permite pegar uma classe e utilizar ou alterar suas propriedades e métodos incluindo nossas propriedades, métodos.

Gabarito: alternativa A.

5.

A análise e o projeto orientados a objeto modelam um sistema em termos de objetos, que têm propriedades e comportamentos, e de eventos, que disparam operações que mudam o estado dos objetos. Considere, abaixo, os fundamentos da Orientação a Objetos – OO:

I. Ato de empacotar ao mesmo tempo dados e objetos. O objeto esconde seus dados de outros objetos e permite que os dados sejam acessados por intermédio de seus próprios métodos. Protege os dados do objeto do uso arbitrário e não intencional. Separa a maneira como um objeto se comporta da maneira como ele é implementado.

II. Refere-se à implementação de um tipo de objeto. Especifica uma estrutura de dados e as operações permissíveis que se aplicam a cada um de seus objetos.

III. Pode ser real ou abstrato. Possui informações (dados) e desempenha ações (funcionalidades). É qualquer coisa, real ou abstrata, a respeito da qual são armazenados dados e operações.

Os conceitos da OO indicados em I, II e III, são, correta e respectivamente,

a) Herança, Classe, Atributo.

b) Encapsulamento, Método, Objeto.

c) Polimorfismo, Superclasse, Método.

d) Encapsulamento, Classe, Objeto.

e) Herança, Método, Atributo.

Comentários

Analisando os itens, temos:

I - O mecanismo ou conceito OO que "esconde" os detalhes internos de implementação dos objetos do mundo externo é o Encapsulamento. É alcançado aplicando-se modificadores de visibilidade ou acesso. Seus principais benefícios são:

- redução dos impactos propagados a partir de mudanças

- diminuição do acoplamento (dependência) entre classes e objetos

- simplificação das interfaces entre os objetos

II - Classe é o elemento OO que especifica uma estrutura de dados (atributos) e define operações (métodos). São estruturas criadas em linguagens orientadas a objeto que tem como objetivo ser um modelo (molde) para objetos. As classes abstraem um conjunto de objetos semelhantes do mundo real, especificando apenas detalhes que são importantes para o domínio ao qual estará inserido.

III - Objeto é a instância de uma determinada classe. Os dados ou informações são armazenados em seus atributos e as funcionalidades ou ações desempenhadas pelos seus métodos. Segundo Grady Booch, um objeto tem estado, apresenta um comportamento bem definido, e tem uma identidade única. É uma abstração que representa uma entidade do mundo real podendo especificar algo concreto (computador, carro) ou abstrato (transação bancária, histórico, taxa de juros).

Gabarito: alternativa D.

6.

Na orientação a objetos com Java as classes

a) permitem apenas um construtor, que pode ser declarado explicitamente ou não.

b) estáticas são necessárias para se conseguir polimorfismo.

c) podem possuir variáveis com modificadores de acesso *public, private, protected e void*.

d) precisam possuir um método *main* por meio do qual são instanciados os objetos.

e) podem possuir métodos com o mesmo nome, desde que recebam parâmetros de tipos diferentes.

Comentários

Vamos analisar as alternativas.

a) ERRADA. É possível uma classe ter mais de um construtor que receba parâmetros que retratem como o objeto deve ser iniciado

b) ERRADA. Polimorfismo pode ser paramétrico, *overloading* e *overriding*, completamente sem relação com estáticas.

c) ERRADA.

d) ERRADA. O método *main* não é necessário para instanciar classes

e) CORRETA.

Gabarito: alternativa E.

7.

Em uma aplicação Java, a anotação @Override na linha anterior à declaração do método indica que ele

a) está sendo sobrecarregado.
b) é estático.
c) faz referência a outros métodos.
d) está sendo sobrescrito.
e) é abstrato.

Comentários

Cuidado para não confundir *override* com *overload*.

Override é quando um método é sobrescrito.

Não é algo obrigatório, mas sua utilização é considerada uma boa prática.

Gabarito: alternativa D.

8.

Na orientação a objetos, no que se refere à sobrecarga de métodos, um método é considerado sobrecarregado se

a) tiver nome diferente de outros métodos da mesma classe.

b) for público, estático e receber mais de um parâmetro.

c) tiver o mesmo nome de outro método da mesma classe, mas receber parâmetros diferentes.

d) tiver, em uma subclasse, o mesmo nome de um método da superclasse.

e) a quantidade de parâmetros que receber excede o limite permitido.

Comentários

Dizemos que o método está sobrecarregado (*overloading*) quando na definição de uma classe criamos métodos com o mesmo nome, porém com argumentos diferentes.

Ou seja, sobrecarga de método permite a existência de vários métodos de mesmo nome, porém com assinaturas levemente diferentes, ou seja, variando no número, tipo de argumentos, no valor de retorno e até variáveis diferentes. Ficará a cargo do compilador escolher de acordo com as listas de argumentos os procedimentos ou métodos a serem executados.

Gabarito: alternativa C.

9.

A afirmativa I refere-se à Programação Estruturada (PE) e a afirmativa II refere-se à Programação Orientada a Objetos (POO). A alternativa que traz as duas afirmativas verdadeiras é:

a) I - Em linguagens estruturadas, como o Assembly, o programador sempre cria códigos de difícil leitura, pois nesse tipo de linguagem os saltos (jumps) estão sempre presentes.

II - A POO provê uma melhor organização do código e contribui para o reaproveitamento de código, mas seus conceitos são de difícil compreensão se comparados aos conceitos da PE.

b) I - A PE possibilita que o programador tenha maior controle sobre o fluxo de execução do programa. Para isso, pode utilizar estruturas de sequência, estruturas de decisão e estruturas de repetição.

II - Os métodos definem o comportamento dos objetos, tendo seus nomes normalmente definidos por verbos. Para uma classe Pessoa, por exemplo, poderia haver os métodos comprar, vender e alugar.

c) I - Uma característica da PE são os saltos (jumps), que funcionam da seguinte forma: o programador define um label no código e depois, a partir de qualquer parte do programa, ele pode executar um desvio de fluxo de execução para aquele label, mediante a avaliação positiva de uma condição.

II - Classe é o molde para criar objetos. Possui todas as especificações de um grupo deles. As interfaces definem características de objetos, por exemplo, uma classe Pessoa pode ter as interfaces Nome, Endereço e Telefone.

d) I - A depuração de um código com muitos labels e saltos (jumps), dificulta o entendimento do fluxo de execução de um programa estruturado.

II - Herança é a capacidade de criar classes a partir de uma superclasse. Essas classes herdam, então, todas as características da superclasse. Encapsulamento é o princípio pelo qual uma classe sobrescreve um comportamento herdado de sua superclasse.

e) I - A PE baseia-se no que deve ser feito e não em como a tarefa deve ser feita. Tende a gerar códigos em que os tratamentos dos dados são misturados com o comportamento do programa.

II - Polimorfismo é a habilidade de esconder de outros objetos, as características intrínsecas de um dado objeto. Toda a comunicação entre objetos deve ser realizada através de interfaces. Um objeto não deve ser capaz de acessar nem alterar métodos de outro objeto diretamente.

Comentários

Vamos analisar as alternativas.

a) ERRADO. O trecho "mas seus conceitos são de difícil compreensão se comparados aos conceitos da PE" torna a alternativa incorreta, porque POO foi feito para ser de fácil entendimento, utilizando conceitos de abstração do mundo real.

b) CERTO.

c) ERRADO. O trecho "As interfaces definem características de objetos, por exemplo, uma classe Pessoa pode ter as interfaces Nome, Endereço e Telefone" torna a alternativa incorreta pois não são as interfaces, mas sim os estados (atributos).

d) ERRADO. O trecho "Encapsulamento é o princípio pelo qual uma classe sobrescreve um comportamento herdado de sua superclasse" torna a alternativa incorreta pois essa definição é de polimorfismo.

e) ERRADO. O trecho "Polimorfismo é a habilidade de esconder de outros objetos, as características intrínsecas de um dado objeto. Toda a comunicação entre objetos deve ser realizada através de interfaces. Um objeto não deve ser capaz de acessar nem alterar métodos de outro objeto diretamente" torna a alternativa incorreta pois essa definição é de encapsulamento.

Gabarito: alternativa B.

...

Forte abraço e bons estudos!

"Hoje, o 'Eu não sei', se tornou o 'Eu ainda não sei'"

(Bill Gates)

Thiago Cavalcanti

Face: www.facebook.com/profthiagocavalcanti
Insta: www.instagram.com/prof.thiago.cavalcanti
YouTube: *youtube.com/profthiagocavalcanti*

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.