Linguagem Orientada a Objetos

O programador James Gosling é conhecido mundialmente por ser o pai da linguagem de programação Java. Gosling é canadense e, em 1977, se formou bacharel em ciência da computação pela Universidade de Calgary. No ano de 1983, obteve PhD em Ciência da Computação pela Universidade Carnegie Mellon. A linguagem Java foi criada quando Gosling trabalhava na Sun Microsystems. Ele permaneceu nessa empresa de 1984 até abril de 2010, quando se demitiu. Entre outras contribuições feitas por Gosling, temos mais de 12 livros publicados, a maioria dos quais é sobre Java. Ele já recebeu diversos prêmios, entre os quais a medalha IEEE John Von Neumann, concedida a pessoas que realizaram trabalhos de destaque na área de computação.

Teste de Mesa

O teste de mesa é a simulação da execução de algum programa, algoritmo ou método e pode ser feito de forma manual ou automatizada.

O teste de mesa manual, normalmente, é feito no papel, em que os valores de cada variável são escritos e acompanhados por meio de um passo a passo. De forma geral, não existem regras rígidas para isso.

O teste de mesa automatizado, normalmente, é feito acompanhado-se a execução linha a linha com o auxílio de algum software, como o Java Tutor.

Java Tutor

A seguir, você pode testar o código, verificando o passo a passo de sua execução.

- Ao clicar em “Next”, você irá para a próxima linha a ser executada.
- Ao clicar no botão “Prev”, você retorna a linha executada anteriormente.
- A seta verde na linha de código representa a linha em execução.
- A seta vermelha representa a próxima linha a ser executada.

```
package avaliacaofaculdade;

public class AvaliacaoFaculdade {
    public static void main(String args[]) {
        int n = 10;
        int v[] = new int[n];
        int i;

        v[0] = 1025;
        v[1] = 547;
        v[2] = 415;
        v[3] = 497;
        v[4] = 8877;
        v[5] = 13;
        v[6] = 4887;
        v[7] = 1267;
        v[8] = 5456;
        v[9] = 1781;
```

```
        int soma = 0;
        int menor = v[0];
        int maior = v[0];

        for(i = 0; i < n; i++) {
           soma = soma + v[i];

           if(v[i] < menor)
              menor = v[i];

           if(v[i] > maior)
              maior = v[i];
        }

        System.out.println("");

        for(i = 0; i < n; i++) {
           if(v[i] == menor)
              System.out.printf("v[%d] = %2d <--- menor valor.\n", i, v[i]);
           else if(v[i] == maior)
              System.out.printf("v[%d] = %2d <--- maior valor.\n", i, v[i]);
           else
              System.out.printf("v[%d] = %2d\n", i, v[i]);
        }

        System.out.printf("\nSoma = %d\n", soma);
     }
}
```

Modificadores de Acesso

Por meio dos modificadores de acesso, podemos restringir ou permitir que um atributo, método, construtor e/ou classe seja acessado ou não. Em Java, temos quatro modificadores de acesso, que são: Público, privado, protegido e default.

Restrição e Acessibilidade

A seguir podemos ver os quatro modificadores de acesso presentes em Java e o relacionamento de restrição baseado em cada um deles, do modificador mais restritivo para o menos restritivo, temos: Privado, default, protegido e público.

A seguir, também podemos ver os quatro modificadores de acesso presentes em Java, bem como a capacidade de se acessar ou visualizar os atributos, métodos, construtores ou, ainda, as classes com base nos modificadores de acesso.

Reutilização de Classes

A ideia de reutilização de classes e reaproveitamento de código é uma das ideias centrais da linguagem Java, que é extremamente rica em bibliotecas que provêm diversos serviços e funcionalidades prontas.

Porém, antes de criarmos alguma aplicação, é sempre bom, antes de sairmos implementando algo, conferir se já existe alguma solução pronta ou, ainda, alguma biblioteca que nos ajude a resolver o nosso problema. Dessa maneira, é importante consultar os repositórios de código MVN Repository, GitHub e GitLab.

Tratamento de Exceções

A fim de apresentarmos, de forma lúdica o tratamento de exceção em Java, podemos ver a seguir uma pessoa (programa) correndo por uma determinada rota. No entanto, no meio do caminho, ela encontra um obstáculo (exceção). Ao avistar o obstáculo, tal pessoa não para e aborta a sua missão (correr), ela simplesmente lida com esse obstáculo (exceção) desviando um pouco a sua rota para o lado. A ideia por trás do tratamento de exceção é exatamente esta: Conseguir contornar os obstáculos (exceções) que possam aparecer durante o programa.

Execução da Aplicação

Vimos a construção de aplicativos que recebem parâmetros via linha de comando, bem como a compilação e execução da aplicação Java por meio da linha de comando. Apesar disso, existem outras formas de execução da aplicação, como, por exemplo, pela IDE (a maioria das IDEs suportam isso). Dessa maneira, pesquise na internet como passar argumentos para executar um programa utilizando a sua IDE de preferência, assim, você não precisará entrar no terminal para executar a sua aplicação, agilizando, então, o desenvolvimento do código.

Interfaces Gráficas

Antes de se construir uma aplicação gráfica que envolva uma certa complexidade, faz-se importante projetar, em primeiro lugar, a tela. Assim, os projetistas de interfaces gráficas costumam desenhar as telas antes de saírem implementando código (isso facilita muito).

Uma das estratégias é desenhara tela em uma folha de papel. Após chegar a uma boa versão, fica mais fácil implementar o que se desenhou.

Outra estratégia é utilizar algum software que cria mockups, como o Balsamiq (existem dezenas de outros softwares que fazem isso).

Maquete ou Mockup

A maquete ou mockup ilustra um esqueleto/rascunho de como a tela de uma aplicação hipotética. Somente na versão final da tela é que terá início a implementação da interface gráfica.

Arquivo Executável

Após a criação de uma aplicação gráfica, em muitos momentos nos depararemos com a necessidade de gerar um executável dessa aplicação. Vimos anteriormente como gerar um arquivo .jar da nossa aplicação por meio do ambiente de desenvolvimento integrado (IDE). Após construir a sua aplicação gráfica, procure sempre gerar o .jar e execute a sua aplicação por meio desse arquivo. A depender das configurações do seu computador, pode ser necessário habilitar o arquivo .jar dando um duplo clique no arquivo.

Principais Características da Linguagem Java

Essas são 12 das mais importantes características da linguagem Java:

- Orientada a Objetos.
- Simples.
- Segura.
- Independente de Plataforma.
- Robusta.
- Portável.
- Arquitetura Neutra.
- Dinâmica.
- Interpretada.
- Alto Desempenho.
- Multithread.
- Distribuída.

As classes String, StringBuilder e StringBuffer possuem diversos métodos para manipulação de strings que se destacam: toLowerCase, toUpperCase, join, split, substring, concat, replace e trim. Esses métodos são extremamente importantes para realizar a manipulação de strings.

Banco de Dados Relacional e NoSQL

Propriedades de uma Transação ACID

O infográfico a seguir ilusta 4 importantes propriedades presentes em um banco de dados relacional, como o MySQL e o MariaDB:

- **Atomicidade:** Unidade lógica atômica (tudo ou nada – toda a operação é executada ou toda a operação falha.
- **Consistência:** Ao final de uma transação, o banco continua consistente.
- **Isolamento:** A execução de uma transação não deve sofrer interferência de outras transações concorrentes.
- **Durabilidade:** Após o ponto de confirmação, as alterações devem persistir no BD.

O XAMPP e o PHPMyAdmin são algumas das ferramentas utilizadas na criação de um banco de dados. O conhecimento dessas ferramentas é muito importante para que se consiga criar e manipular um BD.

Fluxo de Execução/Escalonamento

- **Processos de Threads:** Existem N processos em execução no computador, cada processo tem, no mínimo, uma thread e, no máximo M threads. É importante destacar que cada processo pode estar vinculado a uma aplicação diferente.
- **Sistema Operacional:** Todos esses processos e threads são executadas sobre um Sistema Operacional (SO). Uma das partes do SO é o escalonador de processos, que pega um conjunto de processos e threads e o direciona para que o hardware o execute.
- **CPU:** O hardware do computador em questão possui K núcleos (cores), dessa maneira, o SO direciona cada uma das threads para um núcleo (core) diferente executar, é o escalonador de processos que decide quem vai executar, por quando tempo vai executar e em qual núcleo vai ser executado.

A classe Thread possui diversos métodos para manipulação de thread, destacando-se: start, sleep, isAlive e join, que são extremamente importantes para a realização da manipulação de threads.