Curso Java backend

Módulo I – Introdução à linguagem Java

Sumário

Autor: Ubirajara Portela Campos

Conceitos Básicos de Programação

Linguagem de Programação

Uma linguagem de programação é um conjunto de regras e sintaxe que permite aos programadores escreverem instruções que um computador pode entender e executar. As linguagens de programação são usadas para criar programas de software, aplicativos e scripts. Elas fornecem uma maneira de comunicar-se com o hardware do computador para realizar tarefas específicas.

Lógica de Programação

A lógica de programação é a sequência de instruções que um programa deve seguir para resolver um problema ou executar uma tarefa. Ela envolve a utilização de estruturas de controle, como loops e condicionais, para manipular dados e alcançar o resultado desejado. A lógica de programação é fundamental para o desenvolvimento de software eficiente e funcional.

Tipos de Linguagens de Programação

As linguagens de programação podem ser classificadas de várias maneiras, com base em diferentes critérios. Aqui estão algumas classificações comuns:

Quanto ao Nível de Abstração

1. **Linguagens de Baixo Nível**:
    - **Assembly**: Uma linguagem de baixo nível que está diretamente relacionada ao código de máquina específico de um processador. Permite controle detalhado sobre o hardware, mas é complexa e difícil de usar.
2. **Linguagens de Alto Nível**:
    - **Python, Java, C#**: Linguagens que abstraem muitos detalhes do hardware, tornando o desenvolvimento mais simples e eficiente. São mais fáceis de aprender e usar, mas podem sacrificar algum desempenho em comparação com linguagens de baixo nível.

Quanto ao Paradigma de Programação

1. Linguagens Procedurais:
    - C, Pascal: Focadas em funções ou procedimentos que operam em dados. A lógica do programa é dividida em sub-rotinas ou funções.
2. Linguagens Orientadas a Objetos:
    - **Java, C++, Python**: Baseadas no conceito de "objetos", que são instâncias de "classes" que encapsulam dados e métodos. Facilitam a reutilização de código e a manutenção.
3. Linguagens Funcionais:
    - **Haskell, Lisp, Erlang**: Enfatizam a aplicação de funções, evitando estados e mutabilidade. São usadas para resolver problemas matemáticos e algoritmos complexos de maneira elegante.
4. Linguagens de Script:
    - **JavaScript, Ruby, Python**: Usadas principalmente para automatizar tarefas e adicionar funcionalidades a aplicações web. São interpretadas e frequentemente mais fáceis de usar para tarefas específicas.

Quanto ao Método de Execução

1. **Linguagens Compiladas**:
    - **C, C++, Go**: O código-fonte é transformado em código de máquina por um compilador antes de ser executado. Programas compilados geralmente têm melhor desempenho porque são otimizados para a arquitetura específica do hardware.
2. **Linguagens Interpretadas**:
    - **Python, JavaScript, Ruby**: O código-fonte é executado diretamente por um interpretador, que lê e executa as instruções linha por linha. São mais flexíveis e fáceis de depurar, mas podem ser mais lentas do que linguagens compiladas.
3. **Linguagens Híbridas**:
    - **Java, C#**: Compiladas para bytecode intermediário, que é então executado por uma máquina virtual (JVM para Java, CLR para C#). Combinam a eficiência das linguagens compiladas com a flexibilidade das interpretadas.

O que é Java?

Java é uma linguagem de programação e uma plataforma de computação desenvolvida pela Sun Microsystems em 1995, que atualmente é mantida pela Oracle Corporation. Java é amplamente utilizada no desenvolvimento de software devido à sua portabilidade, robustez e segurança.

Principais características

1. **Linguagem de Programação**

- **Orientada a Objetos:** Java é uma linguagem de programação orientada a objetos (OOP), o que significa que se baseia no conceito de "objetos" que contêm dados e métodos.
- **Sintaxe Inspirada em C/C++:** A sintaxe de Java é semelhante à das linguagens C e C++, facilitando a transição para programadores familiarizados com essas linguagens.

2. **Plataforma**

- **Java Virtual Machine (JVM):** A JVM é uma máquina virtual que executa programas Java. Ela permite que os programas Java sejam executados em qualquer dispositivo ou sistema operacional que tenha uma JVM instalada, promovendo a ideia de "Write Once, Run Anywhere" (Escreva uma vez, execute em qualquer lugar).
- **Java Runtime Environment (JRE):** O JRE inclui a JVM e bibliotecas de classes necessárias para rodar aplicações Java.
- **Java Development Kit (JDK):** O JDK é um pacote que inclui o JRE e ferramentas de desenvolvimento como o compilador (javac), que é usado para compilar código-fonte Java em bytecode que a JVM pode executar.

3. **Portabilidade**

Devido à JVM, programas escritos em Java podem ser executados em diferentes plataformas sem a necessidade de modificação do código.

4. **Multithreading**
   Suporta multithreading, permitindo que programas realizem várias tarefas simultaneamente.

5. Áreas de Aplicação

- **Desenvolvimento Web:** Java é amplamente utilizada no desenvolvimento de aplicações web através de frameworks como Spring e Hibernate.
- **Aplicações Móveis:** Android, o sistema operacional móvel mais popular, utiliza Java como uma das principais linguagens de programação.
- **Sistemas Empresariais:** Muitas grandes corporações usam Java para desenvolver sistemas corporativos devido à sua robustez e escalabilidade.
- **IoT (Internet das Coisas):** Java também é usada no desenvolvimento de aplicações para dispositivos de IoT devido à sua portabilidade e segurança.

6. Evolução e Comunidade

- **Atualizações Constantes:** Java está em constante evolução, com novas versões lançadas regularmente para introduzir novas funcionalidades, melhorias de desempenho e segurança.
- **Comunidade Ativa:** A comunidade Java é uma das maiores e mais ativas, oferecendo vasto suporte, bibliotecas open-source, e frameworks.

IDE

IDE significa Integrated Development Environment (Ambiente de Desenvolvimento Integrado). É uma aplicação de software que oferece um conjunto abrangente de ferramentas para facilitar o desenvolvimento de software. Uma IDE típica inclui:

- Editor de Código: Onde os desenvolvedores escrevem o código fonte.
- Compilador/Interpretador: Para traduzir o código em linguagem que o computador pode executar.
- Depurador: Para encontrar e corrigir erros no código.
- Ferramentas de Gerenciamento de Projeto: Para organizar e gerenciar os arquivos do projeto.
- Outras Funcionalidades: Como controle de versão, visualização de estrutura de código, e integração com outras ferramentas de desenvolvimento.

Algumas das IDE´s mais utilizadas para desenvolvimento em JAVA

IntelliJ IDEA

Desenvolvida pela JetBrains, o IntelliJ IDEA é amplamente considerada uma das melhores IDEs para desenvolvimento Java devido à sua inteligência de código, depuração avançada, integração de ferramentas, e suporte a frameworks modernos como Spring e Hibernate.

Eclipse

Eclipse é uma IDE de código aberto, muito popular e amplamente utilizada no desenvolvimento Java. É conhecida por sua extensibilidade através de plugins e por ser uma ferramenta robusta para grandes projetos.

Visual Studio Code

Visual Studio Code (VS Code) é um editor de código-fonte leve, mas poderoso, desenvolvido pela Microsoft. Embora não seja uma IDE completa por padrão, ele pode ser configurado para desenvolvimento Java com a adição de extensões.

Configurando o Ambiente de Desenvolvimento Java

Para começar a programar em Java, você precisa configurar seu ambiente de desenvolvimento instalando o Java Development Kit (JDK) e uma IDE (Integrated Development Environment). Neste guia, vamos usar o IntelliJ IDEA, uma das IDEs mais populares para o desenvolvimento em Java, e estamos considerando somente a instalação do ambiente Windows.

Instalando o Java Development Kit (JDK)

O JDK é um conjunto de ferramentas necessárias para desenvolver e executar programas Java.

1. **Baixar o JDK:**

   - Acesse o site oficial da Oracle: Oracle JDK Downloads.
   - Escolha a versão mais recente do JDK (por exemplo, JDK 17).
   - Selecione o sistema operacional Windows e baixe o instalador.

2. **Instalar o JDK:**

   - Execute o instalador baixado.
   - Siga as instruções de instalação, aceitando os termos de licença e escolhendo o local de instalação.

3. **Configurar a Variável de Ambiente JAVA_HOME:**

   - Abra o Painel de Controle e vá para **Sistema e Segurança > Sistema > Configurações avançadas do sistema**.
   - Clique em **Variáveis de Ambiente**.
   - Em **Variáveis do sistema**, clique em **Novo** e adicione JAVA_HOME como o nome da variável e o caminho de instalação do JDK como o valor (por exemplo, C:\Program Files\Java\jdk-17).
   - Adicione %JAVA_HOME%\bin ao final da variável de sistema Path.

4. **Verificar a Instalação:**

    - Abra o terminal ou prompt de comando.
    - Execute java -version e javac -version para verificar se o JDK está corretamente instalado.

Instalando o IntelliJ IDEA

IntelliJ IDEA é uma poderosa IDE para desenvolvimento em Java, oferecendo muitas funcionalidades que facilitam o desenvolvimento.

1. **Baixar o IntelliJ IDEA:**
    - Acesse o site oficial da JetBrains: IntelliJ IDEA Downloads.
    - Escolha a versão Community (gratuita) ou Ultimate (paga, com mais funcionalidades).
    - Baixe o instalador apropriado para o Windows
2. **Instalar o IntelliJ IDEA:**
    - Execute o instalador baixado.
    - Siga as instruções de instalação, aceitando os termos de licença e escolhendo o local de instalação.
3. **Configurar o IntelliJ IDEA:**
    - Abra o IntelliJ IDEA após a instalação.
    - No primeiro lançamento, você pode importar configurações anteriores ou iniciar com as configurações padrão.
    - Escolha um tema (Light ou Dark).
    - Instale plugins adicionais, se necessário.
4. **Criar um Novo Projeto Java:**
    - Na tela inicial do IntelliJ IDEA, clique em **New Project**.
    - Selecione **Java** no menu de seleção de linguagem.
    - Verifique se o SDK está configurado corretamente. Caso contrário, adicione o caminho para o JDK instalado anteriormente.
    - Clique em **Next** e siga as instruções para criar o seu projeto.
5. **Executar um Programa Java:**

    - Dentro do projeto criado, crie um arquivo Java:

        - Clique com o botão direito na pasta src e selecione New > Java Class.
        - Nomeie a classe como HelloWorld.

- Adicione o seguinte código ao arquivo HelloWorld.java:

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

- Execute o programa:
    - Clique com o botão direito no arquivo HelloWorld.java e selecione **Run 'HelloWorld.main()'**.
    - Observe o resultado no console do IntelliJ.

Padrões de Nomeação em Java

Pacotes

- **Regra**: Nomes de pacotes devem ser em minúsculas e usar letras, números e o caractere ponto (.) para separar níveis de pacotes.
- **Convenção**: Use o domínio da empresa ao contrário como prefixo, seguido pelo nome do projeto e outros subpacotes descritivos.
- **Exemplo**: com.exemplo.projeto.util

Classes

- **Regra**: Nomes de classes devem ser substantivos e usar PascalCase (também conhecido como UpperCamelCase), onde cada palavra começa com uma letra maiúscula.
- **Convenção**: Use nomes que descrevam claramente o propósito da classe.
- **Exemplo**: Cliente, ContaBancaria, GerenteDeProjeto

Interfaces

- **Regra**: Nomes de interfaces seguem a mesma convenção de classes, utilizando PascalCase.
- **Convenção**: Pode ser útil usar um adjetivo ou descrever uma capacidade ou contrato que a interface representa.
- **Exemplo**: Runnable, Serializable, DataProcessor

Métodos

- **Regra**: Nomes de métodos devem ser verbos ou frases verbais e usar camelCase, onde a primeira palavra começa com letra minúscula e cada palavra subsequente começa com letra maiúscula.
- **Convenção**: Escolha nomes que indiquem claramente a ação ou o comportamento que o método realiza.
- **Exemplo**: calcularSalario, enviarEmail, abrirConexao

Variáveis

- **Regra**: Nomes de variáveis devem ser substantivos ou frases nominais e usar camelCase.
- **Convenção**: Sejam descritivos e evitem abreviações, a menos que sejam bem conhecidas.
- **Exemplo**: salario, numeroDeFuncionarios, nomeDoCliente

Constantes

- **Regra**: Nomes de constantes devem ser em letras maiúsculas, com palavras separadas por underscores (_).
- **Convenção**: Use nomes que descrevam claramente o valor representado pela constante.
- **Exemplo**: MAX_VALUE, DEFAULT_TIMEOUT, PI

Operações Aritméticas e Operadores Aritméticos

Operações Aritméticas

As operações aritméticas básicas são aquelas envolvendo números, como adição, subtração, multiplicação e divisão. Elas são amplamente utilizadas em programação para realizar cálculos e manipular valores numéricos.

Operadores Aritméticos

Em programação, os operadores aritméticos são símbolos especiais que representam operações matemáticas. Os operadores aritméticos básicos são:

- **Adição (+):** Soma dois valores.
- **Subtração (-):** Subtrai um valor de outro.
- **Multiplicação (*):** Multiplica dois valores.
- **Divisão (/):** Divide um valor pelo outro.
- **Módulo (%):** Retorna o resto da divisão entre dois valores.

Variáveis e Tipos Primitivos

- **Variáveis:** São espaços na memória que armazenam valores. Em programação, as variáveis têm um nome, um tipo e um valor. Elas são utilizadas para armazenar resultados de operações, facilitando o reuso e a manipulação de dados.
- **Tipos Primitivos:** São os tipos de dados básicos suportados pela linguagem de programação. Alguns exemplos comuns são:
    - **int:** Números inteiros.
    - **double:** Números decimais (ponto flutuante).
    - **char:** Caracteres individuais.
    - **boolean:** Valores booleanos (verdadeiro ou falso).

Exemplos:

```
public class OperacoesAritmeticas {
  public static void main(String[] args) {
    int a = 10;
    int b = 5;
    int soma = a + b;
    int subtracao = a - b;
```

```
    int multiplicacao = a * b;
    int divisao = a / b;
    int modulo = a % b;
    double numero = 16;
    double raizQuadrada = Math.sqrt(numero);

    System.out.println("Soma: " + soma);
    System.out.println("Subtração: " + subtracao);
    System.out.println("Multiplicação: " + multiplicacao);
    System.out.println("Divisão: " + divisao);
    System.out.println("Módulo: " + modulo);
    System.out.println("A raiz quadrada de " + numero + " é " + raizQuadrada);
  }
}
```

Casting em Programação

Casting é o processo de converter um valor de um tipo de dados para outro. Isso é necessário quando você deseja atribuir um valor de um tipo de dados a uma variável de outro tipo de dados ou quando você precisa forçar a interpretação de um valor como um tipo diferente.

Tipos de Casting

1. **Casting Implícito**: Também conhecido como conversão automática, ocorre quando o compilador realiza a conversão de forma automática e segura. Isso geralmente acontece quando não há perda de dados, como de int para double.
2. **Casting Explícito:** Também conhecido como conversão manual, ocorre quando você precisa forçar a conversão de um tipo para outro, mesmo que haja perda de dados. Isso é feito colocando o tipo entre parênteses antes do valor a ser convertido.

Exemplos de Casting em Java

```
public static void main(String[] args) {
  // Conversão implícita de int para double
  int a = 10;
  double b = a;
  System.out.println(b);

  // Conversão explícita de double para int
```

```
    double c = 10.5;
    int d = (int) c;
    System.out.println(d);
}
```

Entrada de Dados em Java usando Scanner

Em Java, a classe Scanner é usada para obter entradas do usuário a partir do console ou de outros fluxos de entrada. Ela fornece métodos simples para ler diferentes tipos de dados, como inteiros, números de ponto flutuante, strings, entre outros.

Exemplo Básico de Uso do Scanner:

```
package entradaDadosScanner;

import java.util.Scanner;
public class DadosScanner {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite um número inteiro: ");
        int numeroInteiro = scanner.nextInt();
        System.out.println("Você digitou: " + numeroInteiro);

        System.out.print("Digite um número decimal: ");
        double numeroDecimal = scanner.nextDouble();
        System.out.println("Você digitou: " + numeroDecimal);

        System.out.print("Digite seu nome: ");
        String nome = scanner.next();
        System.out.println("Olá, " + nome + "!");

        scanner.close();
    }
}
```

Trabalhando com Arrays

Os arrays em Java são utilizados para armazenar múltiplos valores do mesmo tipo em uma única variável. Eles têm um tamanho fixo que é especificado na criação e podem ser percorridos para acessar ou modificar os seus elementos.

Declaração e inicialização de arrays

```java
package trabalhandoComArray;

public class exemploArrays {
   public static void main(String[] args) {

      // Declaração e inicialização de um array de inteiros
      int[] numeros = new int[5];

      // Inicializando valores no array
      numeros[0] = 10;
      numeros[1] = 20;
      numeros[2] = 30;
      numeros[3] = 40;
      numeros[4] = 50;

      // Outra forma de inicializar
      int[] outrosNumeros = { 1, 2, 3, 4, 5 };

      // Exibindo todos os elementos
      for (int i = 0; i < numeros.length; i++) {
         System.out.println("Elemento na posição " + i + ": " + numeros[i]);
      }

   }
}
```

Arrays Multidimensionais

```
package arraysMultidimensional;

public class ExemploArrayMulti {
   public static void main(String[] args) {

      // Declaração de um array 2D
      int[][] matriz = new int[2][3];

      // Inicialização de um array 2D
      matriz[0][0]=1;
      matriz[0][1]=2;
      matriz[0][2]=3;
      matriz[1][0]=4;
      matriz[1][1]=5;
      matriz[1][2]=6;

      // Acessando elementos
      for (int i = 0; i < matriz.length; i++) {
         for (int j = 0; j < matriz[i].length; j++) {
            System.out.println("Elemento na posição [" + i + "][" + j + "]: " + matriz[i][j]);
         }
      }
   }
}
```

Estruturas de Controle de Fluxo

As estruturas de controle de fluxo em Java são utilizadas para controlar a execução do código com base em condições ou repetições.

If-else e for

```
package controleFluxo;

public class ControleDeFluxo {
  public static void main(String[] args) {
    // if-else
    int idade = 20;

    if (idade >= 18) {
      System.out.println("Pode votar!");
    } else {
      System.out.println("Não pode votar!");
    }

    String[] frutas = {"Banana", "Maça", "Pera","Uva","Morango"};

    // for
    for(int x=0; x < frutas.length;x++){
      System.out.println(frutas[x]);
    }

  }
}
```

Switch Case

```
package controleFluxo;

public class Exemploswitch {
    public static void main(String[] args) {

        // Método Antigo até a versão 11
        int day = 3;
        String dayName;

        switch (day) {
            case 1:
                dayName = "Sunday";
                break;
            case 2:
                dayName = "Monday";
                break;
            case 3:
                dayName = "Tuesday";
                break;
            case 4:
                dayName = "Wednesday";
```

```
                break;
            case 5:
                dayName = "Thursday";
                break;
            case 6:
                dayName = "Friday";
                break;
            case 7:
                dayName = "Saturday";
                break;
            default:
                dayName = "Invalid day";
                break;
        }

        System.out.println("The day is: " + dayName);


        // Novo método a partir da versão 12

        int day2 = 1;
        String dayName2 = switch (day2) {
            case 1 -> "Sunday";
            case 2 -> "Monday";
            case 3 -> "Tuesday";
            case 4 -> "Wednesday";
            case 5 -> "Thursday";
            case 6 -> "Friday";
            case 7 -> "Saturday";
            default -> "Invalid day";
        };

        System.out.println("The day is: " + dayName2);
    }
}
```

Manipulação de Strings

Java possui uma classe String que é utilizada para manipulação de strings, como concatenação, busca de caracteres, entre outras operações.

```java
package manipulandoStrings;

public class ManipulandoStrings {
   public static void main(String[] args) {
      String nome = "João";
      String sobrenome = "Silva";
      String nomeCompleto = nome + " " + sobrenome;
      System.out.println("Nome completo: " + nomeCompleto);

      // Concatenando numeros
      int idade = 30;
      String mensagem = "Idade: " + idade;
      System.out.println(mensagem);

      //Camparando Strings
      String str1 = "hello";
      String str2 = "world";
      if (str1.equals(str2)) {
         System.out.println("As strings são iguais");
      } else {
         System.out.println("As strings são diferentes");
      }

      // Verificação de Substring
      String texto = "Olá, mundo!";
      if (texto.contains("mundo")) {
         System.out.println("A string contém a palavra 'mundo'");
      } else {
         System.out.println("A string não contém a palavra 'mundo'");
      }

      //Divisão de String em Substrings
      String frase = "Esta é uma frase de exemplo";
      String[] palavras = frase.split(" ");
      for (String palavra : palavras) {
         System.out.println(palavra);
      }

      //Remoção de Espaços em Branco
      String textoComEspacos = "   texto com espaços   ";
      String textoSemEspacos = textoComEspacos.trim();
```

```
        System.out.println("Texto com espaços: '" + textoComEspacos + "'");
        System.out.println("Texto sem espaços: '" + textoSemEspacos + "'");

        // Transforma palavra em minúsculo para maiúsculo
        String frase2 = "O Senhor é o meu pastor e nada me faltara.";

        System.out.println("Frase em maiusculo " + frase2.toUpperCase());
        System.out.println("Frase em minusculo " + frase2.toLowerCase());

    }
}
```

Tratamento de Exceções

O tratamento de exceções em Java é utilizado para lidar com situações excepcionais que podem ocorrer durante a execução do programa.

```
package tratamentoExcecoes;
public class TratamentoExcecoes {

    public static void main(String[] args) {
        try {
            int resultado = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Erro ao dividir por zero!");
        }
    }
}
```