JAVA

A linguagem de programação Java foi criada por uma equipe de engenheiros liderada por James Gosling na década de 1990, na empresa Sun Microsystems, que posteriormente foi adquirida pela Oracle Corporation. A história do Java remonta à necessidade de uma linguagem de programação que pudesse ser executada em diferentes plataformas e dispositivos, independentemente de sua arquitetura ou sistema operacional.

A equipe de desenvolvimento, conhecida como "Green Team" (Equipe Verde), tinha como objetivo inicial criar uma linguagem de programação para dispositivos eletrônicos domésticos, como TVs e aparelhos de fax. No entanto, suas ambições logo se expandiram para o desenvolvimento de uma plataforma de software abrangente que pudesse ser usada em uma ampla variedade de dispositivos.

A linguagem Java foi projetada com base em três princípios fundamentais: simplicidade, portabilidade e segurança. Gosling e sua equipe queriam criar uma linguagem que fosse fácil de aprender e usar, permitindo que os programadores escrevessem código de maneira eficiente. Além disso, eles buscaram garantir que o código escrito em Java pudesse ser executado em qualquer máquina virtual Java (JVM), tornando-o altamente portátil.

Um dos marcos importantes na história do Java foi o lançamento da primeira versão pública, chamada de Java 1.0, em 1996. A linguagem rapidamente ganhou popularidade entre os desenvolvedores devido à sua combinação única de recursos, incluindo o gerenciamento automático de memória, coleta de lixo e um modelo de programação orientado a objetos.

Outro fator importante para o sucesso do Java foi a introdução da plataforma Java SE (Java Standard Edition), que fornece uma base sólida para o desenvolvimento de aplicativos desktop e servidores. Posteriormente, a Sun Microsystems também lançou a plataforma Java EE (Java Enterprise Edition) para desenvolvimento de aplicativos corporativos e a plataforma Java ME (Java Micro Edition) para dispositivos com recursos limitados, como telefones celulares.

O Java continuou a evoluir ao longo dos anos, com o lançamento de novas versões que trouxeram recursos aprimorados, melhorias de desempenho e correções de segurança. Além disso, a comunidade Java cresceu significativamente, com uma vasta quantidade de bibliotecas, frameworks e ferramentas de desenvolvimento disponíveis para facilitar o trabalho dos programadores.

Atualmente, o Java é uma das linguagens de programação mais populares e amplamente usadas em todo o mundo. É usado em uma variedade de aplicativos, desde sistemas em larga escala, como servidores e sistemas de gerenciamento de banco de dados, até aplicativos móveis e embarcados.

Em resumo, a história da linguagem de programação Java é marcada pela visão de uma equipe de engenheiros em criar uma linguagem versátil e portátil. Com seus princípios de simplicidade, portabilidade e segurança, o Java conquistou uma posição sólida na indústria de desenvolvimento de software e continua sendo uma escolha popular entre os programadores até os dias de hoje.

JVM - JAVA VIRTUAL MACHINE

CÓDIGO = Linguagem JAVA

"EXECUTÁVEL" = Bytecode Java

JAVA MACHINE VIRTUAL

Linux Windows Mac

Executar o código independente do sistema operacional.

Java você terá sempre o mesmo "executável" ou Bytecode que será executado pela Máquina Virtual Java, totalmente independente do sistema operacional. Assim não é preciso rescrever ou adaptar o código para rodar em um outro sistema operacional. Temos um único executável para todos os sistemas!

CARACTERÍSTICAS DO JAVA:

Muitas Bibliotecas:

Java possui muitas bibliotecas que ainda muito aumentaram devido as milhares de contribuições da comunidade open source.

Multiplataforma:

Java é multiplataforma, roda em qualquer sistema operacional desde que exista a máquina virtual Java.

Orientada a Objetos

Parecido com C++

Grande Comunidade

SISTEMAS QUE PODEM CRIAR COM JAVA:

a) Sistemas para Web (Web Sites ou Web Apps)

b) Sistemas puramente "Server-Side"

c) Aplicativos para Android

d) Applets que rodam no navegador

web ou puramente server-side, aplicativo Android ou um antigo applet, Java atende todas essas áreas! No entanto, os sistemas web e Android são os que mais fizeram sucesso no mundo Java e mais empregam desenvolvedores.

QUAL A DIFERENÇA ENTRE O EXECUTÁVEL DO WINDOWS (exe) E O EXECUTÁVEL DO MUNDO JAVA (Bytecode)?

1 - Os executáveis do mundo Windows podem ser executados diretamente no sistema operacional, os do mundo Java precisam da máquina virtual.

2 - Os executáveis do mundo Java são portáveis, os do mundo Windows não: lembrando portável significa que podem ser executados em vários sistemas operacionais diferentes (Windows, Linux, Mac, etc).

BYTECODE:

O Bytecode é independente do sistema operacional:

O Bytecode é portável = significa que podem ser executados em vários sistemas operacionais diferentes (Windows, Linux, Mac, etc).

O Bytecode é parecido com o Assembly (código de montagem que é traduzido para código de máquina).

O Bytecode é executado (interpretado) pela JVM: o Bytecode é carregado e executado pela JVM.

o Bytecode foi criado e otimizado para a máquina virtual.

o Bytecode que é um código de máquina parecido com o Assembly. Talvez você (como eu!) estranhou o nome Bytecode, no entanto, tem uma explicação bem simples para tal. Existe um conjunto de comandos que a máquina virtual Java entende. Esses comandos também são chamados de opcodes (operation code), e cada opcode possui o tamanho de exatamente 1 Byte! E aí temos um opcode de 1 Byte ou, mais simples, Bytecode.

Instalando o primeiro programa

Baixar o executável do java para dev: JDK ORACLE

JRE = é o ambiente de execução do java.

JDK = são as ferramentas de desenvolvimento junto com o ambiente de execução.

JDK = JRE + ferramentas de desenvolvimento

JVM (Java Virtual Machine) = A responsabilidade da Java Virtual Machine é executar o Bytecode!

Então qual é diferença entre JVM e JRE? Ambos executam o Bytecode, certo?

O JRE (o nosso ambiente de execução) contém a JVM, mas também possui um monte de bibliotecas embutidas. Ou seja, para rodar uma aplicação Java não basta ter apenas a JVM, também é preciso ter as bibliotecas.

JRE = JVM + bibliotecas

Quais das afirmações abaixo são verdadeiras referente a JDK e JRE?

O JDK é o ambiente para executar uma aplicação Java e possui várias ferramentas de desenvolvimento = O JDK são as ferramentas de desenvolvimento (como o compilador) mas também tem JRE embutido!

O JRE é o ambiente para executar uma aplicação Java = Caso queira executar uma aplicação Java apenas, basta o JRE (Java Runtime Environment), mas para desenvolver em JAVA precisa do JDK

PROCESSO DE COMPILAÇÃO E EXECUÇÃO

*1 - * Durante a compilação acontece uma verificação sintática do código fonte.

*2 - * Na compilação e execução podem aparecer erros.

*3 - * A JVM lê e executa o Bytecode.

*4 - * O compilador gera o Bytecode caso não tenha nenhum erro sintático no código fonte.

INSTALANDO O IJ ou ECLIPSE

SOBRE IDES

Um IDE é um ambiente integrado de desenvolvimento que centraliza em um único lugar o compilador da linguagem utilizada, editor de texto, documentação e tudo que gira em torno da criação de aplicações.

NetBeans, ECLIPSE e Intellij são outros IDEs famosos do mundo Java.

ECLIPSE:

Um workspace é a pasta padrão que será utilizado para armazenar todos os projetos criados com a IDE Eclipse.

Cada projeto do Eclipse fica dentro de um workspace.

PROJETO JAVA

Dentro de um projeto Java, criamos uma nova classe através da opção do menu File -> New -> Class.

A saída do nosso programa executado pelo Eclipse é feita através da view console.

Executamos nosso programa no Eclipse através do menu Run -> Run as -> Java Application.

VIEW NAVIGATOR

É parecida com o Windows Explorer do Windows ou o Finder do MAC. Ela nos permite ver o diretório do projeto com seus arquivos na integra.

CAP 1 - INTRODUÇÃO A JAVA

Processamento de dados é a parte do backend. Roda em servidores, máquinas empresariais.

JAVA É USADA PARA:

1 - Desenvolvimento Web;

2 - Terminal do pc: transferência de arquivo, backup;

3 - Desenvolvimento Embarcados: software de câmeras, smartwatch, eletrodomésticos;

4 - Desenvolvimento de aplicações científicas: análises de imagens, geolocalização, qual câmera usada para processar imagem;

5 - Desenvolvimento Mobile;

CAP 2 – SETUP

Java uma linguagem compilada, mas precisa de um ambiente de desenvolvimento.

JRE – JAVA RUNTIME ENVIONMENT

Um ambienta de execução do JVM, como para fazer imposto de renda.

JDK – JAVA DEVELOPMENT KIT

Kit de desenvolvimento para aplicações em JAVA.

Para baixar:

1 - Java jdk download: java oracle.

2 – Java 18 a mais recente. Java 17: versão LTS, versão de longo termo, versão estendida, dá mais segurança, mais recomendável para o mercado.

3 – Abrir no terminal e instalar o path onde está o kit descompactado.

5 – O java pode ser desenvolvido no próprio terminal do computador.

IDES – INTEGRATED DEVELOPMENT ENVIRONMENT

Framework: IJ – melhor ferramenta atual e melhor do mercado.

CAP 3 - VARIÁVEIS EM JAVA

Site para tirar todas as dúvidas sobre o JAVA, tutorial oficial:

https://docs.oracle.com/javase/tutorial/

TIPOS PRIMITIVOS DE VARIÁVEIS

A linguagem de programação Java é tipada estaticamente, o que significa que todas as variáveis devem primeiro ser declaradas antes de serem usadas. Isso envolve informar o tipo e o nome da variável, como você já viu:

engrenagem int = 1;

O tipo de dados de uma variável determina os valores que ela pode conter, mais as operações que podem ser realizadas nela. Além disso int, a linguagem de programação Java oferece suporte a sete outros tipos de dados primitivos. Um tipo primitivo é predefinido pelo idioma e é nomeado por uma palavra-chave reservada. Valores primitivos não compartilham estado com outros valores primitivos.

Os oito tipos de dados primitivos suportados pela linguagem de programação Java são:

byte: O bytetipo de dados é um inteiro de complemento de dois com sinal de 8 bits. Tem um valor mínimo de -128 e um valor máximo de 127 (inclusive). O bytetipo de dados pode ser útil para economizar memória em grandes arrays , onde a economia de memória realmente importa. Eles também podem ser usados no lugar de int onde seus limites ajudam a esclarecer seu código; o fato de o alcance de uma variável ser limitado pode servir como uma forma de documentação.

short: O short tipo de dados é um inteiro de complemento de dois com sinal de 16 bits. Tem um valor mínimo de -32.768 e um valor máximo de 32.767 (inclusive). Assim como com byte, as mesmas diretrizes se aplicam: você pode usar a short para economizar memória em grandes arrays, em situações em que a economia de memória realmente importa.

int: Por padrão, o int tipo de dados é um inteiro de complemento de dois com sinal de 32 bits, que tem um valor mínimo de -231 e um valor máximo de 231 -1. No Java SE 8 e posterior, você pode usar o int tipo de dados para representar um inteiro não assinado de 32 bits, que tem um valor mínimo de 0 e um valor máximo de 2 32 -1. Use a classe Integer para usar int o tipo de dados como um inteiro sem sinal. Veja a seção As Classes de Números para mais informações. Métodos estáticos como compare Unsigned, divide Unsignedetc foram adicionados à Integer classe para suportar as operações aritméticas para inteiros sem sinal.

long: O long tipo de dados é um inteiro de complemento de dois de 64 bits. O longo assinado tem um valor mínimo de -2 63 e um valor máximo de 2 63 -1. No Java SE 8 e posterior, você pode usar o long tipo de dados para representar um comprimento de 64 bits sem sinal, que tem um valor mínimo de 0 e um valor máximo de 2 64 -1. Use esse tipo de dados quando precisar de um intervalo de valores mais amplo do que os fornecidos por int. A Long classe também contém métodos como compareUnsigned, divideUnsignedetc para dar suporte a operações aritméticas por tempo não assinado.

float: O float tipo de dados é um ponto flutuante IEEE 754 de 32 bits de precisão simples. Seu intervalo de valores está além do escopo desta discussão, mas é especificado na seção Tipos, formatos e valores de ponto flutuante da Especificação da linguagem Java. Assim como nas recomendações para byte e short, use a float(em vez de double) se precisar economizar memória em grandes matrizes de números de ponto flutuante. Este tipo de dados nunca deve ser usado para valores precisos, como moeda. Para isso, você precisará usar a classe java.math.BigDecimal . Capas de números e stringsBigDecimal e outras classes úteis fornecidas pela plataforma Java.

double: O double tipo de dados é um ponto flutuante IEEE 754 de 64 bits e precisão dupla. Seu intervalo de valores está além do escopo desta discussão, mas é especificado na seção Tipos, formatos e valores de ponto flutuante da Especificação da linguagem Java. Para valores decimais, esse tipo de dados geralmente é a escolha padrão. Conforme mencionado acima, esse tipo de dados nunca deve ser usado para valores precisos, como moeda.

boolean: O boolean tipo de dados tem apenas dois valores possíveis: true e false. Use esse tipo de dados para sinalizadores simples que rastreiam condições verdadeiro/falso. Esse tipo de dado representa um bit de informação, mas seu "tamanho" não é algo definido com precisão.

char: O char tipo de dados é um único caractere Unicode de 16 bits. Tem um valor mínimo de '\u0000'(ou 0) e um valor máximo de '\uffff'(ou 65.535 inclusive).

VARIAVÉIS DE CARACTERES:

STRING: Embora tecnicamente não seja um tipo de dado primitivo, String é uma classe em Java que representa uma sequência de caracteres. É amplamente utilizado para manipulação e armazenamento de texto. A declaração de uma variável String é feita assim:

String = “Lenita”;

CHAR: O tipo char é usado para armazenar um único caractere Unicode. Ele ocupa 2 bytes de memória e pode representar caracteres individuais, como letras, dígitos, símbolos e caracteres especiais. Você pode declarar uma variável char da seguinte forma:

char = “A”;
char 65; compila pq é A em forma numérica.

OPERADORES LÓGICOS

Operador "E" lógico (AND) (&&): Retorna true se todas as expressões envolvidas forem verdadeiras. Caso contrário, retorna false. Por exemplo: if (x > 0 && y < 10) { ... }

Operador "OU" lógico (OR) (||): Retorna true se pelo menos uma das expressões envolvidas for verdadeira. Caso contrário, retorna false. Por exemplo: if (x > 0 || y > 100) { ... }

Operador "NÃO" lógico (NOT) (!): Inverte o valor de uma expressão booleana. Se a expressão for verdadeira, retorna false; se for falsa, retorna true. Por exemplo: if (!(x > 0)) { ... }

Operador	Descrição
&&	Operador "E" lógico (AND)
||	Operador "OU" lógico (OR)
!	Operador "NÃO" lógico (NOT)

OPERADORES DE COMPARAÇÃO

Igual a (==): Retorna true se os valores comparados forem iguais.
Exemplo: if (x == y) { ... }

Diferente de (!=): Retorna true se os valores comparados forem diferentes.

Exemplo: if (x != y) { ... }

Maior que (>): Retorna true se o valor da esquerda for estritamente maior que o valor da direita.
Exemplo: if (x > y) { ... }

Menor que (<): Retorna true se o valor da esquerda for estritamente menor que o valor da direita.
Exemplo: if (x < y) { ... }

Maior ou igual a (>=): Retorna true se o valor da esquerda for maior ou igual ao valor da direita.
Exemplo: if (x >= y) { ... }

Menor ou igual a (<=): Retorna true se o valor da esquerda for menor ou igual ao valor da direita.
Exemplo: if (x <= y) { ... }

Operador	Descrição
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Esses operadores de comparação são frequentemente usados em estruturas condicionais e loops para tomar decisões ou controlar o fluxo do programa com base nas relações de comparação entre valores.

OPERADORES DE CÁLCULO

Adição (+): Realiza a adição de dois valores.
Exemplo: int resultado = a + b;

Subtração (-): Realiza a subtração de dois valores.
Exemplo: int resultado = a - b;

Multiplicação (*): Realiza a multiplicação de dois valores.
Exemplo: int resultado = a * b;

Divisão (/): Realiza a divisão de dois valores.
Exemplo: double resultado = a / b;

Módulo (%): Retorna o resto da divisão de dois valores.
Exemplo: int resto = a % b;

Incremento (++): Incrementa o valor de uma variável em 1.
Exemplo: a++;

Decremento (--): Decrementa o valor de uma variável em 1.
Exemplo: b--;

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (Resto da divisão)
++	Incremento
--	Decremento

Esses operadores de cálculo são usados em expressões matemáticas para realizar cálculos, atribuir valores a variáveis e manipular valores numéricos em geral.

CAP 3 – OPERADORES BOOLEANOS – TABELA VERDADE

Armazena valores lógicos - true e false.

Estudo da variável boolean

booelan chovendo = true;

OPERADORES LÓGICOS

Operador && (AND) = todas as condições precisam ser verdadeira

true && true = true

true && false = false

false && true = false

false && false = false

OPERADOR || (OU) = Apenas uma das sentenças precisa ser verdade para compilar

true || true = true

true || false = true

false || true = true

false || false = false

OPERADOR TERNÁRIO (?) = Usado para dois comportamentos possíveis, dá uma condição

Estrutura:

String mensagem = fimDeSemana ? "É fim de semana" : "Não é fim de semana";

3 condições:

1º fimDeSemana = termo avaliado ?

2º "É fim de semana" = termo caso seja true :

3º "Não é fimd de semana= termo caso for sale

Olha no projeto JavaAda no arquivo BTG ADA

CAP 4 – ESTRUTURAS CONDICIONAIS

Como controlar o cód. para ser executado ou não baseado em valores lógicos.

IF / Else = Quando a condição é simples. If roda o true e o else o false e quebra o laço.

IF – ELSE IF – ELSE = quando tem mais de uma condição no loop.

Switch = Combinação de condições. Usando case para cada condição e break para parada da condição e análise do código.

CAP 5 - MANIPULAÇÃO DE STRING E DATAS

Olha no IJ

Como fazer manipulação de string deixa maiúsculas, minúsculas, quantidade de caráteres.

Dia das semanas: dias, anos mudar para padrão br.

Hora: horas saber a hora atual, pega para fazer manipulação de condições

CAP 6 - LAÇOS NUMÉRICOS

Repetições contadas, repetir uma quantidade limitadas de vezes os trechos de códigos. Usados para um bloco de cód. que deseja repetir e a quantidade de repetições.

CAP 7 - VETORES

Conhecidos como Arrays. Em Java, fortemente tipada, não pode misturar tipos dentro do array. Se for inteiro TODOS tem que ser números inteiros, etc.

PARA LEMBRA:

1º Java é uma linguagem fortemente tipada: n pode misturar tipos na mesma estrutura. Array só de números, só de caracteres.

2º No array, o primeiro índex sempre é 0 e o último sempre é o tamanho do array – 1. Exemplo: tamanho do array é 10, o último é 10 – 9, logo o último index é 9.

CAP 9 - FUNÇÕES

Definir nossas próprias funções, usado para tamanho de letras, scanner para receber entrada de dados. Do básico ao mais científico, System.out.println é uma função.

Sintaxe de como definir o nome dela pode definir ou não parâmetros e retorna valores sempre usando com return. PARAMETROS SEMPRE VÃO ESTÁ EM PARANTESES ()