POO - Java

Unidade 1 – Parte 1

HISTÓRICO E CONCEITOS DA LINGUAGEM

Java originou-se como parte de um projeto de pesquisa que visava a criação de um software avançado que atendesse a uma extensa variedade de maquinário de redes e sistemas embutidos. O objetivo inicial era desenvolver um ambiente operacional pequeno, confiável, portável, distribuído e que operasse em tempo real. Inicialmente, a linguagem escolhida foi C++. Porém, a com o passar do tempo, as dificuldades encontradas com C++ aumentaram até o ponto em que os problemas poderiam ser melhores endereçados se fosse criada uma linguagem completamente nova. As decisões de arquitetura e desenho da linguagem foram inspiradas em linguagens como Eiffel, SmallTalk, Objective C, e Cedar /Mesa. A sintaxe tem suas raízes claramente definidas em C e C++.

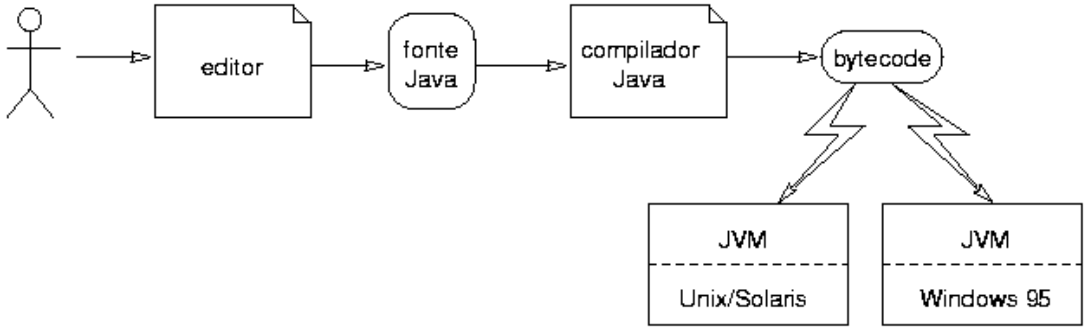
Deste modo, Java foi projetada para atender a vários requisitos desejáveis em uma linguagem de programação, como por exemplo, confiabilidade, devido ao seu gerenciamento de memória, o que resulta em um ganho de eficiência; redigibilidade, por eliminar alguns conceitos de C e C++ que dificultavam nesse sentido; reuso de código.

PRINCÍPIOS DA LINGUAGEM

MECANISMO

Uma das principais funcionalidades da linguagem Java é a portabilidade de código, ou seja, o código é escrito apenas uma vez e roda em diversas plataformas. Isto pode ser alcançado pela utilização de *bytecodes*. Bytecode é um formato de código intermediário entre o código fonte, o texto que o programador consegue manipular, e o código de máquina, que o

computador consegue executar. A máquina Virtual Java é quem interpreta esse bytecode em um sistema Java. Na figura abaixo se pode visualizar esse mecanismo.



CONFIGURAÇÃO DO SISTEMA

Para o desenvolvimento de programas Java é necessário configurar o seu ambiente. Abaixo se descreve os passos para a configuração:

– Download do pacote de desenvolvimento Java no site (https://www.java.com/pt-BR/). Procure por J2SE e faça o download. Há versões para diversos sistemas operacionais, preste atenção para baixar para o sistema operacional correto.

– Instalação do sistema na máquina local. Se o sistema operacional for um sistema Windows é simplesmente executar o arquivo EXE baixado no site.

- Verificar a configuração das variáveis de ambiente (PATH e CLASSPATH) do sistema. Para chamar os executáveis do ambiente Java (javac/Java...) é necessário que o diretório (/bin) esteja setado no PATH do sistema (ou que o executável seja chamado diretamente de dentro do diretório). Para que as classes desenvolvidas sejam “encontradas” pelo programa java.exe é necessário que o diretório das classes esteja setado no CLASSPATH do sistema ou que o programa seja chamado no mesmo dirretório em que tua classe esteja inserido..

Ex:

PATH=”c:\j2sdk1.4.1_03\bin\”

CLASSPATH=”c:\Java\MeusProgramasJava\”

Para instalar e colocar pra rodar código java no **Linux Mint 20**, segue tutorial: (https://netovieiraleo.medium.com/colocando-java-para-rodar-no-linux-mint-20-c98a6e335e87).

PRIMEIRO PROGRAMA

Seguindo as clássicas aulas de programação, segue abaixo o tradicional programas HelloWorld transportado para o ambiente Java.

```
class HelloWorld
{
	public static void main(String args[])
	{
		System.out.println("HelloWorld.");
	}
}
```

O nome da classe deve ter o mesmo nome do arquivo, ou seja, este código-fonte precisa ser salvo como HelloWorld.java

Para compilar o programa execute o comando abaixo:

javac HelloWorld.java

Este comando irá compilar seu código-fonte e gerar uma classe com o nome HelloWorld.class que é a classes compilada.

Para executar essa classe execute a seguinte linha de comando:

java HelloWorld

Esta linha de comando irá executar a classe e a saída do programa irá sair na saída padrão do sistema.

CONSTRUÇÃO DE UM PROGRAMA

Os programas Java são gerados pela *interpretação*, realizada pela máquina virtual JAVA, do código fonte escrito. Para ser gerado uma aplicação (diferente de uma classe) em JAVA é necessário a criação de um método main. Este método determina o início da execução do programa.

Abaixo se demonstra um exemplo básico de uma aplicação:

```
class Teste1
{
        public static void main(String args[])
        {
                System.out.println("Olá mundo.");
               System.out.println("Fantástico.");
        }
}
```

COMENTÁRIOS

Linha: //
Bloco: /* */

TIPOS PRIMITIVOS

Os tipos primitivos podem ser divididos nas seguintes classes:

Inteiros

Valores numéricos inteiros em Java podem ser representados por variáveis do tipo byte, short, int ou long. Todos os tipos contêm valores inteiros com sinal, com representação

interna em complemento de dois. O valor default para atributos desses tipos é 0. Cada um desses tipos de dados tem seu espaço de armazenamento definido na especificação da linguagem, não sendo dependente de diferentes implementações. Variáveis do tipo byte ocupam 8 bits de armazenamento interno. Com esse número de bits, é possível representar valores na faixa de -128 a +127. Variáveis do tipo short ocupam 16 bits, podendo assumir valores na faixa de -32.768 a +32.767. Variáveis do tipo int ocupam 32 bits, podendo assumir valores na faixa de -2.147.483.648 a +2.147.483.647. Finalmente, variáveis do tipo long ocupam 64 bits, podendo assumir valores na faixa de -9.223.372.036.854.775.808 a +9.223.372.036.854.775.807.

- byte (8 bits) (Vl Max: +127)
- short (16 bits) (Vl Max: +32.767)
- int (32 bits) (Vl Max: +2.147.483.647)
- long (64 bits). (Vl Max: +9.223.372.036.854.775.807)

Ponto Flutuante

Valores reais, com representação em ponto flutuante, podem ser representados por variáveis de tipo float ou double. Em qualquer situação, a representação interna desses valores segue o padrão de representação IEEE 754, sendo 0.0 o valor default para tais atributos.

- float (32 bits) (Vl Max: 1.44 x 106)
- double (64 bits) (Vl Max: 3.4254 x 10-2) (15 dígitos significativos)

Booleanos

Os tipos **booleanos** em Java não podem ser convertidos para nenhum tipo numérico. Valores booleanos em Java assumem apenas true ou false.

Char declarado em Java define um caracter de 16 bits, seguindo o padrão Unicode.

Caracteres Especiais:

- \n (Pula linha)
- \r (Retorno de carro)
- \b (Backspace)
- \t (Tabulação)
- \f (Nova página)

- \’ (Apóstrofe)
- \” (Aspas)
- \\ (Barra invertida)
- \u223d (Caracter UNICODE 233d)
- \g37 (Octal)
- \fca (Hexadecimal)

Todos os tipos numéricos são inicializados para 0, o tipo caracter é inicializado para o Unicode 0 e variáveis do tipo booleano são inicializados para false.

IDENTIFICADORES

Identificadores são seqüências de caracteres Unicode, que devem obedecer às seguintes regras:

- Um nome pode ser composto por letras, por dígitos e pelos símbolos _ e $.
- Um nome não pode ser iniciado por um dígito (0 a 9).
- Letras maiúsculas são diferenciadas de letras minúsculas.
- Uma palavra reservada da linguagem Java não pode ser um identificador.

Java diferencia as letras minúsculas da maiúsculas. Assim, as duas variáveis

```
int socorro;
int Socorro;
```

são variáveis diferentes.

OPERADORES

Regras são necessárias para validar expressões. Os operadores, juntos com variáveis associadas, permitem essa validação.

Incremento/Decremento:

Operadores	**Descrição**
Incremento	x++; ++x;
Decremento	x--; --x

Comparativos:

Operadores	Descrição	Exemplos
==	Igualdade	if (C == A)
!=	Diferença	if (C != A)
!	Negação	A = !0
+ - * /	Aritméticos	A = B + C
&&	e	If (A && B)
||	Ou	If (A || B)
%	Resto da divisão	A = B % C
>	Maior que	If (A < B)
<	Menor que	If (A > B)
>=	Maior ou igual	If (A >= B)
<=	Menor ou igual	If (A <= B)

CONTROLE DE FLUXO

IF

A forma mais básica de estrutura de controle condicional é o *if*, no qual se é escolhido um entre dois blocos de código a partir de uma condição

Ex:

if *(Expressão1 [&& Expressão2...])* *{* *bloco_de_comandos* *}* ***else*** *{* *bloco_de_comandos* *}*	*If (2 > 1)* *{* *a = b;* *}* *else* *{* *sResult = "A matemática pirou";* *}*

SWITCH

A estrutura switch proporciona a possibilidade de transferir a execução para um bloco de comandos (entre muitos) a partir do valor de uma expressão ou variável.

Ex:

switch *(Expressão) {* ***case*** *ExpressãoConstante : {* *bloco_de_comandos* *break;* *}* ***case*** *ExpressãoConstante : bloco_de_comandos* ***default*** *: bloco_de_comandos* *}*	***switch*** *(Valor) {* ***case*** *1: {* *a = b;* *break;* *}* ***case*** *2 : break;* ***default*** *: b = c;* *}*

WHILE

A estrutura *while* é um loop que é executado até que determinada expressão seja verdadeira.

Ex:

while *(Expressão1 [&& Expressão 2...])* *{* *bloco_de_comandos* *}*	***while*** *(a > b)* *{* *a = a – 1;;* *}*

DO-WHILE

Similar ao *while*, sendo que esta estrutura sempre executará o bloco de comandos uma vez e o teste será executado no final.

Ex:

Do *{* *bloco_de_comandos* *}* ***while*** *(Expressão);*	***do*** *{* *a = a + 1;* *}* ***while*** *(a < b);*

FOR

A estrutura *for* é um loop que será executada durante um intervalo definido (início e fim).

Ex:

for *(ExpressãoInicialização; Expressão; ExpressãoAvanço)* *{* *bloco de comandos* *}*	***for*** *(int a=0; a < 2; a++)* *{* *System.out.println("Valor = " + a);* *}*

O **break** transfere o controle para o final da iteração no qual está inserido. O **continue** não finaliza a execução da estrutura de controle, apenas força o término da interação atual, voltando para o teste de controle.

CONVERSÕES

Assim como a maioria das linguagens, o Java testa a compatibilidade dos tipos de dados, seja em comparações ou atribuições. A sintaxe é a vista abaixo:

(< type >) < expression >

Ex:

variavelMeuTipo = (< cMeuTipo >) param;

Valores booleanos não podem ser convertidos para outro tipo de dados e vice-versa. O mesmo se aplica a referencias a objetos nulos. Tipos de conversão utilizando casts são chamados de conversões explícitas. Em certos contextos, o compilador realiza conversões implícitas, por exemplo, quando um char é atribuído a um int.

OPERADORES TERNÁRIOS

Podemos escrever uma operação condicional sem a utilização do comando if/else com os operadores ternários.

```
a = x ? b : c;

class Ternario {
  public static void main(String args[]) {
    short v1 = 10;
    short v2 = 5;
    short menor;
    menor = (v1 < v2)? v1 : v2;
    System.out.println(menor);
  }
}
```

Maiores informações sobre a linguagem de programação Java podem ser acessadas no site (https://www.java.com/pt-BR/).