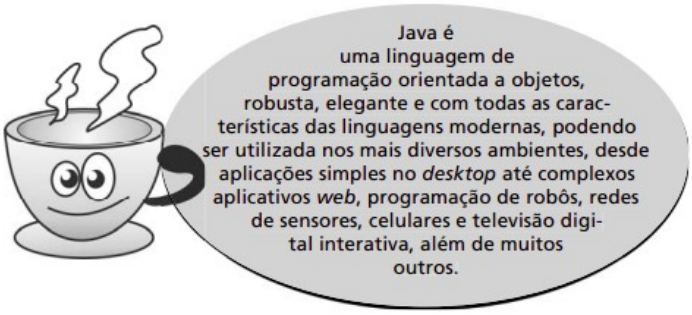


O que a difere de outras linguagem é que quando de compila um código java, não é gerado um arquivo executável (um arquivo .exe). É gerado um arquivo .class, chamado bytecode e para rodar esse arquivo bytecode, precisamos de uma JVM, o que torna o código extremamente portátil.

JDK: Java Development Kit - download do JDK do Java SE (Standard Edition) - formado pela JRE somado às ferramentas como o compilador.

O JDK podem ser baixado do site http://www.oracle.com/technetwork/java/. Para encontrá-los, acesse o link Java SE dentro dos top downloads. Consulte o apêndice de instalação do JDK para mais informações.

Java é uma das linguagens mais utilizadas da atualidade. Criada em 1991 pela Sun MicroSystem e adquirida pela oracle em 2008 é responsável pela propagação da Programação Orientada a objeto.
Características:

- muito verbosa – melhorou com a utilização do frameworks
- o código é compilado para byteCode que será interpretado pela JVM – java vitual machine
- poder ser interpretado por todos os sistemas operacionais
- mantém compatibilidade com as versões anteriores
- divisões do java - XXXXXXXXX
- java SE development kit - XXXXXXXXX
- caseSensitive
- fortemente tipada

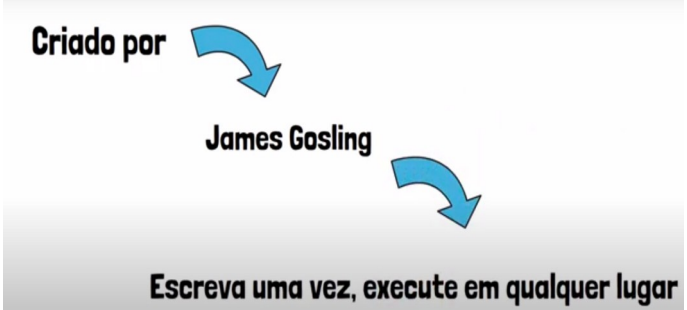
Os procedimentos para executar seu programa são muito simples. O javac é o compilador Java, e o Java é o responsável por invocar a máquina virtual para interpretar o seu programa.

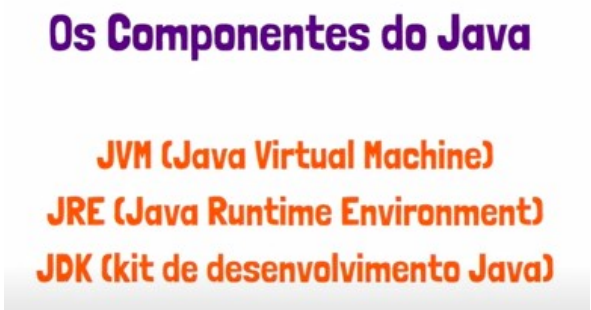
Após digitar o código do seu programa, grave-o em algum diretório. A fim de compilar, você deve pedir para que o compilador de Java da Oracle, chamado javac, gere o bytecode correspondente ao seu código Java.

Depois de compilar, o **bytecode** foi gerado. Um arquivo **.class** é gerado com o mesmo nome da sua classe Java.

Como é o bytecode?

O arquivo .class gerado não é entendido pelo programador. Está escrito no formato que a Virtual Machine compreende. É como um assembly escrito para uma máquina em específico. É o código de máquina da máquina virtual.





CARACTERÍSTICAS DA LINGUAGEM JAVA

Orientada a Objetos:

Java é uma linguagem voltada para a programação orientada a objetos e, por isso, todo o código está contido dentro de classes. java suporta herança simples, mas não herança múltipla. a ausência de herança múltipla pode ser compensada pelo uso de herança e interfaces, onde uma classe herda o comportamento de sua superclasse além de oferecer uma implementação para uma ou mais interfaces.

Compilada e Interpretada:

Todo programa Java é compilado e interpretado. Um programa em Java é compilado para um código composto por instruções chamadas de “bytecode”. O “bytecode” é um código de uma máquina virtual, chamada Máquina Virtual Java (Java Virtual Machine - JVM), idealizada pelos criadores da linguagem. Os bytecodes são independentes de plataforma e são interpretados pela JVM para serem executados no computador.

O PRIMEIRO PROGRAMA JAVA

Primeiro vamos escrever, compilar e executar o tradicional aplicativo “Hello World”. O código deste programa será explicado posteriormente para que você possa começar a entender os fundamentos da programação Java.

```
1. /* Meu programa Java */
2. public class HelloWorld
3. {
4.    public static void main (String args[ ])
5.    {
6.       System.out.println (“Hello, World!”);
7.    } // da rotina main
8. } // da class
```

EXPLICANDO

Linha 2: public class HelloWorld

Esta linha utiliza a palavra reservada class para declarar que uma nova classe será definida aqui. HelloWorld é o nome usado para identificar a classe. Toda a definição da classe, inclusive todo o código e os dados, estará entre a chave de abertura “{“ e a chave final “}” que se encontram nas linhas 5 e 8 deste exemplo.

Linha 4: public static void main (String args[ ])

A linha 4 contém a declaração do método main. O método main é simplesmente um ponto de partida para o interpretador Java. É por onde será iniciada a execução. O método main deverá sempre ser declarado na forma acima.

Linha 6: System.out.println (“Hello, World!”); Esta linha executa o método println do objeto out. Este objeto é uma instância da classe OutputStream e foi declarado como variável de classe (static) na classe System. Este método imprime na tela uma mensagem texto, no caso, “Hello, World!”. Por causa do modelo de objeto, uma saída simples de console é complicada para entender.

Que nome dar ao arquivo?

Após digitar o código-fonte, você deve nomear o arquivo. Os arquivos que contém o código-fonte Java devem sempre ter a terminação “.java”. Geralmente, em Java, coloca-se uma classe dentro de cada arquivo. O arquivo conterá o mesmo nome da classe.

Cuidado, o compilador Java diferencia letras maiúsculas de minúsculas, por isso, preste atenção quando for nomear o arquivo.

Se você utilizar o modificador public para a classe, por exemplo, public class HelloWorld, o arquivo deve possuir o mesmo nome que a classe. Caso não utilize public, o arquivo pode ter outro nome.

Compilando o código

Para compilar o código acima você deve digitar:

C:\> javac HelloWorld.java

O compilador é chamado pelo comando javac seguido do nome do arquivo sempre com a terminação “.java”. Ao ser compilado, se o código não possuir nenhum erro, será gerado um arquivo chamado HelloWorld.class composto por bytecodes.

Esse programa é independente de plataforma e, por isso, o seu programa HelloWorld (.class) pode ser executado em qualquer sistema operacional.]

Executando o programa HelloWorld

Após a compilação do programa, em que foi gerado o arquivo HelloWorld.class, você pode executá-lo digitando o seguinte comando:

C:\>java HelloWorld

No código acima estamos chamando o interpretador Java (java.exe) para carregar a classe HelloWorld e executá-la. No comando de execução não se deve digitar a extensão do arquivo (.class).

Declaração Import

Para utilizar os pacotes Java, usa-se a declaração de importação que define onde o compilador pode encontrar as classes destes pacotes. A declaração de importação (import) deve preceder a declaração de todas as classes. O compilador irá procurar por pacotes dentro dos diretórios especificados na variável de ambiente classpath.

VARIÁVEIS

As variáveis são posições na memória do computador que podem armazenar dados. As variáveis são formadas por quatro elementos: **nome, tipo, tamanho e valor**.

Dependendo da linguagem de programação, uma declaração de variável pode ter somente um tipo, um nome e um valor.

Segundo a convenção para identificadores Java, os métodos e variáveis devem ser nomeados com letras minúsculas. No caso do identificador ser formado por mais de um termo, o segundo termo e os termos seguintes devem iniciar com letra maiúscula. As variáveis são compostas por substantivos e adjetivos, enquanto que os nomes de métodos começam sempre com um verbo.

Exemplos: hora, horaDoDia, valorCorrente, obterHoraDoDia().

O java possui dois tipos de dados:

a) **por valor (tipos primitivos)**

b) **por referência (tipos por referência)**

Os tipos primitivos são **boolean, byte, char, short, int, long, float e double**. Os tipos por referência, são classes que especificam os tipos de objeto **Strings, Arrays Primitivos e Objetos**.

Uma variável do tipo primitivo pode armazenar exatamente um valor de seu tipo declarado por vez, quando outro valor for atribuído a essa variável, seu valor inicial será substituído.

As variáveis de tipo primitivo: byte, char, short, int, long, float e double são inicializadas como 0, e as variáveis do tipo boolean são inicializadas como **false**.

TIPOS POR REFERÊNCIA

Os programas utilizam as variáveis de tipos por referência para referenciar as localizações de objetos na memória do computador. Esses objetos podem conter várias variáveis e métodos dentro do objeto apontado. As variáveis de referência são inicializadas com o valor "**null**" (nulo).

Por exemplo,

ClasseCarro = new ClasseCarro()

O código acima cria um objeto de classe ClasseCarro e poderá invocar todos os seus métodos e atributos da classe. A palavra chave new solicita a memória do sistema para armazenar um objeto e inicializa o objeto.

VARIÁVEIS PRIMITIVAS

Pode-se declarar variáveis e usá-las. Em Java, toda variável deve ter um tipo que não pode ser modificada uma vez declarada:

Por exemplo: **int idade**

Comentários em Java

Com o objetivo de fazer um comentário em Java, você pode usar o
// para comentar até o final da linha ou,
/* */ para comentar o que estiver entre eles.
/* comentário daqui
até aqui */.

Além de atribuir, você pode utilizar esse valor.

```
// declara a idade.
int idade;
idade = 15;

// imprime a idade.
System.out.println(idade);
```

Como rodar esses códigos?

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

        // imprime a idade.
        int idade = 20;
        System.out.println(idade);
        }
}
```

No mesmo momento em que você declara uma variável, também é possível inicializá-la por praticidade:

```
int idade = 15;
```

Você pode usar os operadores +, -, / e * para operar com números, sendo eles responsáveis pela adição, subtração, divisão e multiplicação, respectivamente. Além desses operadores básicos, há o operador % (módulo), que é o **resto de uma divisão inteira**. Veja alguns exemplos:

```
int num1 = 2 + 2;
int num2 = 5 - 2;

int num3 = 4 * 2;
int num4 = 64 / 4;

int num5 = 5 % 2; // 5 dividido por 2 dá 2, e tem resto 1;
                  // o operador % pega o resto da divisão inteira.
```

Tipos Inteiros:

Tipo	Tamanho	Alcance
byte	8 bits	-128 até 127
short	16 bits	-32.768 até 32.767
int	32 bits	-2.147.483.648 até 2.147.483.647
long	64 bits	-9223372036854775808 até 9223372036854775807

Existem dois tipos de **números de ponto-flutuante**:
float (32 bits, precisão simples) e
double (64 bits, precisão dupla).

Representar números inteiros é fácil, mas como guardar valores reais, tais como frações de números inteiros e outros? Outro tipo de variável muito utilizado é o **double**, que armazena um número com ponto flutuante (e que também pode armazenar um número inteiro).

O Java fornece dois tipos primitivos para armazenar números de ponto flutuante na memória, o tipo float e double.

A diferença entre eles é que as variáveis double podem armazenar números com maior magnitude e mais detalhes, ou seja, armazena mais dígitos à direita do ponto de fração decimal, do que as variáveis float. As variáveis do tipo float representam números de ponto flutuante de precisão simples e podem representar até 7 dígitos.

As variáveis do tipo double representam números de ponto flutuante de precisão dupla, onde precisam duas vezes a quantidade de memória das variáveis float fornecendo 15 dígitos, sendo o dobro da precisão de variáveis float.

```
double pi = 3.14

double x = 5 * 10
```

O tipo **boolean** armazena um valor verdadeiro ou falso: nada de números, palavras ou endereços como em algumas outras linguagens.
boolean verdade = **true**; As palavras true e false são reservadas ao Java.

O tipo declarado como **char** é sempre declarado com aspas simples porque o tamanho é somente de 1 caractere.

```
char letra = 'a';
System.out.println(letra);
```

Os tipos float sempre irão possuir o caractere “f” no final do valor para sua identificação, sendo a mesma coisa com o tipo long só que é inserido o caractere “L”.

```
int i = 5;        // i recebe o valor 5;
int j = i;        // j recebe o valor de i;
i = i + 1;        // i vira 6, j continua 5.
```

Exercício: Variáveis e tipos primitivos

Em uma empresa é guardado o valor de vendas de cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total. Sabendo que:
janeiro: 15 mil reais;
fevereiro: 23 mil reais;
março: 17 mil reais.

Faça um programa que calcule e imprima a despesa total no trimestre e a média mensal de gastos.

Às vezes, precisamos que um número quebrado seja arredondado e armazenado em um número inteiro. Para fazer isso sem que haja o erro de compilação, é preciso ordenar que o número quebrado seja **moldado (casted)** como um número inteiro. Esse processo recebe o nome de **casting**.

double d3 = 3.14;
int i = (**int**) d3;
O casting foi feito para moldar a variável d3 como um int. O valor de i agora é 3.

O mesmo caso ocorre entre valores int e long.
long x = 10000;
int i = x; // não compila, pois pode estar perdendo informação.
E se quisermos realmente fazer isso, fazemos o casting:
long x = 10000;
int i = (**int**) x;

Casos não tão comuns de casting e atribuição

float x = 0.0;
O código acima não compila, pois todos os literais com ponto flutuante são considerados double pelo Java. E float não pode receber um double sem a perda de informação. Para fazê-lo funcionar, podemos escrever:
float x = 0.0f;
A letra f, que pode ser maiúscula ou minúscula, indica que aquele literal deve ser tratado como float.
Outro caso que é mais comum:
double d = 5;
float f = 3;
float x = f + (**float**) d;

Até casting com variáveis do tipo char podem ocorrer. O único tipo primitivo que não pode ser atribuído a nenhum outro tipo é o boolean.

PARA: / DE:	byte	short	char	int	long	float	double
byte	----	*Impl.*	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
short	(byte)	----	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
char	(byte)	(short)	----	*Impl.*	*Impl.*	*Impl.*	*Impl.*
int	(byte)	(short)	(char)	----	*Impl.*	*Impl.*	*Impl.*
long	(byte)	(short)	(char)	(int)	----	*Impl.*	*Impl.*
float	(byte)	(short)	(char)	(int)	(long)	----	*Impl.*
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

```
class ArithmeticTest {
public static void main ( Strings args[] ) {
      short x = 6;
      int y = 4;
      float a = 12.5f;
      float b = 7f;

      System.out.println ( "x é " + x + ", y é " + y );
      System.out.println ( "x + y = " + (x + y) );
      System.out.println ( "x - y = " + (x - y) );
      System.out.println ( "x / y = " + (x / y) );
      System.out.println ( "x % y = " + ( x % y ) );
```

Mais sobre atribuições

Variáveis podem atribuídas em forma de expressões como:

int x, y, z; x = y = z = 0;

No exemplo as três variáveis recebem o valor 0;

Operadores de Atribuição:	Expressão Significado
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y

Resumindo:

Grupo	Tipo	Tamanho	Intervalo de Valores	Valor Default
Inteiros	int	4 bytes	-2.147.483.648 até 2.147.483.647	0
	short	2 bytes	-32.768 até 32.767	0
	long	8 bytes	-9.223.372.036.854.775.808L até 9.223.372.036.854.775.807L	0L
	byte	1 byte	-128 até 127	0
Ponto Flutuante	float	4 bytes	+- 3,40282347E+38F (6-7 dígitos significativos)	0.0f
	double	8 bytes	+- 1,79769313486231570E+308 (15 dígitos significativos)	0.0d
	char	2 bytes	representa um Unicode	'\u0000';
	boolean	1 bit	true ou false	false