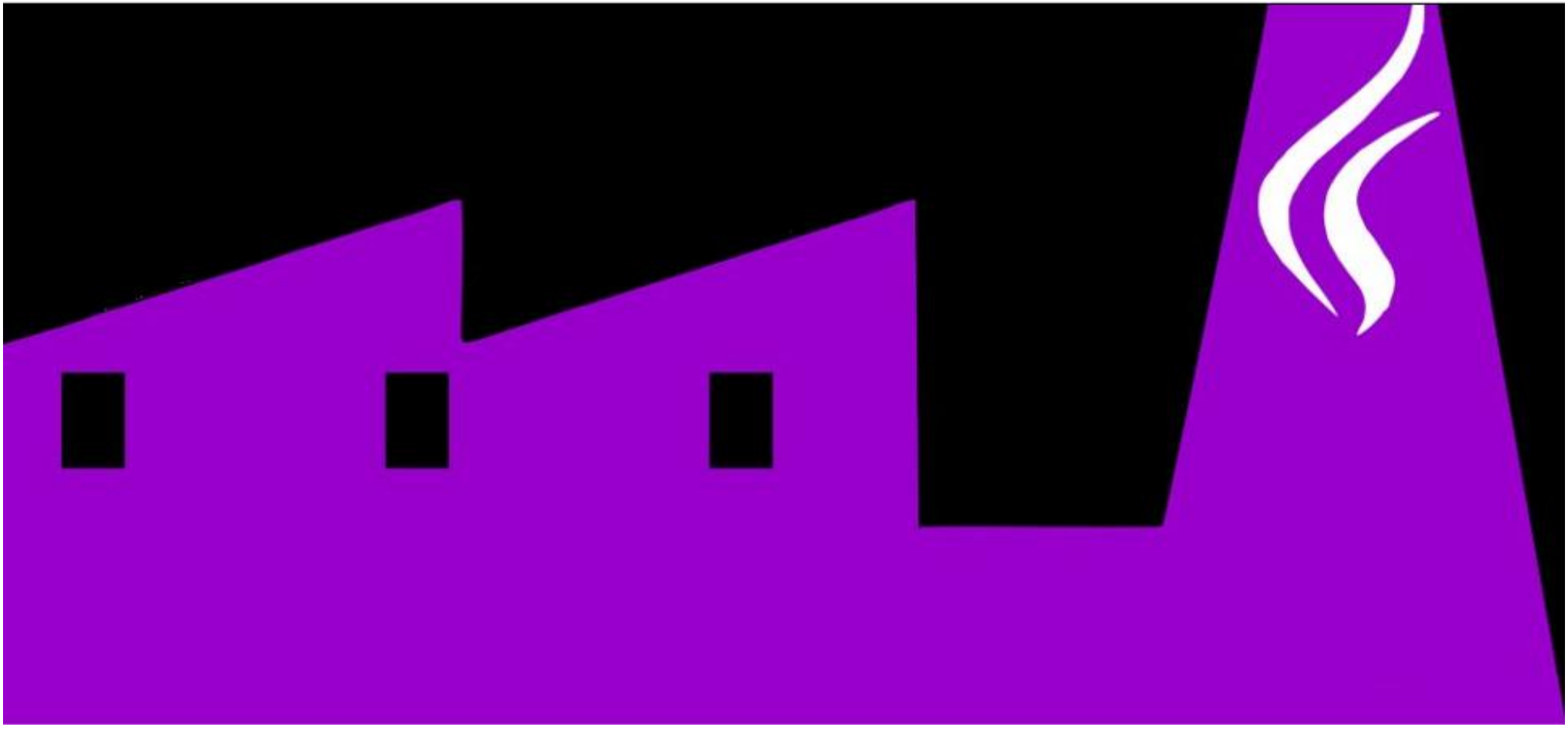
Fábrica de Software

Prof^es. Ivan L. Süptitz e Evandro Franzen

Introdução à Disciplina e à Linguagem Java



Visão Geral

	Segundas	Quintas
Linguagem	Python	Java
IDE	Colab	NetBeans
Paradigma	Estruturado	Orientado a Objetos

O que é Java?

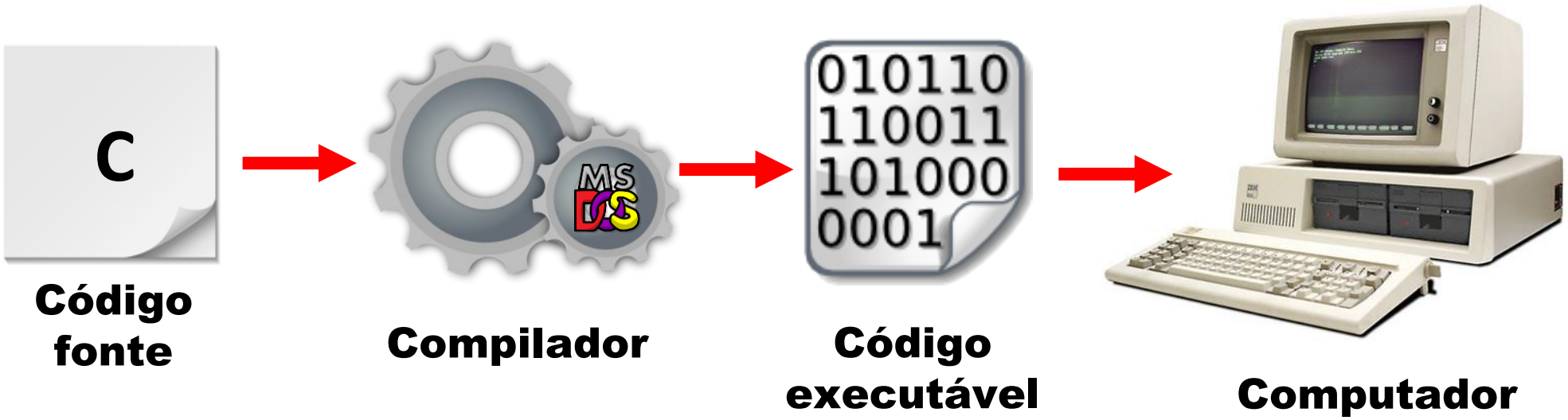
- Linguagem de programação Orientada a Objetos
- desenvolvida e mantida pela Sun (comprada pela Oracle em 2009)
- seu site principal é o http://java.sun.com
- Vamos entender agora como esta linguagem surgiu e porque...

O que o computador entende?

Código fonte

Computador

O que o computador entende?



? Qual o problema?

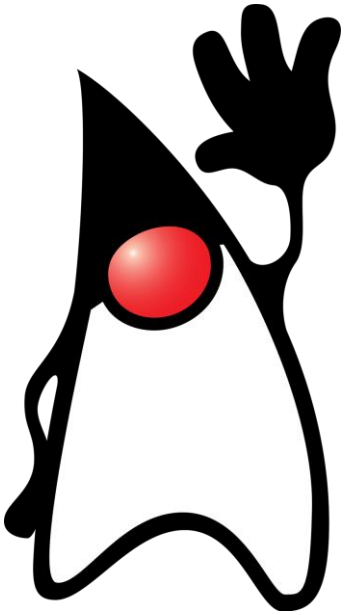
1990 – Sun Microsystems

James Gosling

Novo Projeto:
Dispositivos diferentes pudessem se comunicar entre si.

Oak ← 1991 ← GreenTalk

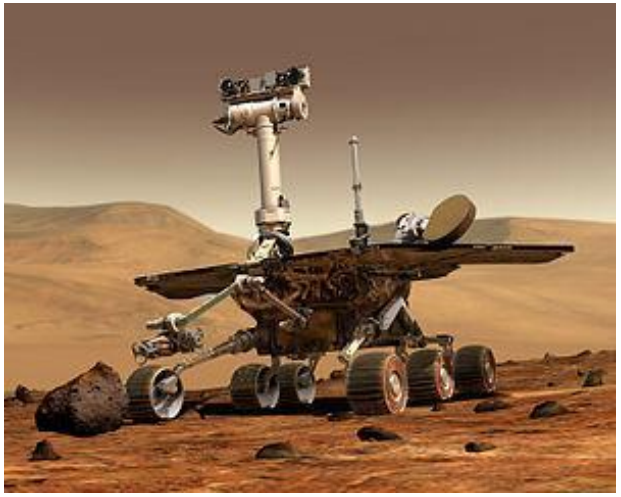
Star Seven - *7



https://www.youtube.com/watch?v=Ahg8OBYixL0

Nome da Linguagem: Java

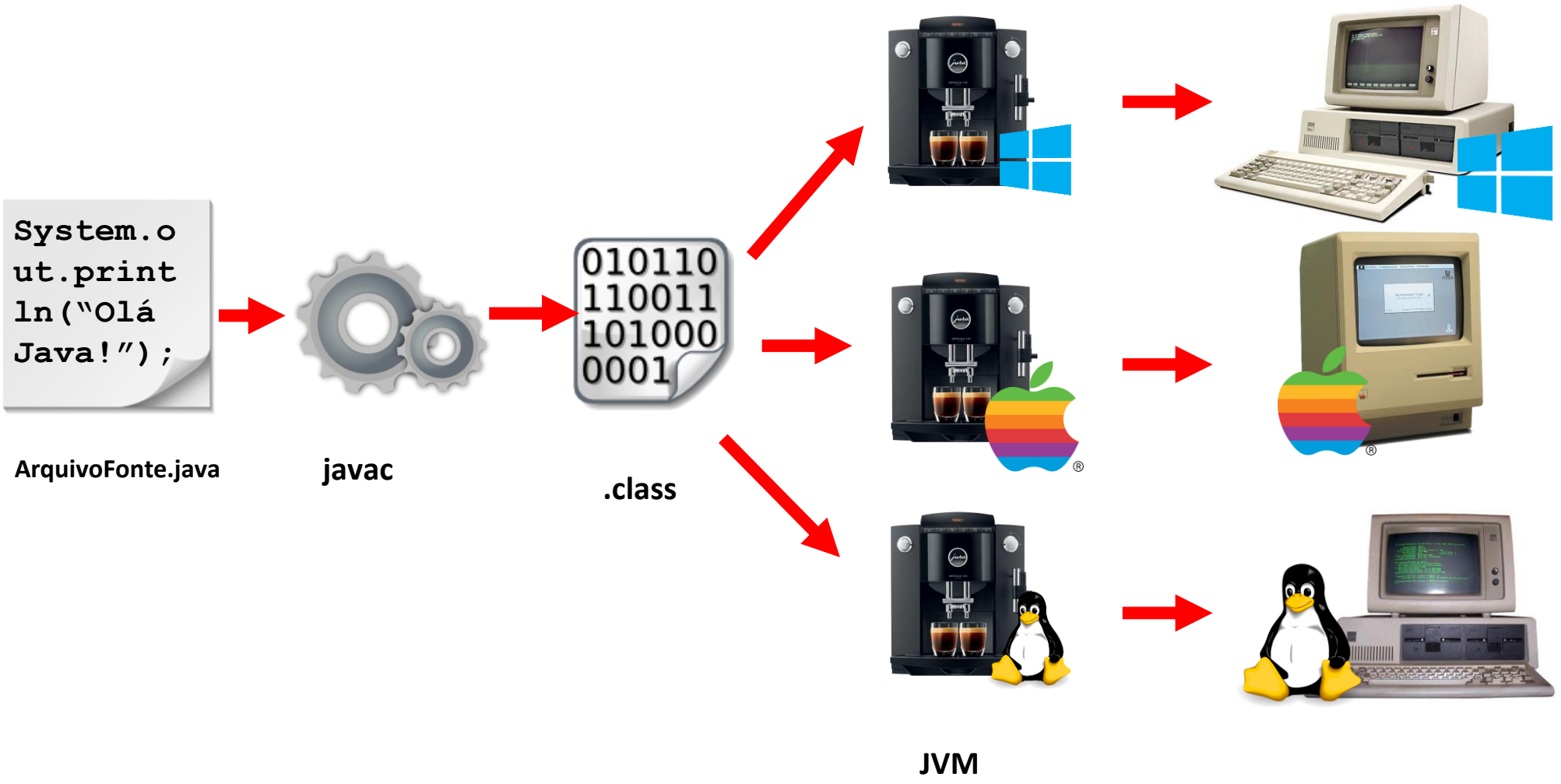
Popularização do JAVA até hoje...



- Em 1997 a NASA utilizou a linguagem Java na comunicação do primeiro rover marciano com a terra;
- Isso levou a linguagem aos noticiários e iniciou seu processo de pupularização.

- Hoje a linguagem Java está entre as 3 mais utilizadas no mundo;
- Utilizada em sistemas embarcados, desenvolvimento web, desktop, mobile (Android foi construído em C mas as aplicações rodam em Java)

Como o Java funciona...(multiplataforma)



Distribuições

Standard Edition **SE**

Enterprise Edition **EE**

Micro Edition **ME**

SE

EE

ME

JVM? JRE? JDK? O que devo baixar?

- JVM = apenas a virtual machine, esse download não existe, ela sempre vem acompanhada no JRE.

- JRE = Java Runtime Environment, ambiente de execução Java, formado pela JVM e bibliotecas, tudo que você precisa para executar uma aplicação Java.

- JDK = Java Development Kit: Nós, desenvolvedores, faremos o download do JDK do Java SE (Standard Edition). Ele contém o compilador.

- http://java.sun.com

E quanto ao NetBeans?

- NetBeans é “apenas” uma IDE;

- Existem outras (como Eclipse)
- É a que vamos utilizar na disciplina: https://netbeans.apache.org/download/index.html

Primeiro programa

- Faça o download e instale o NetBeans;
- Abra o NetBeans;
- Acesse o menu Arquivo->Novo Projeto;
- Categoria=Java, Projeto=Aplicação Java
- Coloque o nome: MeuPrograma
- Adicione o conteúdo conforme a seguir;
- Mande “Executar o Projeto” (ícone do “Play” = F6)

```
public class MeuPrograma {
    public static void main (String args[]) {
        System.out.printf("Olá Mundo Java!");
    }
}
//Comentário de uma linha
```

Elementos Básicos em Java - Tipos de dados

- Java possui oito tipos simples de dados:
  - Lógicos: boolean (verdadeiro/falso):
  - Caracteres: char
  - Inteiros: byte, short, int, long
  - Números de ponto flutuante: float e double

Variáveis de Tipos Primitivos

- Em Java, uma variável deve: ser declarada antes de ser usada. Ter um tipo definido (o tipo não muda). Iniciar o valor da variável antes de usá-la ser usada dentro do escopo (método ou bloco).

Tipo	*Tamanho (bits)*	*Valor Mínimo*	*Valor Máximo*	*Sem Sinal*
boolean	1	false	true	X
char	16	0	$2^{16} - 1$	X
byte	8	-2^{7}	$2^{7} - 1$	
short	16	-2^{15}	$2^{15} - 1$	
int	32	-2^{31}	$2^{31} - 1$	
long	64	-2^{63}	$2^{63} - 1$	
float	32			
double	64			

Tipos primitivos

Lógico: (verdadeiro/falso) - boolean

O tipo booleano pode representar dois estados:

true ou false

```
boolean result = true;
```

Na instrução acima, é declarada uma variável chamada result do tipo boolean e lhe é atribuída o valor true.

Literal (texto - caractere) - char

```
char letra = 'a';
```

Tipos primitivos

Inteiro (positivos ou negativos)

- `byte b = 97;`
- `short s = 97;`
- `int i = 97;`

Tamanho em memória	*Dado primitivo*	*Faixa*
8 bits	byte	-2^{7} até $2^{7}-1$
16 bits	char	0 até $2^{16}-1$
16 bits	short	-2^{15} até $2^{15}-1$
32 bits	int	-2^{31} até $2^{31}-1$
64 bits	long	-2^{63} até $2^{63}-1$

Nas instruções acima, são declaradas variáveis chamadas b, s e i, cada uma representando um determinado tipo inteiro e lhes são atribuídas o valor 97.

- **byte** (8 bits) – de (-128) até (127)
- **short** (16 bits) – de (-32.768) até (32.767)
- **int** (32 bits) – de (-2.147.483.648) até (2.147.483.647)
- **long** (64 bits) – de (-9.223.372.036.854.775.808) até (9.223.372.036.854.775.807)

Tipos primitivos

Real (ponto flutuante – números com casas decimais, positivos ou negativos) – **float** (32 bits) **double** (64 bits).

Tamanho em memória	*Dado primitivo*	*Faixa*
32 bits	float	-10^{38} até $10^{38}-1$
64 bits	double	-10^{308} até $10^{308}-1$

Exemplos:

- **`float`** `pi = 3.14f;`
- **`double`** `tamanho = 3.672938619846274;`

Identificadores

- As regras para nomeação de identificadores (**variáveis, nomes de funções ou classes**) seguem a seguinte regra:

- Nomes devem começar com letra ou os caracteres _ ou $ os caracteres seguintes podem conter números, letras, _ ou $

- Veja exemplos de nomes de identificadores:
  - `valor` `// válido`
  - `$preco` `// válido`
  - `20itens` `// inválido`
  - `_teste` `// válido`
  - `IDADE` `// válido`

- Observação: O Java considera diferença entre maiúsculas e minúscula.

Palavras-Chaves do Java

- O Java possui 53 palavras-chaves (palavras reservadas)

abstract	class	extends	implements	null	strictfp	true
assert	const	false	import	package	super	try
boolean	continue	final	instanceof	private	switch	void
break	default	finally	int	protected	synchronized	volatile
byte	do	float	interface	public	this	while
case	double	for	long	return	throw	
catch	else	goto	native	short	throws	
char	enum	if	new	static	transient	

Declaração de variáveis

A declaração de variáveis em Java é dada da seguinte forma:

```
tipoDaVariavel nomeDaVariavel;
```

Exemplo, variável idade que armazena um inteiro:

```
int idade;
```

Pronto, agora a variável “idade” existe perante seu programa.

Atribuição de variáveis

A atribuição de valores a uma variável em Java:

```
nomeDaVariavel = valorDaVariavel;
```

Exemplo, variável idade deve valer agora quinze:

```
idade = 15;
```

Atribuição de variáveis

Após a atribuição, você pode utilizar o valor atribuído a variável, segue alguns exemplos:

```
 //calcula a idade no ano seguinte
int idade = 25;
int idadeNoAnoQueVem;
idadeNoAnoQueVem = idade + 1;
System.out.println(idadeAnoQueVem);
```

Operadores matemáticos

- Operadores +, -, /, * e %, sendo que o operador % (módulo) é que o resto de uma divisão inteira. Exemplos:

```
• int quatro = 2 + 2;
• int tres = 5 – 2;
• int oito = 4 * 2;
• int dezesseis = 64 / 4;
• int um = 5 % 2;
```

- 5 dividido por 2 dá 2 e tem resto 1;
- o operador % pega o resto da divisão inteira

Método *main( )*

A assinatura do método *main( )* é o ponto de partida de um programa Java:

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

O parâmetro passado para o método *main( )* é um array de Strings, que contém os valores dos argumentos passados na linha de comando da execução do programa.

Exemplo:

java *nomePrograma* argumento1 argumento2 argumento3

Strings

String é uma **classe** que manipula cadeias de caracteres, possuindo métodos para manipulações.

```
String str = "Isto é uma String do Java";
String xyz = new String("Isto é uma String do Java");

if( str == xyz ) System.out.println("IGUAL");
else System.out.println("DIFERENTE");

if( str.equals( xyz ) ) {
 //MANEIRA CORRETA DE SE COMPARAR O CONTEÚDO DAS STRINGS
}

System.out.println( "Tamanho da String: " + str.length() );

System.out.println( "SubString: " + str.substring(0, 10) );

System.out.println( "Caracter na posição 5: " + str.charAt(5) );
```

Strings

```
String str = "Isto é uma String do Java";

// O método split quebra a String e várias outras,
// pelo separador desejado
String[] palavras = str.split(" ");

int i = str.indexOf("uma"); //retorna o índice da palavra na String

if( str.startsWith("Olá") || str.endsWith("Mundo!") ) {
  // testa o começo e o fim da String - retorna boolean
}

str = str.trim();  // elimina os espaços em branco no início e fim

str = str.replace('a','@');  // substitui os caracteres

// substitui uma palavra (usa expressões regulares)
str = str.replaceAll("String","Cadeia de caracteres");
```

Operadores unários

Incremento e Decremento: ++ e --

```
int a = 0;
int b = a++;    //incrementado depois de atribuir
int c = ++a;    //incrementado antes de atribuir
b = a--;        //decrementado depois de atribuir
c = --a;        //decrementado antes de atribuir
```

Mais e Menos Unário: + e -

```
int x = +3;     //x recebe o positivo 3
x = -x;         //x recebe -3, neste caso
```

Inversão de Bits: ~

```
int i = ~1;     //i = -2  (os bits foram invertidos)
```

Complementar booleano: !

```
boolean falsidade = ! (true);     // inverte o valor booleano
```

Conversão de Tipos: (tipo)

```
double d = 1.99;
int i = (int) d;   // converte de double p/ int (perda de precisão)
```

Operadores Aritméticos

Multiplicação e Divisão: * e /

```
int um = 3 / 2;                         // divisão de inteiros gera um inteiro
float umEmeio = (float) 3 / 2;          // ocorre promoção aritmética para float
double xyz = umEmeio * um;              // ocorre promoção aritmética para float
```

Módulo: %

```
int resto = 7 % 2;                      // resto = 1
```

Adição e Subtração: + e -

```
long l = 1000 + 4000;
double d = 1.0 - 0.01;
```

Concatenação:

```
long var = 12345;
String str = "O valor de var é " + var;
```

Na concatenação de Strings, as variáveis ou literais são promovidos a String antes:

```
String str = "O valor de var é " + Long.toString( var );
```

Operadores de Comparação

Comparação ordinal: >, >=, < e <=

Compara tipos primitivos numéricos e o tipo char.

```
boolean b = ( 10 < 3 );
boolean w = (x <= y);
if( x >= y ) { }
```

Operador instanceof

Compara o tipo da classe de uma referência de um objeto.

```
String str = "Uma String";
if( str instanceof String  ) { }   // true
if( srt instanceof Object  ) { }   // true
```

Comparação de Igualdade: == e !=

Comparam tipos primitivos, valores literais e referências de objetos.

```
if( abc == 10 ) { }
boolean b = ( xyz != 50 );
if( refObj1 == refObj2 ) { }
```

Operadores Lógicos: && e ||

```
if( (a>10) && (b<5) )
   { // isso
}

if( (x==y) || (b<5) )
   { // aquilo
}
```

Operadores ternários: ? :

O código do operador ternário abaixo:

```
int x = 10;
int y = (x > 10) ? x : x+1;
```

é semelhante ao código abaixo:

```
int x = 10;
int y;
if( x > 10 ) {
   y = x;
} else {
   y = x + 1;
}
```

Operadores de Atribuição

Estes operadores atribuem um novo valor a uma variável ou expressão.
O operador = apenas atribui um valor.
Os operadores +=, -=, *= e /= calculam e atribuem um novo valor.

```
int i = 10;

int dois = 1;
dois += 1;        // dois = dois + 1;

int cinco = 7;
cinco -= 2;       //  cinco = cinco - 2;

int dez = 5;
dez *= 2;         // dez = dez * 2;

int quatro = 12;
quatro /= 3;      // quatro = quatro / 3;
```

Conversão de Tipos Primitivos (cast)

- Permite a conversão entre tipos diferentes
- Pode ser implícito de um tipo menor para um maior
- Deve ser explícito quando for de um tipo maior para um menor:
  - Pois pode causar perda de precisão e truncamento.

```
double d = 1.99d;
int i = (int) d;      // i recebe o valor 1

short s = 15;
long x = s;           // conversão explicita
long y = (long) s;    // não é necessária
```

Controles de Fluxo do Programa

- if / else
- switch ()
- while ()
- for ()

Cláusula if( ) / else

```
public class ClausulaIf {
  public static void main( String[] args ) {
   int idade = 20;
    if( idade <= 12 ) {
       System.out.println( "Criança" );
    }
    else if( idade <= 19 ) {
       System.out.println( "Adolescente" );
    }
    else if( idade <= 60 ) {
       System.out.println( "Adulto" );
    }
    else {
       System.out.println( "Idoso" );
    }
  }
}
```

Cláusula switch( )

```
public class ClausulaSwitch {
  public static void main( String[] args ) {
    int numero = 1;
    switch( numero ) {
      case 1 :
        System.out.println( "UM" );
        break;
      case 2 :
        System.out.println( "DOIS" );
        break;
      case 3 :
        System.out.println( "TRES" );
        break;
      default :
        System.out.println( "NENHUM" );
        break;
    }
  }
}
```

O switch recebe um argumento do tipo int.

Laço while( )

```
public class LacoWhile {
   public static void main( String[] args )
      { int i = 0;

        while( i < 10 ) {
            System.out.println( "Linha: " + i );
            i++;
      }
   }
}
```

A expressão é avaliada antes de executar o bloco de código
Ele repete enquanto a expressão for verdadeira (true)

Laço do / while( )

```
public class LacoWhile {
   public static void main( String[] args )
      { int i = 0;

        do {
            System.out.println( "Linha: " + i );
            i++;
      } while( i < 10 );
   }
}
```

O bloco é executado ao menos um vez
Após a primeira repetição é que a expressão é avaliada

Laço for( )

A sua estrutura é definida como a seguir:

```
for( iniciação; condição; incremento ){
     bloco_de_código_a_executar;
}
```

```
public class LacoFor {
  public static void main( String[] args )
    { for( int i=0; i < 10; i++ ) {
       System.out.println( "Linha: " + i );
    }
   }
}
```

Laço for( ) avançado (Enhanced for loop)

- Evita erros ao percorrer arrays e coleções (implementações de java.util.Collection).
- É similar ao ***for each*** de outras tecnologias.

```
public class LacoForAvancado {
  public static void main( String[] args ) {
    for( String s : args ) {
      System.out.println("Argumento: " + s );
    }
  }
}
```

```
List lista = new ArrayList();
// adiciona itens à lista
for( String s : lista ) {
  System.out.println( s );
}
```

Cláusula break

- Aborta a execução de um laço, quando executado.

```
public class ClausulaBreak {
  public static void main( String[] args ) {
    char letras[] = { 'A', 'B', 'C', 'D', 'E' };
    int i;
    for( i=0; i<letras.length; i++ )
      { if( letras[i] == 'C' ) {
        break;
      }
    }
    System.out.println( "Último índice: " + i );
  }
}
```

Lendo e atribuindo valores às variáveis

- Inteiro: scanner.nextInt();
- Double: scanner.nextDouble();
- Float: scanner.nextFloat();

- Necessita importar java.util.Scanner;

```
Scanner scanner = new Scanner(System.in);
String valor = scanner.next();
System.out.println(valor);
```

Literais

- boolean: true e false

- inteiro: 10, 0x10, 010 (decimal, hexadecimal e octal, respectivamente)
- ponto-flutuante: 1.99, 2.55f, 10.99d, 4.23E+21 (double, float, double e notação científica)

- Caracteres de escape do tipo char:

```
'\n' - quebra de linha
'\r' - retorno ao início da linha
'\t' - tabulação
'\\' - barra invertida
'\b' - backspace
'\f' - form feed
'\'' - aspa simples
'\"' - aspa dupla
```

Exercício J.1.1:

- Escreva o mesmo programa de teste alterando-o para que ele peça o seu nome e imprima a mensagem “Bem vindo ao Java [SEU NOME]!”

- Dica: Para ler uma string utilize:
- Scanner sc = new Scanner(System.in);
- String nome = sc.nextLine();

- Vamos precisar importar a biblioteca java.util.Scanner;
- Teste a impressão de 2 formas:
  - Utilizando System.out.printf //(funciona igual ao do C)
  - Utilizando System.out.println com concatenação de strings //(muito mais fácil em java)

Exercício J.1.2:

- Faça um programa em Java que peça um número inteiro **n**;
- Depois faça um laço preenchendo um vetor do tipo float com **n** posições (valores informados pelo usuário);
- Ao final o programa deve imprimir o maior valor, o menor, a média e o somatório de valores.

Exercício J.1.3 - Entregar no virtual:

- Escolha um programa que você fez em C na disciplina passada ou em Python nessa mesma disciplina.
- Traduza-o para Java;
- Poste o projeto completo do NetBens compactado em .zip na tarefa do virtual para
- Traga ambos na próxima aula.

https://vidadeprogramador.com.br/2017/09/26/orientacao-a-objetos/