



Sintaxe Java



Sumário

- Estrutura de um programa
- O utilitário Javadoc
- Tipos de dados
- Variáveis
- Entrada e Saída de Dados (Formatada e não-formatada)
- Java doc



Estrutura de um programa em Java

- Um programa em Java pode ser composto de um ou mais arquivos-fonte (.java), que podem conter:
 - Uma declaração de pacote (package);
 - Uma ou mais diretivas de importação (import);
 - Uma ou mais declarações de classe (class);
 - Uma ou mais declarações de interface (interface);



Entendendo a estrutura do programa

```
package org.ufpr.exerciciosloo;  
  
import java.util.Scanner;  
  
public class BemVindo {  
  
    public static void main(String args[]) {  
        System.out.println("Digite seu nome:");  
        Scanner scn = new Scanner(System.in);  
        String nome = scn.next();  
        System.out.println("Bem vindo, "+ nome);  
    }  
}
```

Declaração do pacote (opcional)

Importação das bibliotecas
(APIs necessárias)
(obrigatório se utilizar)

Declaração da classe
(obrigatório)

Declaração do método
main
(obrigatório)



Tipos de Dados

- Um tipo de dado estabelece um conjunto de valores que podem ser representados dentro de um programa;
 - Possuem operações específicas;
- Os tipos de dados que fazem parte de uma linguagem são chamados de tipos de dados primitivos;



Tipos de Dados Primitivos do Java

Categoria	Tipo	Tamanho	Intervalo
Inteiro	byte	1	de -128 a +127
	short	2	de -32.768 a +32.767
	int	4	de -2.147.483.648 a +2.147.483.647
	long	8	de -9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
Real	float	4	aproximadamente de -3,4E+38 a +3,4E+38
	double	8	aproximadamente de -1,7E+308 a +1,7E+308
Caractere	char	2	de \u0000 a \uFFFF
Lógico	boolean	1	false e true



Caracteres Especiais

- Um caractere especial é um símbolo que produz um efeito particular, mas que não possui uma forma visual;

Representação	Significado
\n	Pula linha
\r	Retorno de carro
\b	Retrocesso (backspace)
\t	Tabulação
\f	Nova página
\'	Apóstrofe
\"	Aspas
\\	Barra invertida
\u223d	Caractere UNICODE 223d
\g37	Octal 37
\fca	Hexadecimal fca



Variável

- Uma variável é um nome definido pelo programador, ao qual pode ser associado um valor pertencente a certo tipo de dados;
- Corresponde a uma área de memória, representada pelo seu nome;
- Variável \rightarrow Nome + Tipo + Conteúdo;



Variável no Java

- No Java, o nome de uma variável pode ser formado por uma seqüência de um ou mais caracteres alfabéticos e numéricos;
- Iniciado por letra, _ ou \$;
- A linguagem é sensível ao caixa;
- Declaração:
 - `<tipo> <nome> [, nome2 [, nome3 [..., nome N]]];`
 - `int i;`
 - `float total, preco;`
 - `char letra = 'c';`



Palavras Reservadas

- Toda linguagem de programação possui palavras que indicam os nomes dos tipos primitivos, as diretivas da linguagem, e outros elementos que pertencem a sua sintaxe;
- São as palavras reservadas e, por isso, não podem ser utilizadas para outro fim;



Palavras Reservadas no Java

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

Exemplo Variáveis

```
public class ExVariaveis
{
    public static void main (String args[])
    {
        int a = 5;
        int b = 2;
        float pi = 3.14f;
        char c = 'A';
        char tab = '\t';
        char enter = '\n';

        System.out.println ("a= " + a);
        System.out.println ("b= " + b);
        System.out.println ("pi= " + pi);
        System.out.println ("c= " + c);
        System.out.println ("tab= " + tab + "fim do tab");
        System.out.println ("enter= " + enter + "fim do
enter");
    }
}
```

Diferencie
onde está a
declaração
e o **uso** da
variável



Escopo de Variáveis

- O conjunto de locais onde uma declaração de variável tem validade é chamado de escopo;
- As variáveis, no Java, tem validade dentro do seu bloco de comandos (`{ }`), a partir do ponto onde foram declaradas;
- O uso de variáveis ou objetos fora de seu escopo constitui um erro de programação detectado pelo compilador;



Exemplo de Escopo

```
public class ExEscopo
{
    public static void main (String args[])
    {
        //Inicio do Bloco 0
        int i = 5;
        {
            //Inicio do Bloco 1
            int j = 0;
            j = 5 * i;
            System.out.println (i); //Dentro do escopo de i
            System.out.println (j); //Dentro do escopo de j
            {
                //Inicio do Bloco 2
                System.out.println (i); //Dentro do escopo de i
                System.out.println (j); //Dentro do escopo de j
            }
        }
        System.out.println (i); //Dentro do escopo de i
        System.out.println (j); //ERRO! Fora do escopo de j
    }
}
```



Entrada de dados não formatada

- `char c = System.in.read();`
 - `System` : Classe que disponibiliza um objeto `in` para a aplicação;
 - `in` : Objeto que representa a entrada de dados do console em qualquer ambiente;
 - `read` : Método que lê caracteres individuais da entrada;
- O uso de `in` é complexo pois permite apenas a leitura de caracteres;



Saída de dados não formatada

- `System.out.println ("Olá!");`
 - `System` : Classe que disponibiliza um objeto out para qualquer aplicação;
 - `out` : Objeto que representa a saída para o console em qualquer ambiente;
 - `println` : Método que imprime valores, mensagens e objetos;
- `System.out.print ("Olá!");`



Entrada de dados formatada

- A partir da versão 5 do Java foi incluída a classe `Scanner`;
- Esta classe pertence ao pacote `java.util`, que deve ser declarado na diretiva `import` no início do programa;
- A leitura do dado é feita pelo método *next<tipo>()*: `nextInt()`, `nextByte()`, `nextDouble()`, `nextFloat()`, `nextLong()`, `nextShort()`;
- Para `String`, utilizar `next()` ou `nextLine()`;

Exemplo de entrada de dados formatada

Ver no JavaDoc

```
//Importação da classe Scanner do pacote java.util
import java.util.Scanner;

public class ExEntradaFormat
{
    public static void main (String args[])
    {
        System.out.println ("Olá!"); //Mensagem inicial

        //Exibe mensagem e prepara entrada de dados
        System.out.println ("Digite um inteiro: ");
        Scanner s = new Scanner (System.in);
        int valor = s.nextInt(); //declara e atribui valor à
variável

        //exibição do valor lido
        System.out.println ("Valor digitado = " + valor);
    }
}
```



Saída de dados - Não formatada

- `System.out.println ("Olá!");`
 - `System` : Classe que disponibiliza um objeto out para qualquer aplicação;
 - `out` : Objeto que representa a saída para o console em qualquer ambiente;
 - `println` : Método que imprime valores, mensagens e objetos;
- `System.out.print ("Olá!");`



Saída de dados formatada

- `System.out.printf(<formatacao> [, expr1 [, expr2 [...]]]);`
 - A formatação é uma String que pode conter apenas uma mensagem ou uma mensagem intercalada com marcadores para inclusão de valores;
 - Quando os marcadores estão presentes, deve ser fornecido o mesmo número de expressões;
 - Quando a mensagem é impressa, cada marcador é substituído pela expressão correspondente;
 - Utilização da formatação:
`%[argument_index$][flags][width][.precision]conversion`
 - Classe Formatter:
<http://download.oracle.com/javase/7/docs/api/>



Exemplo de saída de dados formatada

```
import java.util.Scanner;

public class ExSaidaFormat
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner (System.in);
        System.out.println ("Digite um número: ");
        double numero = sc.nextDouble();
        System.out.printf ("Saída com uma casa
decimal: %.1f", numero);
    }
}
```



javaDoc: Comentários

- `//` Comentário de linha
- `/*` Comentário de bloco
(múltiplas linhas) `*/`
- `/**` Comentário de documentação
(múltiplas linhas). Utilizado para execução do
`javadoc*/` -> Ver
<http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>



javaDoc: Exemplo de classe comentada

```
package org.ufpr.exerciciosloo;

import java.util.Scanner;
/** Classe com um único método
 *  @author Rafael
 */
public class BemVindo {
    /*Método main.
    A JVM chama este método quando inicia*/
    /**
    * Método principal da classe que solicita o nome do usuário
    e imprime uma mensagem.
    * @param args Os argumentos da linha de comando.
    */
    public static void main(String args[]){
        System.out.println("Digite seu nome:");
        Scanner scn = new Scanner(System.in); //Obtém entrada
        String nome = scn.next();
        System.out.println("Bem vindo, "+ nome);
    }
}
```



Executando o javaDoc

- A JDK possui um aplicativo que compila o código com a intenção de gerar uma documentação sobre ele: javadoc
- Com a seguinte linha de comando é possível criar uma documentação do seu projeto, desde que você utilize as anotações corretas no código
- Para o exemplo anterior podemos gerar o javaDoc com a seguinte linha de comando:

```
javadoc -d .\javadoc org.ufpr.exerciciosloo
```




javaDoc: Entendendo a linha de comando do javadoc

- É necessário executar o comando do javadoc da raiz de seu projeto.
- Neste caso o nosso projeto está no pacote `org/ufpr/exerciciosloo`
- A opção `-d` possibilita colocar o resultado (HTMLs gerados) em um diretório separado



Java Doc gerado

All Classes

BemVindo

Package **Class** Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

org.ufpr.exerciciosloo

Class BemVindo

java.lang.Object
org.ufpr.exerciciosloo.BemVindo

public class BemVindo
extends java.lang.Object

Classe com um único método

Constructor Summary

Constructors

Constructor and Description
BemVindo ()

Method Summary

Methods

Modifier and Type	Method and Description
static void	main(java.lang.String[] args)



Codificação em grupo (DOJO)

- Passos de bebê
- Programação em duplas: um piloto e um co-piloto.
- Depois de 5 minutos o co-piloto tora-se piloto e algum “voluntário” assume o lugar do co-piloto



Referência Bibliográfica

- JANDL JUNIOR, Peter. Java: guia do programador. São Paulo: Novatec Editora, 2007