Guia de Referência (resumo) para Linguagem de Programação Java

(Prof. Bruno B. Boniati – Colégio Agrícola de Frederico Westphalen – Universidade Federal de Santa Maria)

Estrutura básica de uma aplicação

```
/* Estrutura básica de uma aplicação */

package nomeDoPacote; //Nome do pacote ao qual a classe faz parte

import java.util.*; //Importa as classes do pacote java.util

class Exemplo { //Declaração da classe
  public int numero; //atributo público
  private float metade; //atributo privado

  public void setNum(int n) { //Método
     numero = n;
     metade = n/2;
  }

  public Exemplo(int num) { //Construtor
     setNum(num);
  }

  public static void main (String args[]) {
     //corpo principal do programa
     Exemplo ex = new Exemplo(10);
  }
}
```

Tipos Primitivos

Tipo	Tamanho	Valores válidos
boolean	1 bit	**true** ou **false**
char	2 byte	0 a 65535
byte	1 byte	-128 a 127
short	2 bytes	-32.768 a 32.767
int	4 bytes	-2.147.483.648 a 2.147.483.647
long	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	4 bytes	$1.40129846432481707^{-45}$ a $3.40282346638528860^{+38}$
double	8 bytes	$4.94065645841246544^{-324}$ a $1.79769313486231570^{+308}$
String	2 bytes por caracter	Tamanho limitado à memória disponível.

Operadores

Tipo	Operador	Propósito	Exemplo
Aritméticos	+	Adição	a = 4 + 1; // 5
	-	Subtração	a = 4 – 1; // 3
	*	Multiplicação	a = 2 * 4; // 8
	/	Divisão	a = 8 / 2; // 4
	%	Módulo (resto da divisão)	a = 5 % 2; // 1
Concatenação	+	Concatenação de Strings	String a = "Olá " + "Mundo";
Atribuição	=	Atribuição simples	a = 50;
Lógicos	&&	"e" lógico	(a > 1) && (b < 1)
	||	"ou" lógico	(a > 1) || (b < 1)
	!	não (inversão)	!(a > 2)
Condicionais (Comparação)	==	igualdade de valores ou endereços dos objetos.	(a == 0)
	!=	diferente de	(a != 0)
	<	menor que	(a < 0)
	>	maior que	(a > 0)
	<=	menor ou igual a	(a <= 0)
	>=	maior ou igual a	(a >= 0)
	instanceof	Verificação de tipo ()	//x é uma String?) (x instanceof String)
Incremento e **Decremento**	++	Incremento	a++;
	--	Decremento	a--;
Conversão	(tipo)	Conversão de tipo	int b = (int) 40.5;
Classe	new	Criação de objeto	Aluno a = new Aluno();

Saída de Dados

System.out.println(Objeto);

Função para saída de valores

Ex.:
```
System.out.println("Olá Mundo");
System.out.println(10);
```

Modificadores de Acesso

- **public** → podem ser vistos e utilizados por qualquer classe;
- **private** → só podem ser vistos e utilizados pela própria classe;
- **protected** → podem ser vistos e utilizados pela própria classe e classes filhas;

Comandos da Linguagem

Comando	Propósito	Sintaxe
Declaração de variável	Declaração de variável	`tipo nome_variavel = valor_inicial;`
Declaração de constante	Declaração de constante	`final tipo nome_constante = valor;`
Bloco	Marcar um bloco de cód.	`{ } //Abre e fecha chaves "{}"`

if	Comando condicional	`if (a > b) {`
		`  System.out.println("A é maior que B");`
		`} else {`
		`  System.out.println("A é igual ou menor que B");`
		`}`

switch	Comando condicional	`switch (i) {`
		`  case 0 :`
		`  System.out.println("ZERO");`
		`  break;`
		`  case 1:`
		`  System.out.println("UM");`
		`  break;`
		`  case 2:`
		`  System.out.println("DOIS");`
		`  break;`
		`}`

while	Laço com pré validação	`int i = 1;`
		`while (i <= 10) {`
		`  System.out.println(i++);`
		`}`

do	Laço com pós validação	`int i = 1;`
		`do {`
		`  System.out.println(i++);`
		`} while (i <= 10);`

for	Laço simplificado	`for (i=1;i<=10;i++){`
		`  System.out.println(i);`
		`}`

break	Saída de bloco	`break;`
continue	Reinício de bloco	`continue;`
return	Retorno de método	`return <valor ou objeto>;`

Métodos (Sub-rotinas)	Funções	`float area(float altura, float base) {`
		`  return altura * base;`
		`}`

	Procedimentos	`void area(float altura, float base) {`
		`  System.out.println(altura * base);`
		`}`

Vetores	Variáveis unidimensionais	`int v[] = new int[10]; //Vetor de inteiros`
		`//v[0] é o primeiro elemento e v[9] o último`

Matrizes	Variáveis multidimensionais	`float mat[][] = new float[4][3];` `//Tabela com 4 linhas e 3 colunas`

synchronized	Seção Crítica	`synchronized void xxx {`
		`  //código sincronizado`
		`}`
		`void xxx() {`
		`  synchronized(this) {//código sincronizado}`
		`}`

throw	Levantamento de Exceção	`throw new java.lang.Exception("Erro Teste");`

try	Tratamento de Exceção	`try {`
		`  //código que pode gerar um erro`
		`} catch (XException x) {`
		`  //tratamento de erros do tipo XException`
		`} catch (YException y) {`
		`  //tratamento de erros do tipo YException`
		`} finally {`
		`  //finalização, sempre sera executado após`
		`//o try (independente da ocorrencia de erros`
		`}`