Técnicas de Programação e Algoritmos - TPA

– Tipos de Dados

- JAVA -

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)

Declaração: `<tipo da variável> <nome da variável>;`
Declaração e atribuição: `<tipo> <nome> = <valor>;`

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 de dados em Java

- **int**: Formado por 32 bits, suporta valores entre -2.147.483.648 e 2.147.483.648;
- **long**: Formado por 64 bits, é usado quando você realmente precisa guardar valores muito altos;
- **float**: Formado por 32 bits, é usado para guardar valores em ponto flutuante que possuam até 7 casas decimais;
- **double**: Formado por 64 bits, é utilizado quando se deseja armazenar valores em ponto flutuante com até 15 casas decimais. Um valor em ponto flutuante no Java é, por default, um valor double;
- **boolean**: Possui apenas os valores lógicos true (verdadeiro) e false (falso);
- **char**: Formado por 16 bits, armazena caracteres Unicode (além da tabela ASCII, possibilita o uso de caracteres que possam ser utilizados em qualquer idioma do mundo).

Operadores

- Aritméticos

Função	Sinal
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da divisão	%
Incremento	++
Decremento	--

Operadores

- Relacionais

Função	Sinal
Igual	==
Diferente	!=
Maior que	>
Maior ou igual a	>=
Menor que	<
Menor ou igual a	<=

Operadores

- Lógicos

Função	Sinal
E	&&
OU	||
Não	!

Conversão de tipos

Supondo a variável x	Converter em	y recebe o valor convertido
✓ **Entre tipos numéricos**		
int x = 10	float	float y = (float) x
int x = 10	double	double y = (double) x
float x = 10.5	int	int y = (int) x
✓ **De string para numéricos**		
String x = “10”	int	int y = Integer.parseInt(x)
String x = “20.5”	float	float y = Float.parseFloat(x)
String x = “20.5”	double	double y = Double.parseDouble(x)
✓ **De numéricos para string**		
int x = 10	String	String y = Integer.toString(x) ou String y = String.valueOf(x)
float x = 10.5	String	String y = Float.toString(x) ou String y = String.valueOf(x)
double x = 10.5	String	String y = Double.toString(x) ou String y = String.valueOf(x)

Inserção de Comentários

```
// Comentários em uma única linha


/* Comentários em
 * várias linhas
 */


/** Comentários inseridos no formato reconhecido
 *  por um utilitário de documentação chamado javadoc
 *  fornecido pela Sun junto com o JDK
 */
```

Identificadores

As regras para nomeação de identificadores (variáveis, nomes de função, classes ou label) seguem a seguinte regra:

- nomes 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
INT        // válido
```

Observação: O Java considera diferença entre maiúsculas e minúscula.

Recursos básicos da linguagem

```
// Declaração de variáveis
int num1 = 0, op = 0;
double valor;
String usuario;

// Declaração de constantes
final double pi = 3.1416;
```

- Tipos primitivos são escritos sempre com letras minúsculas.
- As variáveis devem ser inicializadas na declaração
- O Java disponibiliza algumas classes que podem ser utilizadas como tipos (como a String no exemplo acima).

Operadores

Veremos agora os operadores da linguagem Java, que agregam importantes funcionalidades aos programas.

Eles possuem uma ordem de precedência na execução da expressão.

Para garantir a ordem de precedência desejada, agrupe as expressões com parênteses.

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 Lógicos de Curto Circuito: && e ||

Estes operadores não precisam testar toda a expressão.
Ele pára assim que uma das condições o satisfaça.
O retorno da expressão é um boolean

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

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

boolean b = x && y || z;
```

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;
```

Algumas Funções

Potência

```
Math.pow(x, y);
```

Legenda:
x = base
y = potência

Raiz Quadrada

```
int x = 2;

Math.sqrt(x);
```

ou pode ser o prórprio numero:

```
Math.sqrt(2);
```

Math é uma “biblioteca” de funções matemáticas. Nela encontramos funções para cálculos de:

-- seno, coseno, tangente, logaritmos, raiz cúbica, etc.

Variáveis de Tipos Primitivos

```
public class TiposPrimitivos {
  public static void main( String[] args ) {
    //declara um int e atribui um valor
    int idade = 25;
    //declara um float e, depois, atribui um valor
    float valor;
    valor = 1.99f;
    //declarando um boolean
    boolean verdadeiroOuFalso = false;
    verdadeiroOuFalso = true;
    //declarando um char
    char letraA = 'A';
    letraA = 65;            //valor ASCII para 'A'
    letraA = '\u0041';      //valor Unicode para 'A'
    //declarando um byte
    byte b = 127;
    //declarando um short
    short s = 1024;
    //declarando um long
    long l = 1234567890;
    //declarando um double
    double d = 100.0;
    //declaração múltipla
    int var1=0, var2=1, var3=2, var4;
  }
}
```

Obs: Incluir código para “apresentar” na tela todas as variáveis declaradas.
Exemplo: system.out.println(“Valor da variável idade: “+idade);

String

String é uma classe que manipula cadeias de caracteres
A classe String possui métodos para essas manipulações
Trabalha com Pool de Strings para economizar memória

```
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) );
```

String - continuação

Outros métodos úteis da classe String:

```
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");
```

```java
public class Testando {
    public static void main(String[] args)
    {
        String str = "Isto é uma string do Java";
        String xyz = new String ("Isto é uma string do Java");
        // maneira errada de comparar string, portanto dará diferente
        if(str==xyz)
            System.out.println("IGUAL");
        else
            System.out.println("DIFERENTE");
        // maneira correta de comparar string
        if (str.equals(xyz))
        System.out.println("Tamanho: "+ str.length());
        System.out.println("Substring: "+ str.substring(0,10));
        System.out.println("Caracter na posição 5: "+ str.charAt(5));
        // outros métodos
        //Quebra sempre que encontrar o espaço
        String palavras[]=str.split(" ");
        System.out.println("palavras: "+ palavras[0]);
        System.out.println("palavras: "+ palavras[1]);
        System.out.println("palavras: "+ palavras[2]);
        //aponta o index
        int i = str.indexOf("uma");
        System.out.println("Indice: "+ i);// o indice conta a partir do 0
        //compara o início e o fim da string
        boolean teste =(str.startsWith("Olá") || str.endsWith("Mundo"));
        System.out.println("Resultado: " + teste);
        // elimina espacos vazios no início e fim da string
        str=str.trim();
        System.out.println(str);
        //substitui caracteres
        str=str.replace('a','@');
        System.out.println(str);
        // substitui palavras
        str=str.replaceAll("string","Cadeia de caracteres");
        System.out.println(str);
    }
}
```