Nome: _____ Data: ____/____/____

Estruturas fundamentais

Comentários:

Programas em Java podem ter dois tipos de comentários: comentários de implementação e de documentação:

Comentários em Java	
Tipo	**Forma**
Comentário de documentação	/** comentário de documentação */
Comentário de implementação	/* comentário de implementação*/
	//comentário de implementação até fim de linha

Tipos de dados:

Java é uma linguagem fortemente tipada, ou seja, as variáveis precisam tem os tipos bem definidos.

Há 8 tipos primitivos em Java: **byte**, **short**, **int**, **long**, **float**, **double**, **char** e **boolean**.

Tipos inteiros:

Tipo	tamanho	Intervalo
byte	8 bits	-128 a 127
short	16 bits	-32.768 a 32.767
int	32 bits	-2.147.483.648 a 2.147.483.647
long	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

Tipos de ponto flutuante:

Tipo	tamanho	Intervalo
float	32 bits	**Intervalo negativo:** -3,4028234663852886e+38 a -1,40129846432481707e-45 **Intervalo positivo:** 1,40129846432481707e-45 a 3,4028234663852886e+38 (6-7 dígitos significativos: IEEE 754)
double	64 bits	**Intervalo negativo:** -1,7976931348623157e+308 a -4,94065645841246544e-324 **Intervalo positivo:** 4,94065645841246544e-324 a 1,7976931348623157e+308 (15 dígitos significativos: IEEE 754)

Tipo caractere (char):

O tipo char denota caracteres segundo o esquema Unicode de codificação. Caracteres são representados entre aspas simpes (‘’).

Seqüência de controle	Nome	Valor Unicode
\b	Backspace	\u0008
\t	Tab	\u0009
\n	Linefeed	\u000a
\r	carriage return	\u000d
\”	aspas duplas	\u0022
\’	aspas simples	\u0027
\\	barra invertida	\u005c

Tipo lógico (boolean):

O tipo lógico em Java (boolean) tem dois valores possíveis: true e false.

Conversão de tipos numéricos:

A atribuição de uma variável de um tipo à outra de outro tipo; é possível em Java, sem conversão explícita de tipos, na seguinte ordem:

byte → short → int → long → float → double
char → int

Variáveis:

Um nome de variável precisa começar com uma letra “A..Z”, “a..z”, ou com underline “_”. Lembre-se que em Java, letra significa qualquer caracter Unicode que represente uma letra.

Declaração de variáveis: <Tipo> <nome da variável>;
<Tipo> <nome da variável> = <valor inicial>;

Operadores:

Operador	Direção
[] . () (chamada de método)	Da esquerda para a direita
! ~ ++ -- + - () (cast) new	Da direita para a esquerda
* / %	Da esquerda para a direita
+ -	Da esquerda para a direita
<< >> >>>	Da esquerda para a direita
< <= > >= instanceof	Da esquerda para a direita
== !=	Da esquerda para a direita
&	Da esquerda para a direita
^	Da esquerda para a direita
|	Da esquerda para a direita
&&	Da esquerda para a direita
||	Da esquerda para a direita
?:	Da esquerda para a direita
= += *= /= %= &= |= ^= <<= >>= >>>=	Da direita para a esquerda

Exemplo:

```
3 public class TesteOperadores {
4
5     public static void main(String[] args) {
6       int a = 5;
7       int b = 10;
8
9       System.out.println("a.............: " + a);
10      System.out.println("b.............: " + b);
11      System.out.println("-b............: " + (-b));
12      System.out.println("a + b.........: " + (a + b));
13      System.out.println("a - b.........: " + (a - b));
14      System.out.println("a * b.........: " + (a * b));
15      System.out.println("a / b.........: " + (a / b));
16      System.out.println("(float) a / b.: " + ((float) a / b));
17      System.out.println("a % b.........: " + (a % b));
18      System.out.println("++a...........: " + (++a));
19      System.out.println("a++...........: " + (a++));
20      System.out.println("--b...........: " + (--b));
21      System.out.println("b--...........: " + (b--));
22      System.out.println("a.............: " + a);
23      System.out.println("b.............: " + b);
24    }
25 }
```

Strings (java.lang.String):

Strings são seqüências de caracteres como "Senac". Java não possui um tipo primitivo string, ao invés disso possui uma classe chamada String. Nessa classe são definidos métodos para a manipulação de uma String, como extrair uma substring ou descobrir o seu tamanho.

Método	**Descrição**
charAt(int indice)	Retorna o caracter da posição indice
equals(String s)	Retorna true se a String for igual a s
indexOf(String s)	Retorna o índice da primeira ocorrência de s
length()	Retorna o comprimento da String
substring(int ini, int fim)	Retorna uma nova String contendo os caracteres a partir de ini (inclusivo), até fim (exclusivo)
trim()	Retorna uma nova String eliminando todos os espaços em branco que existirem no início ou no fim da String
toUpperCase()	Retorna uma nova String contendo todos os caracteres da original, porém com as minúsculas convertidas para maiúsculas
toLowerCase()	Retorna uma nova String contendo todos os caracteres da original, porém com as maiúsculas convertidas para minúsculas

Classes empacotadoras (wrappers):

Em várias situações precisamos trabalhar somente com objetos e não com tipos básicos. Para isso Java possui uma classe para cada tipo básico:

Tipo primitivo	Classe empacotadora
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Obs.: Essas classes possuem métodos para a conversão de String para seus respectivos tipos básico e para seus respectivos objetos.

Estruturas condicionais:

```
if (<condição>) {
    <instruções>;
}

if (<condição>) {
    <instruções>;
} else if (<condição>) {
    <instruções>;
}
```

```
if (<condição>) {
    <instruções>;
} else {
    <instruções>;
}

(<condição>) ? <verdadeiro> : <falso>
```

Exemplo:

```
3  public class TesteIf {
4
5      public static void main(String[] args) {
6
7          int valor = Integer.parseInt(args[0]);
8
9          if (valor > 10) {
10             System.out.println("Valor maior que 10!");
11         } else {
12             System.out.println("Valor menor que 10!");
13         }
14     }
15 }
```

Estruturas de repetição:

```
while (<condição>) {
    <instruções>;
}
```

```
do {
    <instruções>;
} while (<condição>);
```

```
for (<variável> = <valor inicial>; <condição>; <incremento/decremento>) {
    <instruções>;
}
```

Exemplos:

```
3 public class TesteWhile {
4
5     public static void main(String[] args) {
6         int i = 0;
7         while (i < 11) {
8             if ((i % 2) == 0) {
9                 System.out.println("Numero " + i + " e par !");
10            }
11            i++;
12        }
13    }
14 }
```

```
3 public class TesteFor {
4
5     public static void main(String[] args) {
6         for (int i = 1; i <= 15; i++) {
7             System.out.println ("O quadrado de " + i + " = " +  i * i);
8         }
9     }
10 }
```

Seleções múltiplas (switch):

```
switch (<opção>) {

    case <valor1>: {
        <instruções>;
        break;
    }
    case <valor2>: {
        <instruções>;
        break;
    }
    default: {
        <instruções>;
    }
}
```

Obs.: Pode-se testar os tipos: short, byte, int e char. Não pode-se usar faixas de valores.

Exemplo:

```
3 public class TesteSwitch {
4
5     public static void main(String[] args) {
6         for (int i = 1; i < 5; i++) {
7             switch (i) {
8                 case 1: {
9                     System.out.println ("Numero UM");
10                    break;
11                }
12                case 2: {
13                    System.out.println ("Numero DOIS");
14                    break;
15                }
16                case 3: {
17                    System.out.println ("Numero TRES");
18                    break;
19                }
20                default: {
21                    System.out.println ("NUMERO INVALIDO");
22                }
23            }
24        }
25    }
26 }
```

Arrays:

Os arrays representam uma forma de armazenar uma lista de itens que tem o mesmo tipo primitivo de dados, ou a mesma classe.

```
<tipo>[] <nome do array>;
<tipo>[] <nome do array> = new <tipo>[<tamanho>];
<tipo>[] <nome do array> = {<valor1>, <valor2>, <valor3>};
```

Exemplo:

```
3 public class TesteArray {
4
5     public static void main(String[] args) {
6         int[] arrayInteiros = new int[3];
7         arrayInteiros[0] = 20;
8         arrayInteiros[1] = 30;
9         arrayInteiros[2] = 50;
10        int total = 0;
11
12        for (int i = 0; i < arrayInteiros.length; i++) {
13            total = total + arrayInteiros[i];
14        }
15
16        System.out.println("Total:" + total);
17    }
18 }
```

```java
public class Histograma {

    public static void main(String[] args) {

        int[] array = {19, 3, 15, 7, 11, 9, 13, 5, 17, 1};

        String saida = "Elemento\tValor\tHistograma";

        for (int i = 0; i < array.length; i++) {

            saida += "\n" + i + "\t\t" + array[i] + "\t";

            for (int j = 0; j < array[i]; j++) {
                saida += "*";
            }

        }
        System.out.println(saida);
    }
}
```

Arrays multidimensionais:

Ao contrário de outras linguagens de programação, Java não possui arrays multidimensionais. Mas podemos criar um array de array, ou seja, teremos um array onde cara um dos elementos é um outro array.

O fato de Java usar arrays de array, permite que se construa arrays "irregulares", ou seja, que linhas diferentes tenham comprimentos diferentes:

```java
public class TesteArray1 {

    public static void main(String[] args) {
        int n = 9;
        int[][] valores = new int[n+1][n];

        for (int i = 0; i <= n; i++) {
            valores[i] = new int[i+1];
        }

        for (int i = 0; i < valores.length; i++) {
            for (int j = 0; j < valores[i].length; j++) {
                valores[i][j] = i;
                System.out.print(valores[i][j]);
            }
            System.out.println();
        }
    }
}
```

Métodos de classe:

Os métodos são as características comportamentais de uma classe, portanto são definidos nas mesmas. Eles podem ou não retornar valores para quem os chamou.

```
3  public class TesteMetodos {
4
5      public static void mensagem(String argumento) {
6          System.out.println(argumento);
7      }
8
9      public static double divide(int dividendo, int divisor) {
10         double resultado;
11         resultado = dividendo / divisor;
12         return resultado;
13     }
14
15     public static void main(String[] args) {
16         double valor = 0;
17
18         mensagem("usando um metodo void");
19         valor = divide(10, 5);
20         System.out.println("valor: " + valor);
21     }
22 }
```

Variáveis e constantes de classe:

Ocasionalmente, precisamos declarar variáveis que precisam ser vistas por todos os métodos da classe. Podemos também utilizar variáveis que seu valor não será alterado no decorrer do programa, essas variáveis são variáveis finais (final) e são chamadas contantes:

```
3  public class TesteVariaveis {
4
5      private static final float PERCENTUAL = 0.1f;
6      private static float       valor      = 0;
7
8      public static void calculaValor(float valorBase) {
9          valor = valorBase * PERCENTUAL;
10     }
11
12     public static void main(String[] args) {
13         calculaValor(500);
14         System.out.println(valor);
15     }
16 }
```