

## Lista Revisão Sintaxe Básica

Os programas devem ser feitos em um mesmo projeto chamado `ListaSintaxeBasica`. As classes deverão se chamar **ExercicioXX** (onde **xx** é o número do exercício), deverão conter o método `main` (pois devem executar) e estarem no pacote `com.lista.main`. **Isto será avaliado!**

A maioria dos exercícios abaixo devem utilizar os métodos estáticos da classe `com.terra.util.Teclado` da biblioteca externa `TerraUtil.jar` (disponibilizada no Moodle). Como inseri-la no projeto está detalhada na apostila.

1. Crie um arranjo de cinco inteiros e leia cada posição. Depois imprima todas as posições.
2. Peça para o usuário informar o tamanho de um arranjo de caracteres, crie um arranjo deste tamanho, peça para o usuário informar o valor de cada posição (use **for**) e depois imprima todas as posições (use **foreach**).
3. Declare e inicialize uma variável **int** chamada **x** e uma variável **double** chamada **y** com valor 5 e depois imprima se o valor delas é igual ou não (use **if**).

4. Leia dois pontos flutuantes e depois imprima o resultado de  $\frac{x^{120}}{\sqrt{y}}$ .

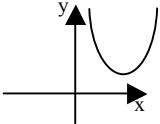
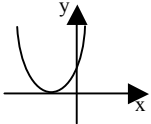
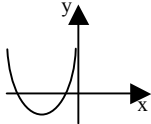
Dica: Entre na API Java e pesquise os métodos `sqrt` e `pow` da classe `java.lang.Math`.

5. Leia o raio de um círculo e imprima seu perímetro ( $2 \cdot \pi \cdot r$ ) e sua área ( $\pi \cdot r^2$ ).
6. Leia dois números inteiros e imprima a soma, a subtração, a multiplicação, a divisão e o resto da divisão entre esses dois números. (use funções). Observe que o valor do segundo número não deve ser 0. Isso deverá ser alertado, caso ocorra (use **if**).
7. Declare e inicialize (não é necessário ser informado pelo usuário) dois caracteres, dois pontos flutuantes e dois *strings* e verifique se os dois caracteres são iguais, os dois pontos flutuantes são iguais e se os dois *strings* são iguais. Procure ver que comparação de *string* não deve ser feita com o operador `==`. i.e, use o método `equals`.
8. Leia um número inteiro (nota) e se ela for entre 90 e 100 imprima “A”, entre 80 e 89 imprima “B”, entre 70 e 79 imprima “C”, senão imprima “Reprovado!”.
9. Leia um número inteiro e imprima o número resultante da soma de todos os números a partir de 1 até ele inclusive, isto é, se o número for 5, deverá retornar a soma de 1+2+3+4+5 que será 15. Se possível, exiba a expressão, por exemplo 1+2+3=6.
10. Crie um programa Java que imprima o número e o seu caractere da tabela ASCII correspondente em cada linha. Deverá ser feitos para números de 32 a 126.
11. Leia um *string* com uma frase de aproximadamente uns 30 caracteres (não precisa ser exatamente 30, é só para dizer que é um *string* grande) e indique o número de vezes que a letra ‘a’ foi encontrada nesse *string*.  
Dica: Entre na API Java e pesquise o `length` e `charAt` da classe `java.lang.String`.
12. Crie um arranjo de ponto flutuante de cinco posições. Leia do usuário os cinco salários. Depois imprima cada um dos valores e, em seguida, imprima todos os valores somados.
13. Crie um programa Java que dada as três variáveis da equação de 2º grau (a, b e c), imprima as raízes resultantes.

Fórmula de Baskara:

$$x = \frac{-b \pm \sqrt{\Delta}}{2 \cdot a}, \text{ sendo a diferente de 0.}$$

$$\Delta = b^2 - 4 \cdot a \cdot c$$

<p>Se <math>\Delta &lt; 0</math>, a parábola não toca o eixo x.</p> 	<p>Se <math>\Delta = 0</math>, só existe uma raiz.</p> 	<p>Se <math>\Delta &gt; 0</math>, existem duas raízes.</p> 
---	--	--

14. Simplifique o trecho código abaixo (use operador ternário):

```
int a = 40, c = 0;
if (a>20){
    c = 1;
} else {
    c = -1;
}
```

15. O trecho abaixo pode gerar erro? Se sim, como solucioná-lo sem mudar os valores de x e y?

```
int x = 4, y = 0;
if (y!=0 & x/y>2){
    System.out.println("Qual é o erro?");
}
```

16. Ciar um programa Java que:

- Crie uma variável do tipo **char** e peça para o usuário digitar um operador;
- Utilizando instruções **if** faça:
  - Se for + ou -, imprima para o usuário a mensagem: "Você digitou um operador de prioridade simples"
  - Se for / ou \*, imprima para o usuário a mensagem: "Você digitou um operador de prioridade alta"
  - Se não for nenhum destes, imprima a mensagem: "Pô!!! Você não digitou nenhum operador válido"

17. Refazer o programa acima utilizando **switch**.

18. Converta o código abaixo para utilizar **while**.

```
for (int i=0; i<10; i++){
    System.out.println(i);
}
```

19. Imprima todos os múltiplos de 3 ou de 5, entre 1 e 100 (inclusive):

- Dentro do método *main*, faça um for que varie de 1 a 100:

```
for (int i = 1; i <= 100; i++)
```
- Em cada iteração, verifique se *i* é múltiplo de 3 ou 5 (use o operador %):

```
if (i % 3 == 0 || i % 5 == 0)
```
- Se for múltiplo, então imprima o número.

20. Leia o seu nome e o exiba em caixa alta, em caixa baixa e imprima as iniciais de seu nome.

Dica: Entre na API Java e pesquise os métodos **toUpperCase**, **toLowerCase** e **charAt** da classe **String**.

21. Utilizar o **switch** para substituir o código abaixo:

```
char nota = 'C';
if (nota == 'A'){
    SOP("Nota " + nota);
}else if (nota == 'B'){
    SOP("Nota " + nota);
}else if (nota == 'C'){
    SOP("Nota " + nota);
}else{
    SOP("A nota " + nota + " não é uma nota válida;");
}
```

22. Criar um programa java que utiliza dois laços de repetição nos quais:

- No primeiro **for** a varredura é feita de 2008 a 2009.
- No segundo **for**:
  - varredura feita de 1 a 12
  - dentro de seu escopo terá um *switch* que caso seja 1 imprimirá Janeiro, 2 fevereiro, 3 março etc. E, logo em seguida, imprimirá o ano
- Saída esperada:

```
Janeiro/2008
Fevereiro/2008
Março/2008
...
Dezembro/2009
```

23. Imprima a seguinte tabela, usando fors encadeados.

```
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48 56 64
9 18 27 36 45 54 63 72 81
10 20 30 40 50 60 70 80 90 100
Dica de cada linha: n    n*2    n*3    ...    n*n
```

24. Crie um arranjo bidimensional de `int` com 4 linhas e 4 colunas posições e coloque em cada posição um número qualquer (não precisa ser informado pelo usuário). Depois faça um **for** varrendo todas as posições do arranjo e exibido os valores do arranjo conforme exemplo abaixo:

**Saída**

```
arranjo[0][0] = 20;
arranjo[0][1] = 23;
arranjo[0][2] = 60;
arranjo[0][3] = 11;
arranjo[1][0] = 1;
...
```

25. Crie um programa que declare e inicialize um arranjo de pontos flutuantes de precisão dupla (*double*) de 15 posições e imprima a soma total deste arranjo considerando a seguinte fórmula (Use: `for` ou `while`):

```
soma = 1 * array[0] + 2 * array[1] + ... + 14 * array[13] + 15 * array[14]
```

26. Declare e inicialize um arranjo bidimensional (matriz) de 3 linhas e 4 colunas e depois o imprima no formato de matriz, observe:

```
1    2    4    9
3    8    7    3
6    9    3    0
```

Dica: use '\t' e '\n'

27. Faça um arranjo bidimensional de *boolean* de 60 linhas e 80 colunas. Insira um valor *true* em uma posição qualquer. Após isto, faça um *for* varrendo o arranjo que quando encontrar um *true* sairá da iteração e exibirá: “Valor verdade encontrado na linha **X** e coluna **Y**”.

28. Crie um programa Java que, já declarado um arranjo de inteiros de **x** posições, faça a iteração dele. Quando o *loop* for encerrado deverá ser exibido em cada linha:

- o número de leituras realizadas (o tamanho do arranjo),
- o valor total da soma de todos os números lidos e
- a média aritmética dos números.

Ex.: Se o arranjo é 3, 6, 9. Seu tamanho é 3, soma igual a 18 e média igual a 6.

29. Crie um programa Java que dado um arranjo de caracteres (deve ser de 'a' até 'z') de **x** posições, imprima a posição, a letra e se ela é vogal ou consoante.

Ex.: Se o arranjo é 'a', 'f', 'e'. A saída seria:

1	a	vogal
2	f	consoante
3	e	vogal

30. Crie um programa Java que dada um *string* e informe ao usuário o tamanho do *string* digitado, o *string* original e o inverso deste *string*.

31. Crie um programa que leia um arranjo de cinco números inteiros e depois chame uma função que o imprima.

32. Crie um programa Java que leia um *string* e chame uma função que imprima cada uma das suas posições. Nessa função que imprime cada posição, deve ser chamada uma outra função que retorne **true** se for vogal e **false** se for consoante e, a partir do retorno, imprima se é a letra é vogal ou consoante (Utilize op. ternário).

Por exemplo, lido "ANA"

A » vogal

N » consoante

A » vogal

33. Crie um programa que leia um arranjo de cinco números inteiros e depois chame uma função que calcule a média e depois chame uma outra função que calcule a soma.

34. Crie um programa que leia um *string* e depois chame uma função que retorne a primeira posição do *string* em que foi encontrada uma vogal. Para saber se é vogal, deve ser criada uma outra função chamada **isVogal** que retorne **false** se não for vogal ou **true** se for vogal.

35. Crie um programa que possua uma função que calcule o fatorial de forma **iterativa**:

0! = 1

1! = 1

2! = 2 \* 1! = 2 \* 1 = 2

3! = 3 \* 2! = 3 \* 2 \* 1! = 3 \* 2 \* 1 = 6

36. Crie um programa que possua uma função que calcule o fatorial de forma **recursiva**:

37. Crie um programa que possua uma função que calcule o *fibonacci* de forma **recursiva**:

$$F(n) = \begin{cases} 0, & \text{se } n = 0; \\ 1, & \text{se } n = 1; \\ F(n-1) + F(n-2) & \text{outros casos.} \end{cases}$$

38. Crie um outro programa que, como na última questão, faça o cálculo do *fibonacci*, porém de forma **iterativa**.

Dica: Observe o desempenho para valores grandes de *n*. Suponha que faça com **n = 45**.

39. Crie um programa que leia um número e chame uma função que diz se o número é primo ou não. Um número primo é um número que somente é divisível por 1 e por ele mesmo.

40. Crie um programa que leia um número e chame uma função que diz se o número é um número perfeito. Um número perfeito um número inteiro para o qual a soma de todos os seus divisores positivos próprios (excluindo ele mesmo) é igual ao próprio número.

Por exemplo, 6 é divisível por 1, 2 e 3. Logo, é um número perfeito, pois 6 = 1 + 2 + 3.

Por exemplo, 28 é divisível por 1, 2, 4, 7 e 14. Logo, é um número perfeito, pois 28 = 1 + 2 + 4 + 7 + 14.