Linguagem de Programação Java

Tipos de Dados

- Os dados que usamos em um computador representam os elementos do mundo exterior, que são as informações manipuladas por nós seres humanos.
- Os dados utilizados no computador devem primeiramente ser abstraídos e então processados. Podem ser categorizados em três tipos primitivos ou básicos:
  - numéricos (representados por valores numéricos inteiros ou reais),
  - caracteres (valores alfabéticos ou alfanuméricos)
  - lógicos (valores falsos e verdadeiros).
- A linguagem Java nos oferece um conjunto de tipos de dados predefinidos, classificados em: numérico inteiro, numérico de ponto flutuante, caractere e logico.
- **Os tipos de dados são usados a fim de alocar espaço de memória para armazenamento e manipulação de valores de trabalho de um programa.**

Tipos de Dados

		Valores possíveis				
Tipos	***Primitivo***	***Menor***	***Maior***	***Valor Padrão***	***Tamanho***	***Exemplo***
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;

	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d;
						ou
						double ex6 = 10.20;

Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Variáveis

- **Toda informação a ser processada em um computador por um programa necessita ser previamente armazenada na memoria.**
- Conseguimos executar essa ação quando usamos variáveis. Uma variável é uma região de memoria, previamente identificada por um rotulo (nome), que tem por finalidade armazenar os dados de um programa temporariamente.
- **Cada variável armazena apenas um valor por vez**, sendo esse valor um elemento qualificado a partir de um dos tipos de dados da linguagem.
- O nome de uma variável é usado para sua identificação e posterior manipulação dentro do programa.

Exemplo Declaração Variável

```
int a;
int b = 10;
float num1 = 45;
float num2 = 45.6F;
double num3 = 67.8;
String nome = "Pedro Santos";
boolean resposta = true;
```

Operadores Aritméticos

- Os operadores aritméticos podem ser unários ou binários.
- São binários quando atuam em operações de multiplicação, divisão, adição e subtração, em que se utilizam dois componentes.
- São unários quando atuam na inversão de um valor, atribuindo a ele o sinal positivo ou negativo, ou seja, atuam diretamente em apenas um componente.

Operador Aritmético	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)

Operadores incrementais e decrementais

- Os operadores incrementais e decrementais têm a função de aumentar ou diminuir exatamente em 1 o valor de uma variável.

Operador Aritmético	Descrição
Pré incremental (++) ou prefixo	Significa que, se o sinal for colocado antes da variável, primeiramente será somado o valor 1 para esta variável, continuando em seguida a resolução da expressão.
Pós incremental (++) ou sufixo	Significa que, se o sinal for colocado após a variável, é resolvido primeiro a expressão, seja ela adição, subtração, multiplicação ou qualquer outra, para em seguida ser adicionado o valor 1 à variável.
Pré incremental (--) ou prefixo	Significa que, se o sinal for colocado antes da variável, primeiramente será subtraído o valor 1 para esta variável, continuando em seguida a resolução da expressão.
Pós incremental (--) ou sufixo	Significa que, se o sinal for colocado após a variável, é resolvido primeiro a expressão, seja ela adição, subtração, multiplicação ou qualquer outra, para em seguida ser subtraído o valor 1 à variável.

Operadores Aritméticos Reduzidos

- Os operadores incrementais e decrementais têm a função de aumentar ou diminuir exatamente em 1 o valor de uma variável.

Operador Aritmético	Descrição
+ =	mais igual
- =	menos igual
* =	vezes igual
/ =	dividido igual
% =	módulo igual

Operadores Relacionais

- Esses operadores são aplicados especificamente sobre comparações entre duas expressões numéricas ou variáveis de tipos primitivos numéricos.

Operador	Nome	Exemplo	Resultado
= =	Igual	x = = 10	
!=	Diferente	3 != 2	true
<	Menor	10 < 10	false
>	Maior	10 > 6	true
>=	Maior ou igual	3 >= 3	true
<=	Menor ou igual	7 <= 6	false

Atribuição

- Em Computação o comando de atribuição define ou redefine o valor armazenado no local de armazenamento indicado por um nome de variável.
- Na maioria das linguagens de programação imperativas o comando de atribuição é uma das declarações básicas.
- A instrução de atribuição muitas vezes permite que o mesmo nome de variável possa conter valores diferentes em momentos diferentes durante a execução do programa.

Programação Sequencial

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Estrutura de um Programa Java

- Antes de escrevermos um programa na linguagem Java é necessário que conheçamos sua estrutura mínima de operação e definição de código de programa, pois essa linguagem é do tipo case-sensitive. Isso significa que a linguagem diferencia caracteres maiúsculos de caracteres minúsculos, sendo necessário muita atenção e cuidado na codificação dos programas.
- A linguagem de programação se comunica com o computador segundo um formato sintático básico e próprio.
- As instruções de código podem ser formadas por um ou mais comandos, escritos em uma ou mais linhas.
- O final de uma instrução é indicado com o uso de um ponto e virgula.

Estrutura de um Programa Java

- Antes de escrevermos um programa na linguagem Java é necessário que conheçamos sua estrutura mínima de operação e definição de código de programa, pois essa linguagem é do tipo case-sensitive. Isso significa que a linguagem diferencia caracteres maiúsculos de caracteres minúsculos, sendo necessário muita atenção e cuidado na codificação dos programas.
- A linguagem de programação se comunica com o computador segundo um formato sintático básico e próprio.
- As instruções de código podem ser formadas por um ou mais comandos, escritos em uma ou mais linhas.
- O final de uma instrução é indicado com o uso de um ponto e virgula.

Estrutura de um Programa Java

```
public class programa2 {
    public static void Main (String[] args)
    {

    }
}
```

Abre chaves do Main: significa início do programa

O programa Java deve ser escrito neste local: entre as chaves do Main

Fecha chaves do Main: significa final do programa

Exemplo de um Programa Java

```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```

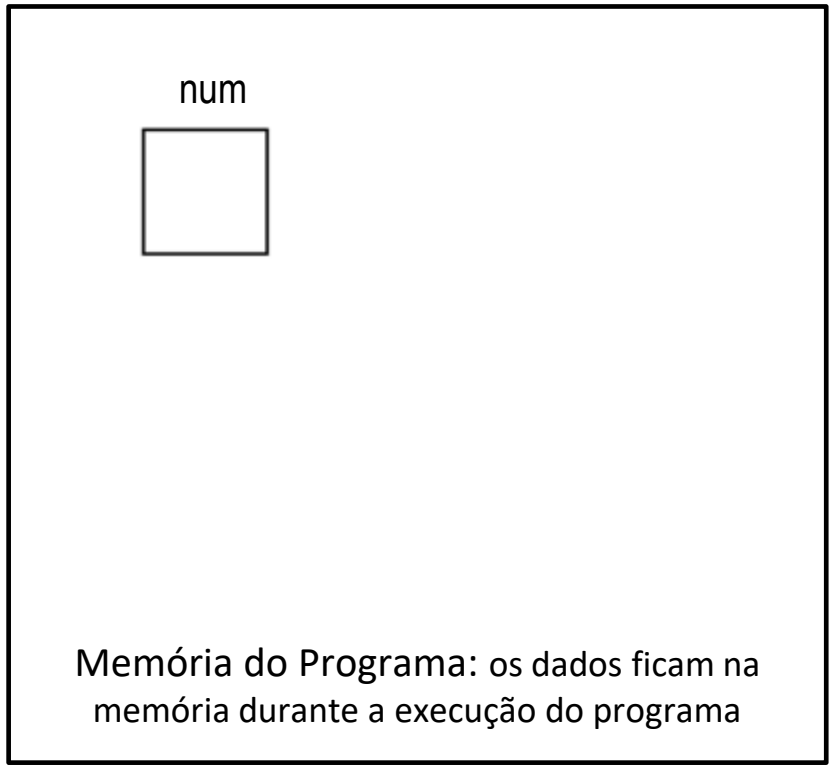
Exemplo de um Programa Java

```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```

Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

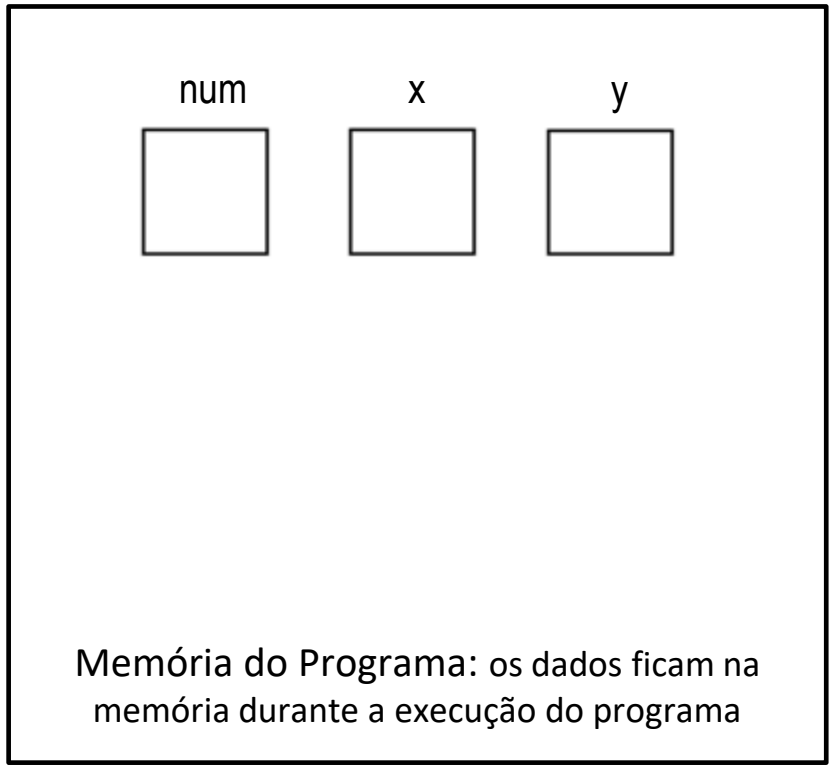
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

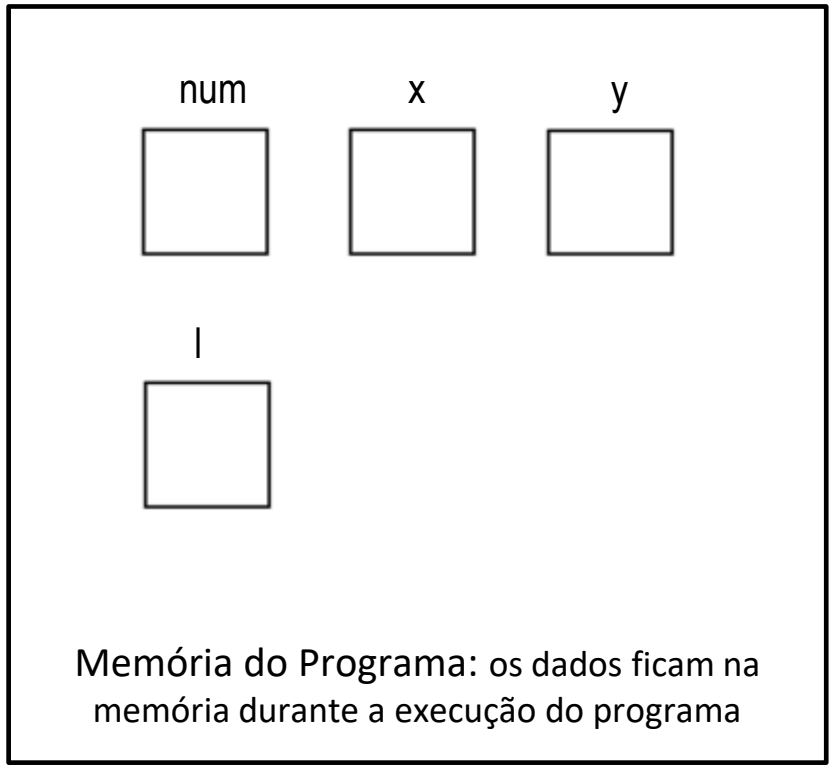
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

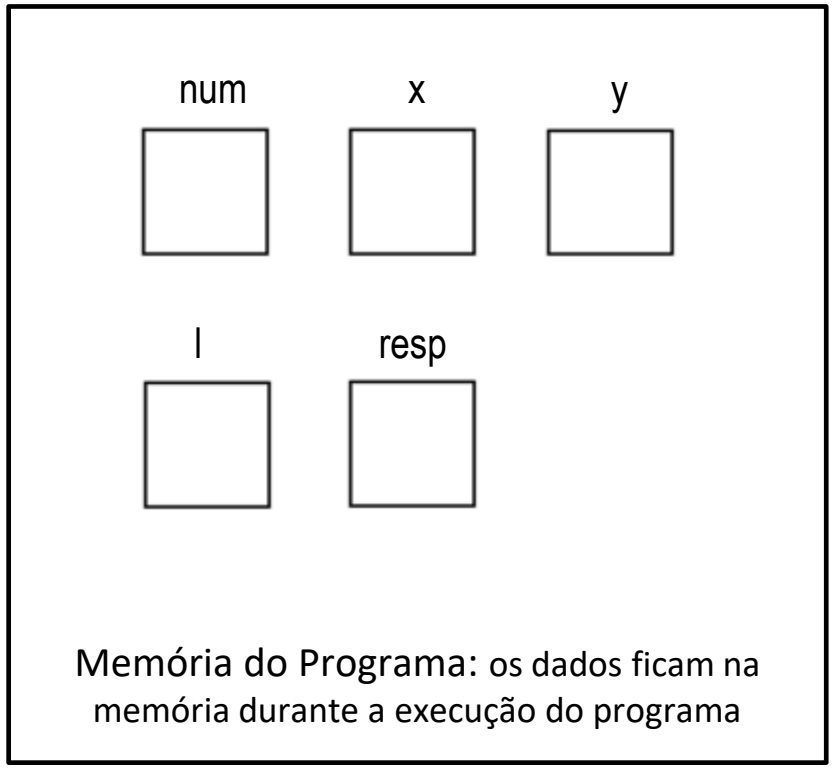
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

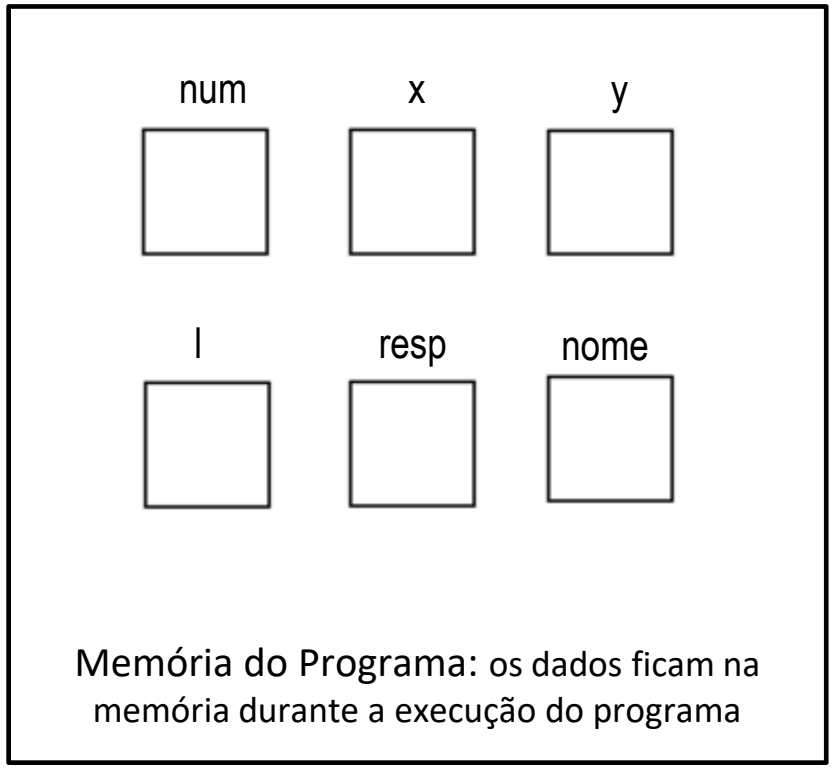
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

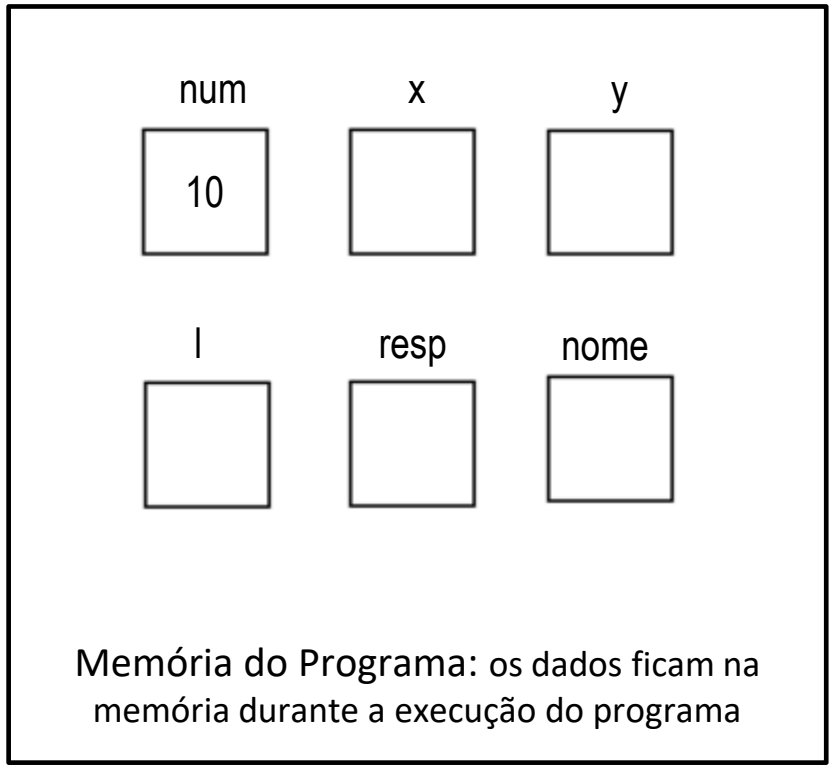
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

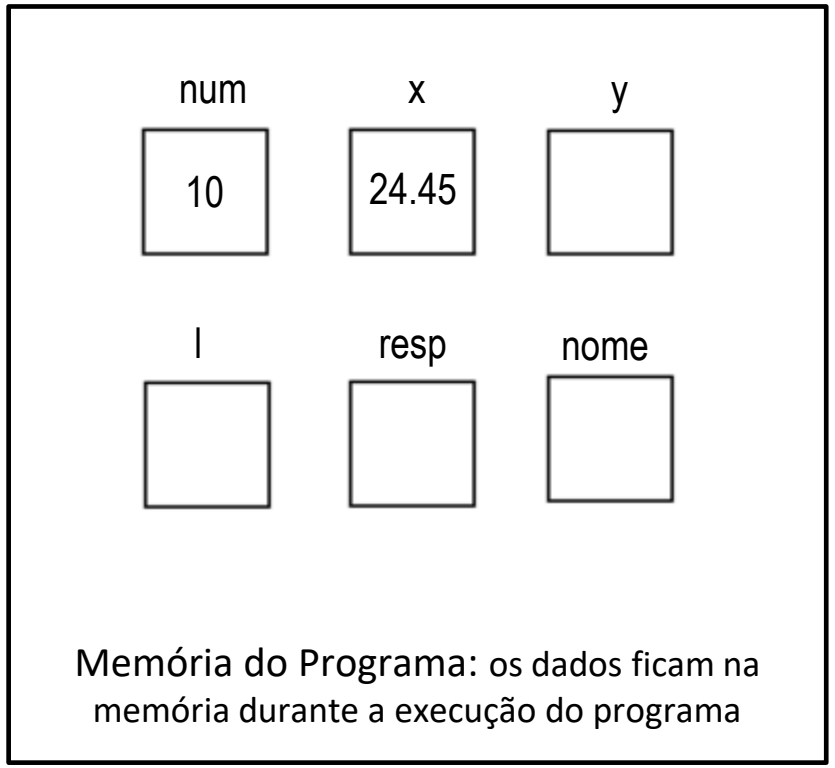
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

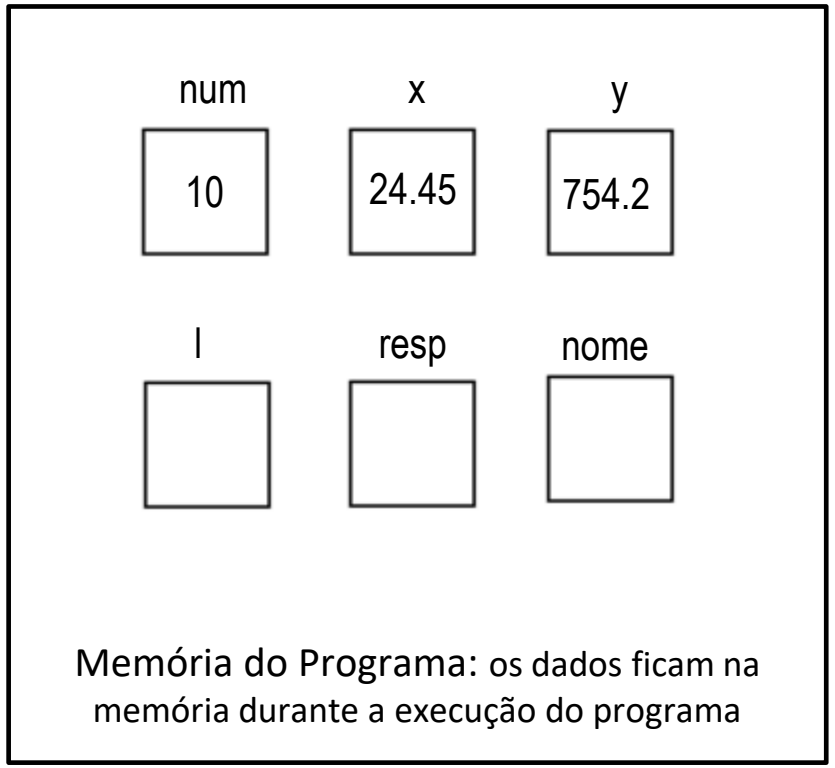
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

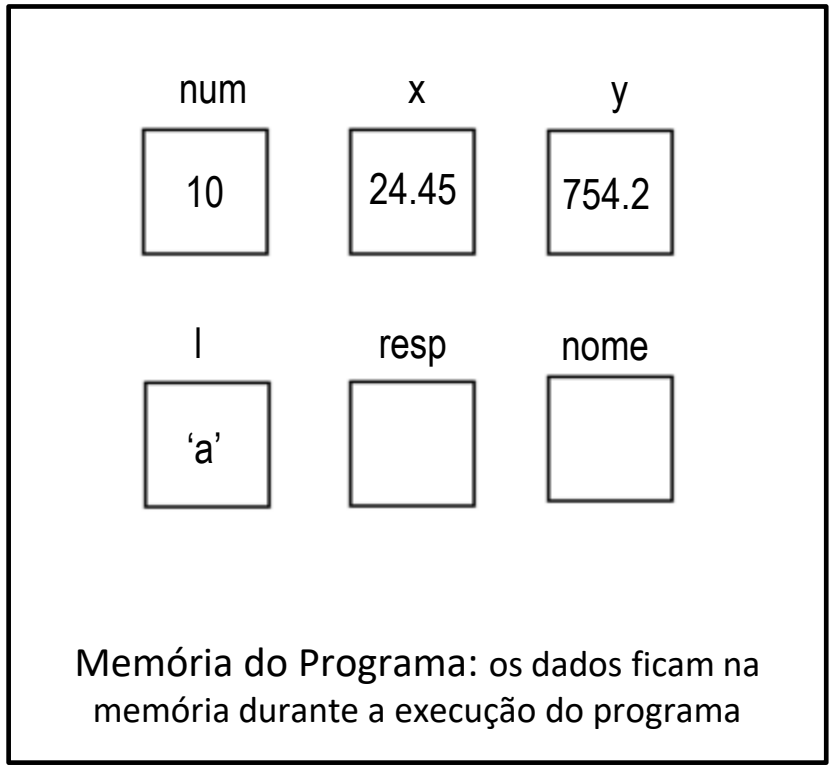
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

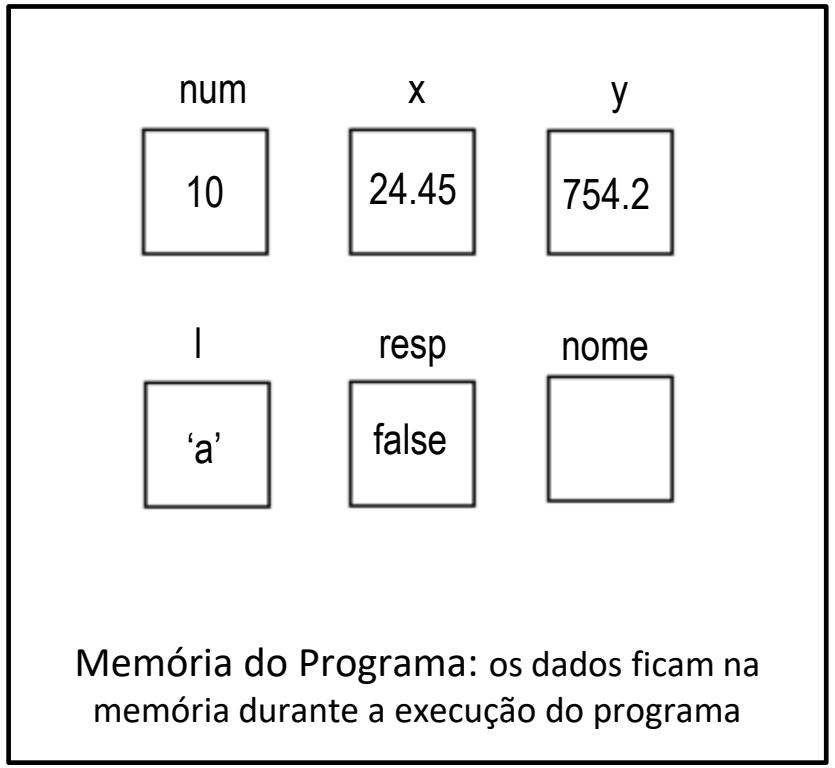
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

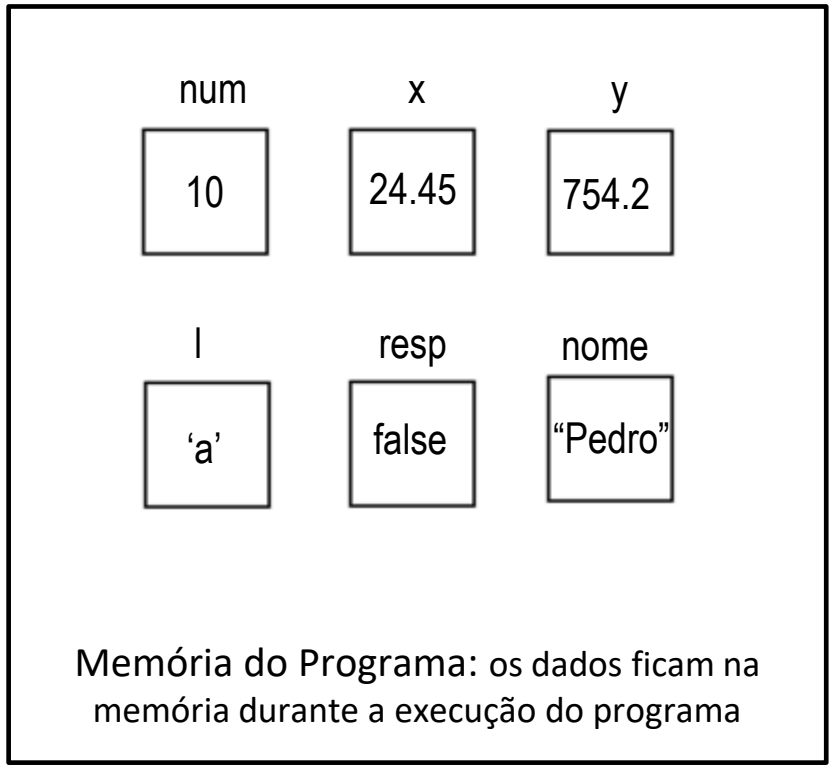
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

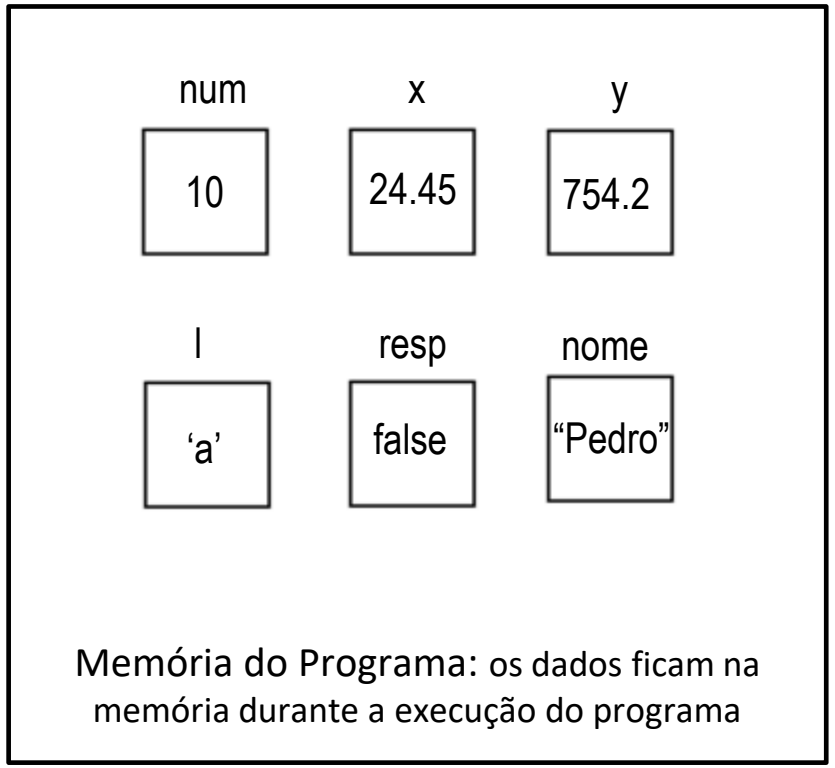
```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Exemplo de um Programa Java

```
public class programa2 {
    public static void Main (String[] args)
    {
        int num;
        double x, y;
        char l;
        boolean resp;
        String nome;
        num = 10;
        x = 24.45;
        y = 754.2;
        l = 'a';
        resp = false;
        nome = "Pedro";
    }
```



Memória do Programa: os dados ficam na memória durante a execução do programa

Entrada e Saída usando Console

```java
import java.util.Scanner;

public class programa3 {
    public static void main(String args[]) {

        Scanner ler = new Scanner(System.in);
        double area, lado1, lado2;

        System.out.printf("Digite um lado do retângulo: ");
        lado1 = ler.nextDouble();

        System.out.printf("Digite outro lado do retângulo: ");
        lado2 = ler.nextDouble();

        area = lado1 * lado2;
        System.out.printf("Área = %.2f\n", area);
    }
}
```

Exercícios

1. Fazer programa que calcule o valor da área de um triangulo, a partir do valor da base e altura.
2. Fazer programa que receba dois números e mostre a soma, a subtração, a multiplicação e a divisão dos números.
3. Fazer programa que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que esse vendedor ganha 20% de comissão sobre suas vendas efetuadas, faça que o algoritmo informe o seu nome, o salário fixo e salário no final do mês.
4. Fazer programa que leia o nome de um aluno e as notas das três provas que ele obteve no semestre, com pesos 2, 4 e 6. No final, deve-se informar o nome do aluno e a sua média ponderada.
5. Fazer programa que calcula a tensão elétrica (em Volts) onde o usuário entra com o valor da corrente elétrica (em Amperes) e o valor da resistência (em Ohms).

Programação Com Desvios

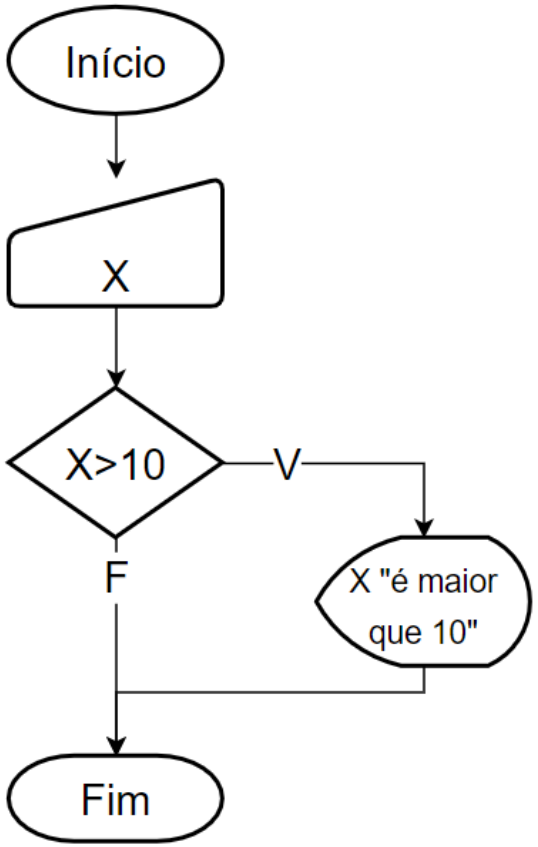
Comandos de Decisão

- Para um programa de computador tomar decisões, é necessário criarmos para ele uma condição.
- **Uma condição pode produzir uma de duas respostas: pode ser uma resposta verdadeira ou uma resposta falsa**.

- Desvio Condicional Simples: comando **if**
- Desvio Condicional Composto: comando **if... else**
- Desvio Condicional Seletivo: comando **switch .... case**

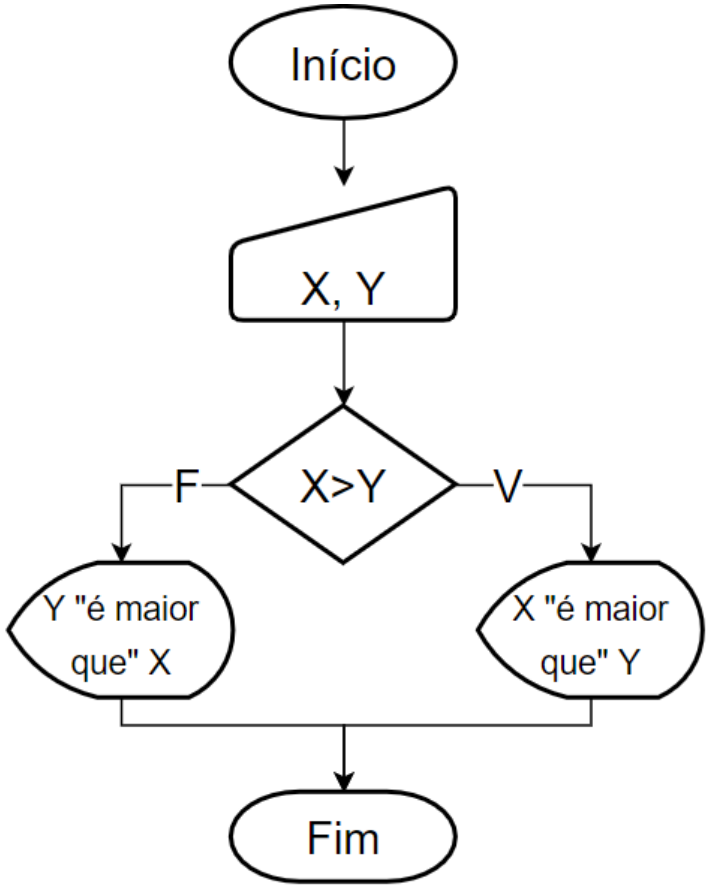
Comando de Decisão: IF

```
import java.util.Scanner;
public class programa4 {
    public static void main(String[] args)
    {
        Scanner entrada  = new Scanner(System.in);
        int x = entrada.nextInt();
        if (x>10)
        {
            System.out.println (x+ " é maior que 10\n");
        }
    }
}
```



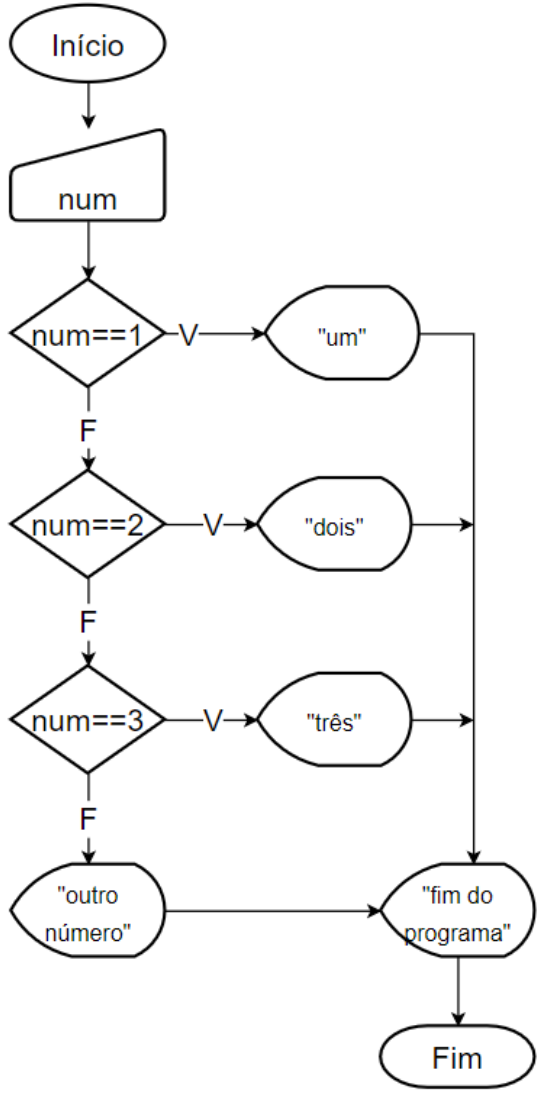
Comando de Decisão: IF...ELSE

```
import java.util.Scanner;
public class programa5 {
    public static void main(String[] args)
    {
        Scanner entrada  = new Scanner(System.in);
        int x = entrada.nextInt();
        int y = entrada.nextInt();
        if (x>y)
        {
            System.out.printf ("%d é maior que %d\n",x,y);
        }
        else
        {
            System.out.printf ("%d é menor que %d\n",x,y);
        }
    }
}
```



Comando de Decisão: SWITCH

```java
import java.util.Scanner;
public class programa6 {
    public static void main(String[] args)
    {
        Scanner entrada  = new Scanner(System.in);
        int num = entrada.nextInt();
        switch (num)
        {
            case 1: System.out.println("um");
                break;
            case 2: System.out.println("dois");
                break;
            case 3: System.out.println("três");
                break;
            default:
                System.out.println("outro numero");
                break;
        }
        System.out.println("fim do programa");
    }
}
```



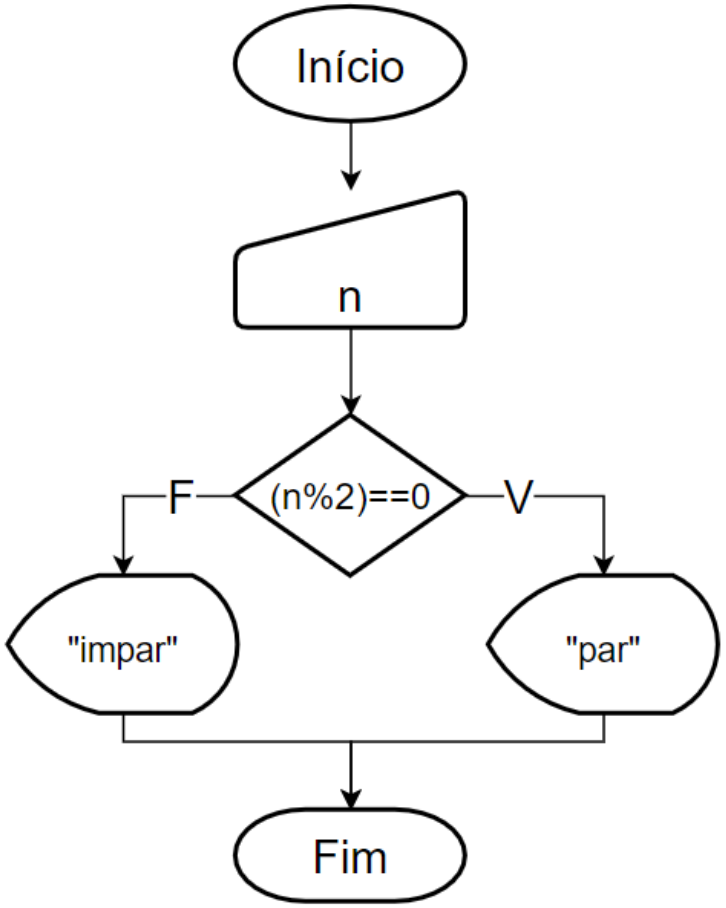
Divisibilidade

- Quando o resto de uma divisão de números naturais é igual a zero, tem-se divisibilidade, ou seja, resultado de divisão exata.
- A linguagem Java possui como operador aritmético para auxiliar o calculo de divisibilidade o símbolo % (porcentagem).
- Usamos esse operador aritmético quando necessitamos calcular o valor do resto de uma divisão de valores inteiros
- Exemplo:

```
X = 4%2;   // x é zero
Y = 5%3;   // y é 2
z = 6%3;   // z é zero
```

Divisibilidade

```
import java.util.Scanner;
public class programa7 {
    public static void main(String[] args)
    {
        Scanner entrada  = new Scanner(System.in);
        int n = entrada.nextInt();
        if (n%2 ==0)
        {
            System.out.println ("par");
        }
        else
        {
            System.out.println ("impar");
        }
    }
}
```



Exercícios

1. Fazer programa que receba um número e diga se ele está no intervalo entre 100 e 200.
2. Fazer programa que receba três inteiros e mostre qual deles é o maior e qual o menor.
3. Efetue a leitura de dois valores numéricos inteiros e apresente o resultado da diferença do maior valor pelo menor.
4. Fazer programa que receba um número inteiro do usuário e diga se ele é ou não múltiplo de 5.
5. Para doar sangue é necessário ter entre 18 e 67 anos. Elabore o programa que pergunte a idade de uma pessoa e diga se ela pode doar sangue ou não. Use alguns dos operadores lógicos OU (||) e E (&&).

Programação Com Repetição

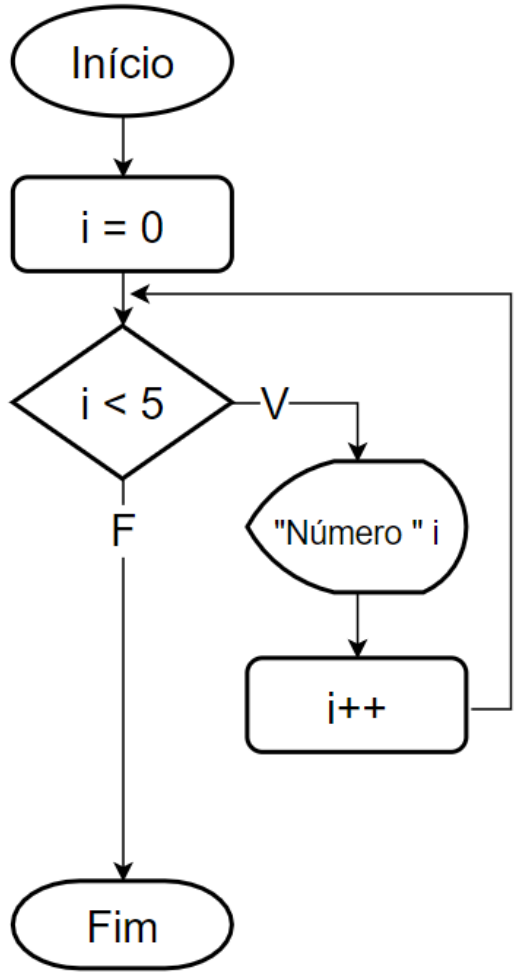
USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Comandos de Repetição

- A capacidade operacional de um computador é executar programas e que os programas são sequências de instruções que dão certa ordem de execução a um computador. Por vezes algumas dessas ordens devem ser repetidas , e para fazer esse tipo de ação temos a técnica de uso dos **laços de repetição**.
- Os laços que podemos usar com a linguagem Java são três:
  - Condição de controle no início do laço: WHILE
  - Condição de controle no fim do laço: DO ... WHILE
  - Laço com variável de controle: FOR

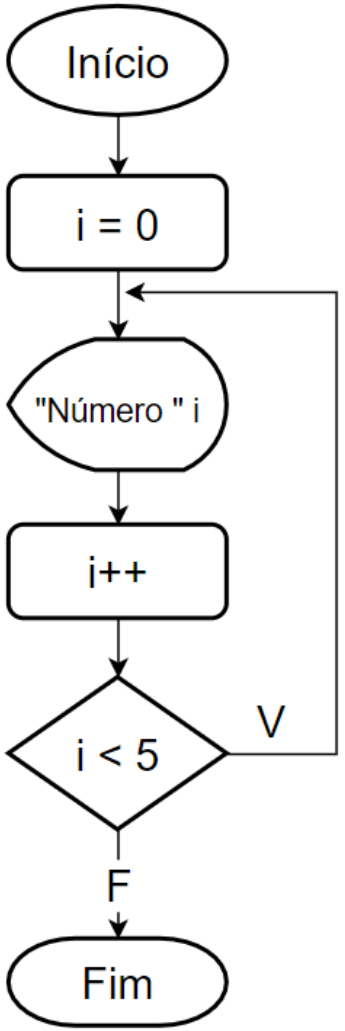
Comando de Repetição: WHILE

```
public class programa8 {
    public static void main(String[] args)
    {
        int i=0;
        while (i<5)
        {
            System.out.printf ("Numero: %d\n", i);
            i++;
        }
    }
}
```



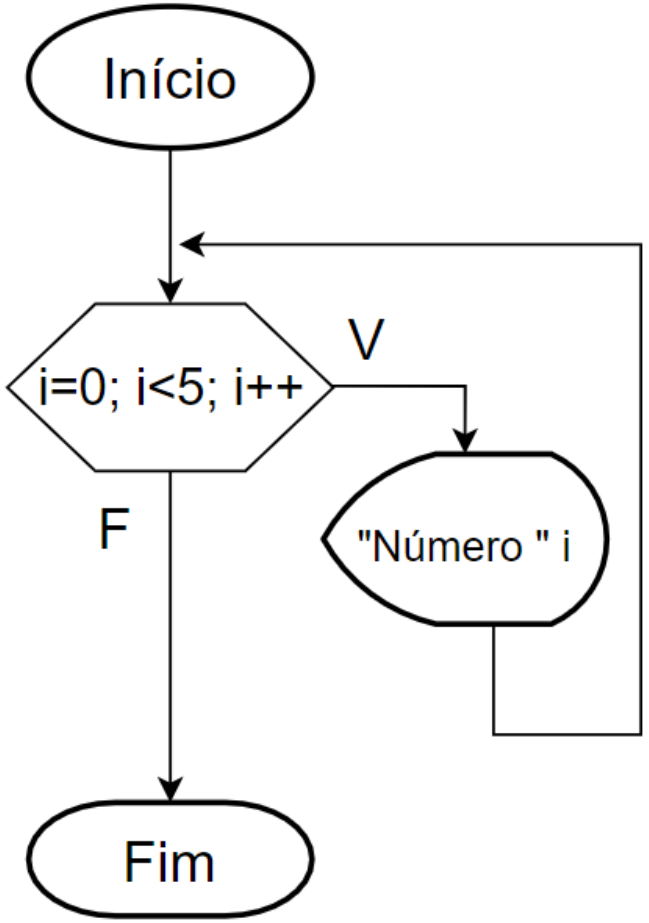
Comando de Repetição: DO WHILE

```
public class programa9 {
    public static void main(String[] args)
    {
        int i=0;
        do
        {
            System.out.printf ("Numero: %d\n", i);
            i++;
        }
        while (i<5);
    }
}
```



Comando de Repetição: FOR

```
public class programa10 {
    public static void main(String[] args)
    {
        int i;
        for (i=0; i<5; i++)
        {
            System.out.printf ("Numero: %d\n", i);
        }
    }
}
```



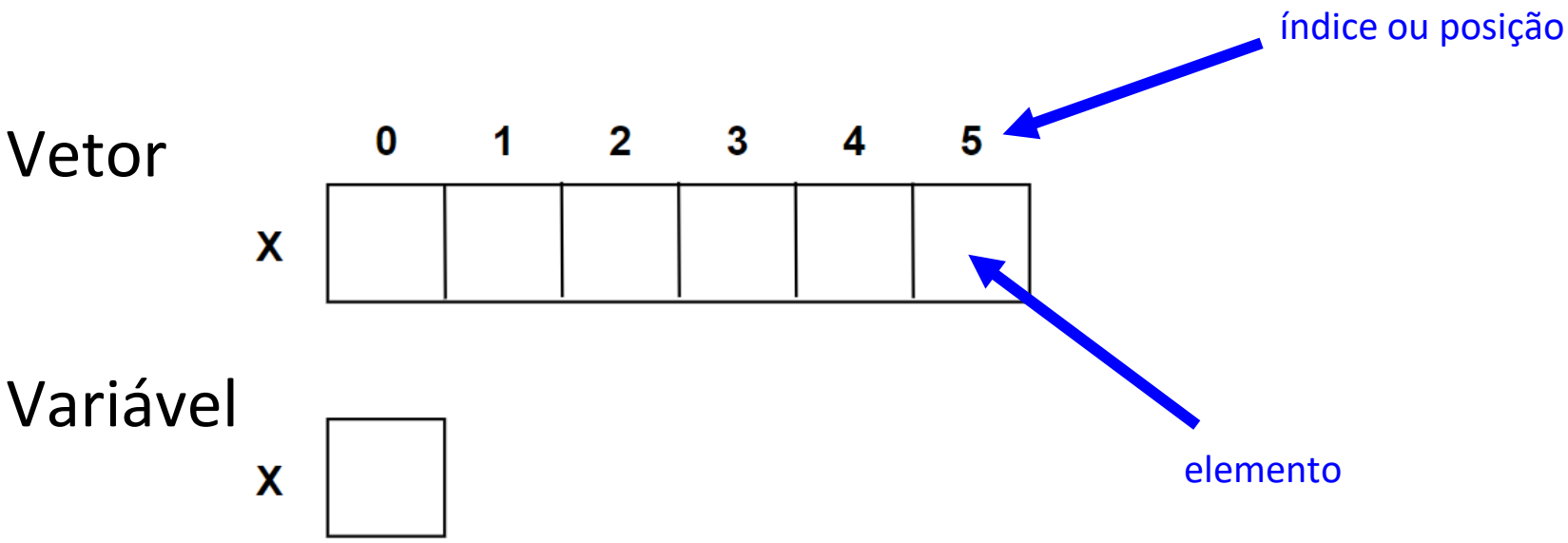
Exercícios

1. Desenvolver a codificação dos seguintes problemas a serem transformados em pro- gramas de computador. Utilize o laço while.
   a) Apresentar todos os valores numéricos inteiros impares situados na faixa de 0 a 20.
   b) Apresentar o total da soma dos cem primeiros números inteiros, representados pela sequencia 1+2+3+4+5+6+7+...+97+98+99+100.
   c) Apresentar todos os números divisíveis por 4 que sejam menores que 30. Iniciar a contagem com o valor 1 (um).
   d) Apresentar os quadrados dos números inteiros de 15 a 200.
   e) Apresentar o resultado da soma de todos os valores pares existentes na faixa numérica de 1 a 50.
2. Desenvolver a codificação dos problemas apresentados no exercício 1 com o laço do...while.
3. Desenvolver a codificação dos problemas apresentados no exercício 1 com o laço for.

Programação Com Vetores e Matrizes

Programando com Vetor

- Vetor é um tipo de estrutura de dados que armazena uma coleção de dados homogênea, ou seja, do mesmo tipo de dado.
- Um vetor pode conter vários valores, cada um em uma posição.

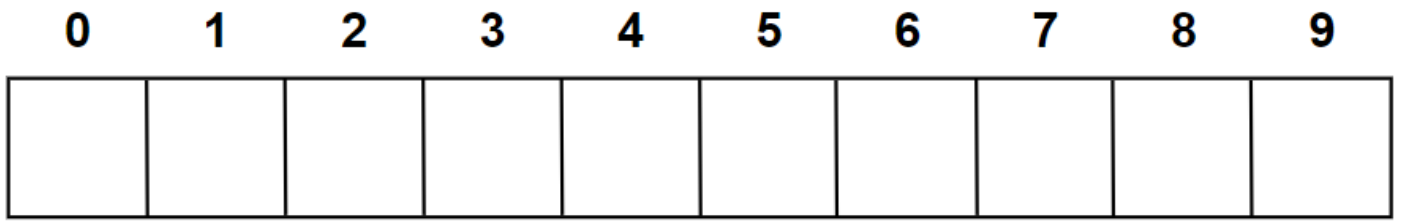


Exemplo com Vetor

```
public class programa10 {
    public static void main(String[] args)
    {
        int [] vetor = new int [10];
        vetor[0]= 76;
        vetor[1]= 22;
        vetor[2]= 10;
        vetor[3]= 5;
        vetor[4]= 36;
        vetor[5]= 67;
        vetor[5]= 89;
        vetor[7]= 92;
        vetor[8]= 15;
        vetor[8]= 28;
    }
}
```

Exemplo com Vetor

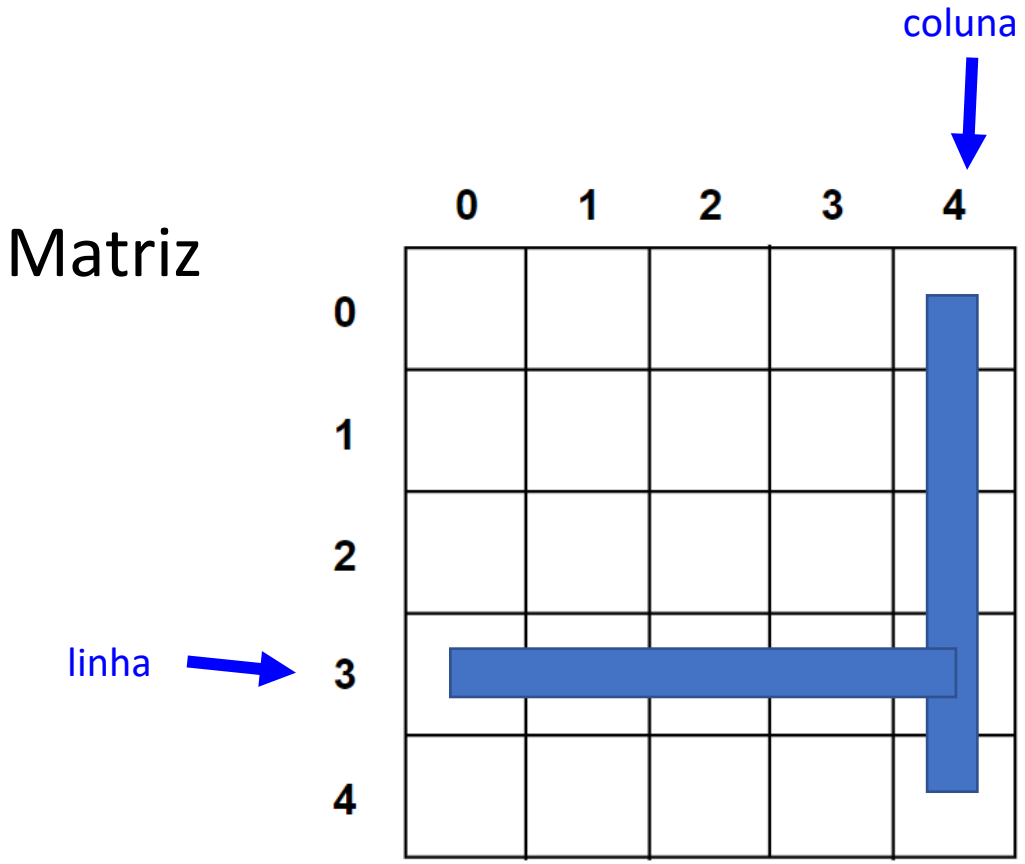
```
public class programa10 {
    public static void main(String[] args)
    {
        int [] vetor = new int [10];
        vetor[0]= 76;
        vetor[1]= 22;
        vetor[2]= 10;
        vetor[3]= 5;
        vetor[4]= 36;
        vetor[5]= 67;
        vetor[5]= 89;
        vetor[7]= 92;
        vetor[8]= 15;
        vetor[8]= 28;
    }
}
```



Programando com Matriz

- Além da forma que temos de organizar os dados em uma lista de valores como a que usamos no tópico anterior podemos **organizar os dados na forma de tabelas com matrizes de 2 dimensões**.
- Em matrizes com mais de uma dimensão, os elementos também são manipulados individualmente, com a referencia feita sempre por meio de **dois índices:**
  - **o 1º índice para controlarmos a linha;**
  - **o 2º índice para controlarmos a coluna.**

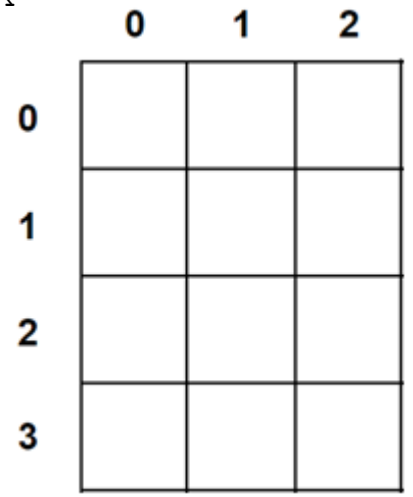
Programando com Matriz



Exemplo com Matriz

```
public static void main(String[] args)
{
    int [][] x = new int[4][3];
    x[0][0] = 35;
    x[0][1] = 26;
    x[0][2] = 12;
    x[1][0] = 52;
    x[1][1] = 74;
    x[1][2] = 27;
    x[2][0] = 93;
    x[2][1] = 28;
    x[2][2] = 49;
    x[3][0] = 44;
    x[3][1] = 60;
    x[3][2] = 98;
    for (int i=0; i<4;i++)
    {
        for (int j=0; j<3; j++)
            System.out.print (x[i][j] + " ");
        System.out.println("");
    }
}
```

X



Exemplo com Matriz

```
public static void main(String[] args)
{
    int [][] x = new int[4][3];
    x[0][0] = 35;
    x[0][1] = 26;
    x[0][2] = 12;
    x[1][0] = 52;
    x[1][1] = 74;
    x[1][2] = 27;
    x[2][0] = 93;
    x[2][1] = 28;
    x[2][2] = 49;
    x[3][0] = 44;
    x[3][1] = 60;
    x[3][2] = 98;
    for (int i=0; i<4;i++)
    {
        for (int j=0; j<3; j++)
            System.out.print (x[i][j] + " ");
        System.out.println("");
    }
}
```

i	3
j	0

X

	0	1	2
0	35	26	12
1	52	74	27
2	93	28	49
3	44	60	98

```
35 26 12
52 74 27
93 28 49
44
```

Exercícios

1. Colocar num vetor os números entre 1 e 10 e depois mostrar de forma invertida.
2. Colocar num vetor os números entre 1 e 10 e depois mostrar somente os pares.
3. Faça um programa que leia um vetor de números inteiros N[20]. A seguir, encontre o menor elemento do vetor N e a sua posição dentro do vetor, mostrando: “O menor elemento de N é ..., e sua posição dentro do vetor é:...”.
4. Faça um programa que, usando vetor, armazene as idades de quarenta alunos e exiba na tela a média de idade, a maior idade e a menor idade desses alunos.
5. Faça um programa que leia 10 números inteiros do teclado e os armazene num vetor. Depois, percorrer esse vetor mostrando os números ímpares.

Programação Com Métodos

Programando com Métodos

- **Um método é um trecho de código de programa independente de qualquer parte do programa**, mas relacionado ao programa com atribuições bem definidas.
- Os métodos são um conjunto de instruções que efetuam uma tarefa especifica. Os métodos também podem ser chamados de sub-rotinas.
- De forma geral, **os métodos podem receber valores de entrada (parâmetros opcionais)** e **gerar opcionalmente um valor de saída (retornar apenas um valor), denominado valor de retorno**.

Método com retorno

```
public class programa12 {
    public static void main(String[] args)
    {
        int a=10;
        int b=20;
        int resultado = Somar (a,b);
        System.out.printf ("Numero: %d\n", resultado);
    }

    public static int Somar(int x, int y)
    {
        int z = x + y;
        return z;
    }
}
```

Método sem retorno

```
public class programa12 {
    public static void main(String[] args)
    {
        int a=10;
        int b=20;
        Imprimir (a,b, a+b);
    }

    public static void Imprimir(int x, int y, int z)
    {
        System.out.printf ("%d + %d é igual à %d\n", x,y,z);
    }
}
```

Exercícios

1. Fazer método que recebe 2 números double e retorna a multiplicação dos números.
2. Fazer método que calcule o valor da área de um retângulo, a partir do valor da base e altura.
3. Fazer método que recebe 1 número double e retorna a raiz quadrada do numero
4. Fazer método que recebe 1 vetor de número inteiros e retorna a quantidade de números impares
5. Fazer método que recebe 1 vetor de número inteiros e retorna o vetor invertido