Usando estrutura de dados - Vetores

- Coleção de dados de **tamanho fixo**.
- Todos os elementos são do **mesmo tipo**.
- Acesso por **índice**, que começa em **0**.

DeclaracaoVetor.java

```java
public class DeclaracaoVetor {
    public static void main(String[] args) {
        // 1. Apenas declarando e instanciando (valores padrão)
        int[] numeros = new int[5]; // Vetor com 5 posições, todas valendo 0

        // 2. Declarando e inicializando com valores
        String[] nomes = {"Ana", "Carlos", "Beatriz"}; // Vetor com 3 posições

        System.out.println("Tamanho do vetor de números: " + numeros.length);
        System.out.println("Primeiro nome: " + nomes[0]);
    }
}
```

Acessando e Modificando Elementos

AcessoVetor.java

```java
public class AcessoVetor {
    public static void main(String[] args) {
        double[] notas = new double[4]; // Vetor para 4 notas
        notas[0] = 7.5; // Atribui valor ao primeiro elemento
        notas[1] = 8.0;

        System.out.println("Nota do 1º bimestre: " + notas[0]);

        // Modificando um valor
        notas[0] = 8.5;
        System.out.println("Nota corrigida do 1º bimestre: " + notas[0]);
    }
}
```

Acessando e Modificando Elementos

AcessoVetor.java

```java
public class AcessoVetor {
    public static void main(String[] args) {
        double[] notas = new double[4]; // Vetor para 4 notas
        notas[0] = 7.5; // Atribui valor ao primeiro elemento
        notas[1] = 8.0;

        System.out.println("Nota do 1º bimestre: " + notas[0]);

        // Modificando um valor
        notas[0] = 8.5;
        System.out.println("Nota corrigida do 1º bimestre: " + notas[0]);
    }
}
```

Percorrendo Vetores com Laços

LoopVetor.java

```java
public class LoopVetor {
    public static void main(String[] args) {
        double[] notas = {8.5, 9.0, 7.2, 10.0};
        double soma = 0.0;

        // Usando for para somar os valores
        for (int i = 0; i < notas.length; i++) {
            soma += notas[i];
        }

        double media = soma / notas.length;
        System.out.println("A média das notas é: " + media);
    }
}
```

Percorrendo Vetores com O Laço for-each (ou for Aprimorado)

`for` clássico, que vimos anteriormente, nos dá total controle sobre a iteração (início, fim, passo). No entanto, em muitas situações, nosso único objetivo é percorrer **todos** os elementos de um vetor ou coleção, do início ao fim. Para esses casos, o laço `for-each` é uma alternativa mais limpa e segura.

LoopVetorForEach.java

```java
public class LoopVetor {
    public static void main(String[] args) {
        double[] notas = {8.5, 9.0, 7.2, 10.0};
        double soma = 0.0;

        // Usando for-each para somar os valores
        for (double nota : notas) {
            soma += nota;
        }

        double media = soma / notas.length;
        System.out.println("A média das notas é: " + media);
    }
}

```

Estruturas de Dados: Matrizes

DeclaracaoMatriz.java

```java
public class DeclaracaoMatriz {
    public static void main(String[] args) {
        // Matriz 3×4 (3 linhas, 4 colunas)
        int[][] matriz = new int[3][4];
        matriz[0][0] = 15; // Acessa a primeira linha e primeira coluna

        // Inicialização direta
        String[][] agenda = {
            {"José", "9999-1111"}, // Linha 0
            {"Maria", "9999-2222"}  // Linha 1
        };

        System.out.println("Contato: " + agenda[0][0]);
        System.out.println("Telefone: " + agenda[0][1]);
    }
}
```

Matrizes: Iterando com Laços Aninhados

LoopMatriz.java

```java
public class LoopMatriz {
    public static void main(String[] args) {
        int[][] tabela = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

        // Laço externo (linhas)
        for (int i = 0; i < tabela.length; i++) {
            // Laço interno (colunas)
            for (int j = 0; j < tabela[i].length; j++) {
                System.out.print(tabela[i][j] + "\t"); // \t para tabular
            }
            System.out.println(); // Pula para a próxima linha
        }
    }
}
```

Quando falamos de processamento de dados por um computador, a entrada de dados são os dados obtidos de forma bruta, colhidos do mundo real através de algum dispositivo de entrada.

Exemplos:

- Teclado
- Arquivo
- Leitores
- Mouse
- Sensores

Para realizarmos entrada através do teclado podemos utilizar a classe **Scanner.** Essa classe possui vários métodos que possibilitam diferentes entradas de diferentes tipos:

```java
package exercicios;

import java.util.Scanner;

public class Exercicio {
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);
        int a, b;

        System.out.println("Informe o primeiro valor: ");
        a = ler.nextInt();

        System.out.println("Informe o segundo valor.: ");
        b = ler.nextInt();

        System.out.println("\nResultados da soma:\n");
        System.out.println(a + b);

        ler.close();
    }
}
```

Para entrada com a classe **Scanner** podemos utilizar de vários métodos para ler os diferentes tipo.

```java
package exercicios;

import java.util.Scanner;

public class Exercicio {
    public static void main(String[] args){
        Scanner ler = new Scanner(System.in);

        System.out.println("Informe um número inteiro:");
        System.out.println(ler.nextInt());

        System.out.println("Informe um valor em reais:");
        System.out.println(ler.nextDouble());

        ler.close();
    }

}
```

Para saída de dados formatada podemos utilizar o método **printf()**

System.out.printf(expressão_de_controle, argumento1, argumento2, ...);

```
public class TesteFormat {

    public static void main(String[] args) {
        System.out.println("Testando o printf!!");

    }

}
```

Formato	Tipo de Dados
%c	Caractere simples (char)
%s	Cadeia de caracteres (String)
%d	Inteiro decimal com sinal (int)
%i	Inteiro decimal com sinal (int)
%ld	Inteiro decimal longo (long)
%f	Real em ponto flutuante (float ou double)
%e	Número real em notação científica com o “e” minúsculo (float ou double)
%E	Número real em notação científica com o “E” maiúsculo (float ou double)
%%	Imprimir o próprio caractere %

No exemplo abaixo queremos imprimir o nome, idade e a altura de uma pessoa

“**Amaral** tem **50** anos e **1,55** de altura”,

onde os caracteres em destaque devem ser substituídos pelos dados do usuário.

Podemos fazer isso pondo indicadores de formato nas posições em que os dados devem ser impressos.
Desse modo, a string de formatação ficaria assim: “%s tem %d anos e %fm de altura”.

```
public class ExemploFormatacao {

    public static void main(String[] args) {
            String nome = "Amaral";
            int idade = 50;
            double altura = 1.55;

            System.out.printf("%s tem %d anos e %.2f de altura", nome, idade, altura);

    }
}
```

Uma outra forma de saída de dados é utilizando classe **JOptionPane** do pacote swing.

```
public class Exemplo3 {
    public static void main(String[] args) {
        String numero1 = JOptionPane.showInputDialog("Entre com o primeiro número");
        String numero2 = JOptionPane.showInputDialog("Entre com o segundo numero");
        Double nota1 = Double.parseDouble(numero1);
        Double nota2 = Double.parseDouble(numero2);
        JOptionPane.showMessageDialog(null, (nota1 + nota2)/2);

    }
}
```

1. Crie uma classe chamada `ListaDeCompras` que funcione como um assistente pessoal. O programa deve solicitar ao usuário, via `Scanner`, que digite 5 itens de supermercado. Armazene cada item em um vetor de `String` e, ao final, exiba a lista completa de forma organizada no console. ✓

2. Desenvolva uma classe `CalculadoraDeMediaVetor` para calcular a média de um aluno. Utilizando `Scanner`, o programa deve solicitar as 4 notas bimestrais, armazenando-as em um vetor de `double`. Após a inserção de todas as notas, calcule a média final e exiba o resultado formatado com duas casas decimais. ✓

3. Crie uma classe `MaiorNumero` que analise um conjunto de pontuações. Peça ao usuário para inserir 6 números inteiros, que podem representar pontuações de um jogo, usando `Scanner` ou `JOptionPane`. Armazene esses números em um vetor, processe os dados para encontrar a maior pontuação e, ao final, exiba o recorde encontrado. ✓

4. Construa uma classe `MontadorDeGrid` que permita ao usuário preencher um grid numérico de 3x3. O programa deve usar laços aninhados e `Scanner` para solicitar um número para cada célula da matriz. Após o preenchimento completo, exiba a matriz no console, formatada como uma grade com colunas bem alinhadas. ✓

5. Crie a classe `BoletimEscolar`, um sistema para gerenciar as notas da turma. Utilizando `JOptionPane` para uma interface mais interativa, o programa deve solicitar as 4 notas bimestrais para 3 alunos diferentes, armazenando tudo em uma matriz 3x4. Os prompts de entrada devem ser claros (ex: "Digite a 2ª nota do Aluno 1"). Ao final do preenchimento, o sistema deve calcular a média final de cada aluno (a média de cada linha da matriz) e exibir um relatório único com o resultado de todos.