



# PROGRAMAÇÃO ORIENTADA A OBJETO



# VETORES



# VETORES

## X Vetor

- Variável composta **unidimensional**
  - Contém espaço para armazenar diversos valores de um mesmo tipo
  - É acessada via um índice
- A ideia de vetor é comum na matemática, com o nome de variável subscrita
  - Exemplo:  $x_1, x_2, \dots, x_n$

## X Vetor

- Forma geral:

TIPO[ ] NOME = new TIPO[TAMANHO];

ou

TIPO[ ] NOME;

- Exemplos:

String[ ] nomes = new String[40];

float[ ] notas = new float[40];

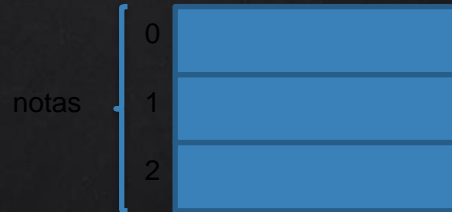
boolean[ ] presenca; // declara a variável como array

presenca = new boolean[5];

# VETORES

## X Vetor

- X É possível saber o tamanho de um vetor acessando a propriedade *length*
  - Exemplo: `notas.length` → 40
- X No Java, todo vetor inicia na posição 0 (zero) e termina na posição *length - 1*
  - Exemplo: `float[] notas = new float[3];`





# VETORES

## X Vetor

X Para acessar (ler ou escrever) uma posição do vetor, basta informar a posição entre colchetes

```
notas[0] = 8;
```

```
notas[1] = 5.5f;
```

```
notas[2] = 1.5f;
```

```
media = (notas[0] + notas[1] + notas[2]) / 3;
```

notas	0	8.0
	1	5.5
	2	1.5
media		5.0



# VETORES

## X Vetor

X Também é possível iniciar os valores de vetores diretamente no código, colocando-os entre chaves (`{}`), separados por vírgula

```
notas = { 8, 5.5f, 1.5f };
```

```
media = (notas[0] + notas[1] + notas[2]) / 3;
```





# VETORES

## X Vetor

X Iterar por todos os seus valores

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





# VETORES

## X Vetor

```
// declara variável vet como array e o inicializa com um objeto array  
int[] vet = new int[10]; // cria o objeto array
```

```
System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna
```

```
// gera saída do valor de cada elemento do array  
for (int counter = 0; counter < vet.length; counter++)  
    System.out.printf("%5d%8d%n", counter, vet[counter]);
```

Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0



# VETORES

## X Vetor

X Outra forma de iterar

```
for (parâmetro: nomeDoVetor) {  
    instrução;  
}
```

// declara variável **a** como array e o inicializa com um objeto array  
`int[] vet = new int[10];` // cria o objeto array

```
for (int nota: vet)  
    System.out.println(nota);
```

2.

# EXEMPLO CONTA CORRENTE



# VETORES

```
public static void main(String[] args) {  
    double[] contaCorrente = new double[5];  
    int numeroLancamento = -1;  
    double valorDeposito;  
  
    int op;  
    do {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("1 - Depositar Valor");  
        System.out.println("2 - Sacar Valor");  
        System.out.println("3 - Consultar Saldo");  
        System.out.println("4 - Consultar Histórico");  
        System.out.println("0 - Sair");  
        System.out.println("Entre com uma opção: ");  
        op = sc.nextInt();  
        switch (op) {  
            case 1:
```



EXEMPLOCONTACORRENTECLASSE



# VETORES

```
public static class InfoLancamento {
```

```
    public double valorDeposito;
```

```
    public String nomeDepositante;
```

```
};
```

Como um struct do C



```
public static void main(String[] args) {
```

```
    InfoLancamento[] contaCorrente = new InfoLancamento[5];
```

```
    int numeroLancamento = -1;
```

```
    double valorDeposito;
```

```
    String nome;
```

```
    InfoLancamento infoLancamento;
```



# ARRAYLIST





# ARRAYLIST

## X ArrayList

- A classe Java ArrayList usa uma estrutura dinâmica para armazenar os elementos.
- Semelhante a um array (mas, não há limite de tamanho) .
- É um Array dinâmico
- É possível adicionar ou remover elementos a qualquer momento.



# ARRAYLIST

## X ArrayList – características

- Pode conter elementos duplicados.
- Mantém a ordem de inserção.
- Permite acesso aleatório com base de índice.
- É necessário importar: **import java.util.ArrayList;**
- Forma Geral

```
ArrayList nomeDoObjeto = new ArrayList();  
ArrayList<tipo> nomeDoObjeto = new ArrayList<>();
```



# ARRAYLIST

## X ArrayList – características

- Não pode criar um ArrayList de tipos primitivos como int, char etc. Você precisa usar tipos box como Integer, Character, Boolean etc.

```
ArrayList<int> arrayList = new ArrayList<>(); // incorreto  
ArrayList<Integer> arrayList = new ArrayList<>();
```



# ARRAYLIST

## X ArrayList – operações típicas

- **add(element)**: para adicionar um novo elemento ao final desta lista
- **add(index, element)**: para adicionar um elemento no especificado index.
- **addAll(collection)**: para anexar outra coleção (por exemplo, em outra ArrayList).
- **set(index, element)**: substitui o elemento na posição especificada nesta lista pelo elemento especificado.
- **get(index)**: para obter um elemento no índice.
- **isEmpty()**: para verificar se há algum elemento nesta lista. Retorna verdadeiro se esta lista não contiver elementos.



# ARRAYLIST

## X ArrayList – operações típicas

- **size()**: retorna o número de elementos na lista.
- **clear()**: para excluir todos os elementos. Você também pode usar **removeAll()**: para excluir todos os elementos, mas é mais lento que **clear()**.
- **remove(index)**: remove o elemento no índice nesta lista. Desloca quaisquer elementos subsequentes para a esquerda (subtrai um de seus índices).
- **remove(object)**: remove a primeira ocorrência do elemento especificado desta lista.

5.

EXEMPLO ARRAYLIST





## EXEMPLO ARRAYLIST

```
public static void main(String[] args) {  
    // Creating an array list  
    ArrayList<Integer> numeros = new ArrayList<>();  
  
    // Inserir os valores  
    numeros.add(1);  
    numeros.add(2);  
    numeros.add(3);
```





EXEMPLOCONTACORRENTECLASSEARRAYLIST



# EXEMPLO CONTACORRENTECLASSEARRAYLIST

## X Como vetor


```
public static void main(String[] args) {  
    InfoLancamento[] contaCorrente = new InfoLancamento[5];  
    int numeroLancamento = -1;  
    double valorDeposito;  
    String nome;  
    InfoLancamento infoLancamento;
```

## X Como arrayList

```
public static void main(String[] args) {  
    ArrayList<InfoLancamento> contaCorrente = new ArrayList<>();  
    double valorDeposito;  
    String nome;  
    InfoLancamento infoLancamento;
```



STRING



# STRING

## X Características

- Em Java, as strings são objetos da classe `java.lang.String`;
- Um objeto `String` armazena uma sequência de caracteres.
- Criação de uma `String`

```
String nome = new String("Ana");
```

ou

```
String sobrenome = "Ana";
```



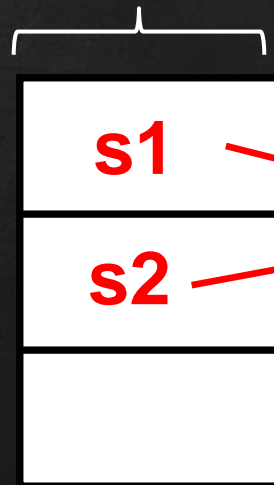
# STRING

## X Características

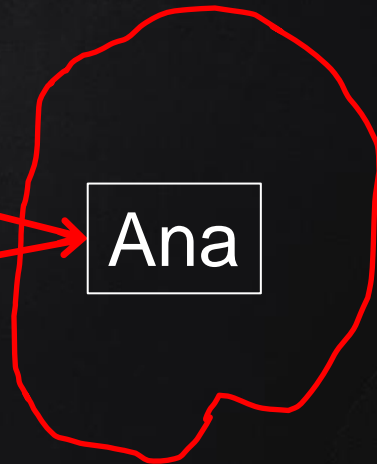
```
String s1 = "Ana";  
String s2 = "Ana"
```

Se a string já existir no pool, uma referência à instância do pool será retornada. Se a string não existir no pool, uma nova instância de string será criada e colocada no pool.

*Pilha de execução*



Heap



# STRING

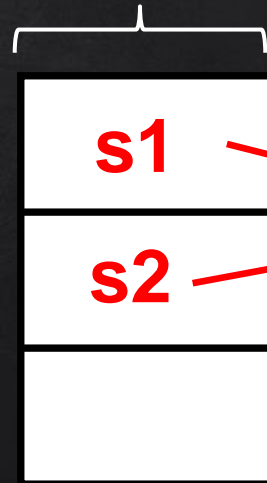
## X Características

```
String s1 = "Ana";  
String s2 = "Ana"
```

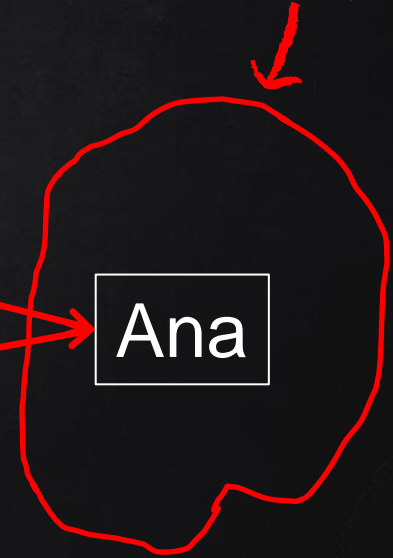
```
if (s1==s2) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferente");  
}
```

```
run:  
iguais  
CONSTRUIDO COM SUCESSO (tempo total: 0 segundos)
```

Pilha de  
execução



Heap



# STRING

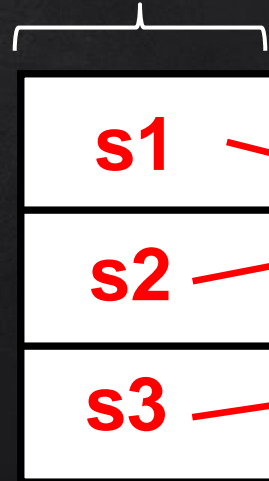
## X Características

```
String s1 = "Ana";  
String s2 = "Ana"
```

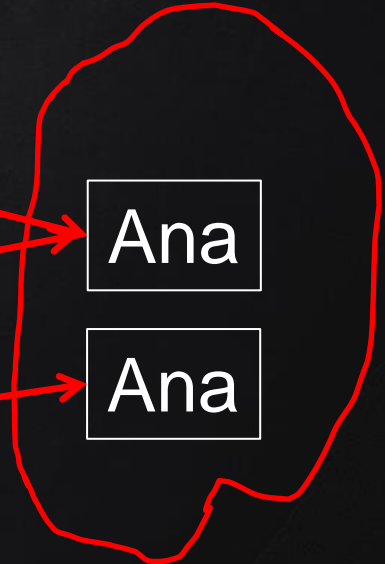
```
String s3 = new String("Ana");  
if (s1 == s3) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferente");  
}
```

```
run:  
diferente  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Pilha de  
execução



Heap







# STRING

## X Características

- Não deveríamos estar preocupados em saber se as strings estão armazenadas no mesmo espaço de memória e sim se os caracteres são iguais.
- Usar o método **Equals**

```
String s4 = new String("Ana");  
if (s1.equals(s4)) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferente");  
}
```

```
String s4 = "Ana";  
if (s1.equals(s4)) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferente");  
}
```

```
run:  
iguais  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```



## X Características

- O valor de uma String é imutável, não se pode alterar seu valor.
- Quando o valor é alterado uma nova String é criada.
- Concatenação

```
String nomeNovo = "Ana" + "Maria";
```



# STRING

## X Características

- Alguns caracteres necessitam que sejam utilizados a barra invertida (\) para serem colocados dentro a string (escape de caracteres);

```
String aux = "''"; // erro de compilação
```

```
String aux = "\'";
```



# STRING

## X Características

- Em Java, há um total de oito sequências de escape:
  - `\t` É usado para inserir uma tabulação no texto neste ponto.
  - `\'` Ele é usado para inserir um caractere de aspas simples no texto neste ponto.
  - `\"` Ele é usado para inserir um caractere de aspas duplas no texto neste ponto.
  - `\r` Ele é usado para inserir um retorno de carro no texto neste ponto.
  - `\\` Ele é usado para inserir um caractere de barra invertida no texto neste momento.
  - `\n` É usado para inserir uma nova linha no texto neste ponto.
  - `\f` Ele é usado para inserir um feed de formulário no texto neste momento.
  - `\b` Ele é usado para inserir um backspace no texto neste ponto.



# STRING

## X Características

- A classe String possui diversos métodos.
  - int **length**(): Retorna o tamanho da string, ou seja, a quantidade de caracteres da string;
  - char **charAt**(int i): Retorna o i-ésimo caractere da string. Obs: assim como nos vetores a posição do primeiro caractere de uma string é igual a 0 (zero).



PROCESSASTRING