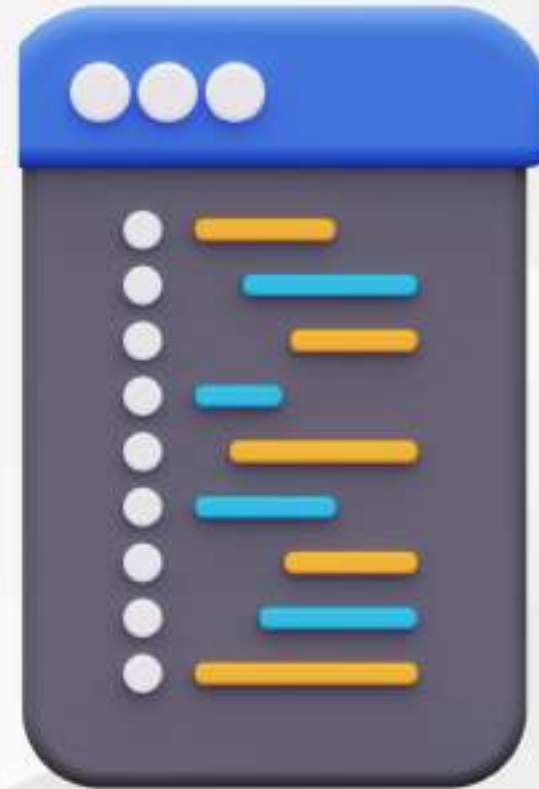


POO

Programação Orientada a Objetos

Material 004

Professor Maromo





Agenda

Outros Recursos da Linguagem

- Associações
- Vetores em Java e Matrizes em Java
- Comando package
- Comando import
- Passagem de Parâmetros (valor/referência)
- Exercícios



Associações de Classes



A associação de classes indica quando uma classe tem um tipo de relacionamento "tem um" com outra classe como, por exemplo, uma pessoa tem um carro e isso indica que a classe Pessoa tem uma associação com a classe Carro.

Esse tipo de relacionamento entre classes é importante, porque define **como as classes interagem entre elas nas aplicações.**

Associação



- A Listagem no próximo slide apresenta uma implementação de uma associação um para um.
- A associação é feita entre as classes **Pessoa** e **Carro**.
- A classe **Carro** tem os atributos **modelo, placa, ano e valor**; e a classe **Pessoa** tem os atributos **nome, endereço, telefone e dataNascimento**. Além disso, **ela tem um relacionamento com a classe Carro**.
- Esse relacionamento é conhecido por **agregação**.

```
public class Carro {
```

```
    private String modelo;  
    private String placa;  
    private int ano;  
    private float valor;
```

```
    // Métodos getters and setters suprimidos
```

```
}
```

```
import java.util.Date;
```

```
public class Pessoa {
```

```
    private String nome;  
    private String endereco;  
    private String telefone;  
    private Date dataNascimento;
```

```
    // Relacionamento com a classe Carro
```

```
    private Carro carro;  
    // Métodos getters and setters suprimidos
```

```
}
```



Associação e cardinalidade



- Além do relacionamento um para um, também existem o relacionamento **um** para **N** e **N** para **N**.
- Esses relacionamentos são modelados com um vetor de objetos de uma classe para outro como, por exemplo, se uma **pessoa pode ter mais de um carro**, é necessário mapear esse relacionamento com uma lista de carros na classe Pessoa. A Listagem no próximo slide mostra a alteração necessária na classe **Pessoa**, mas na classe **Carro não é necessária nenhuma alteração**.



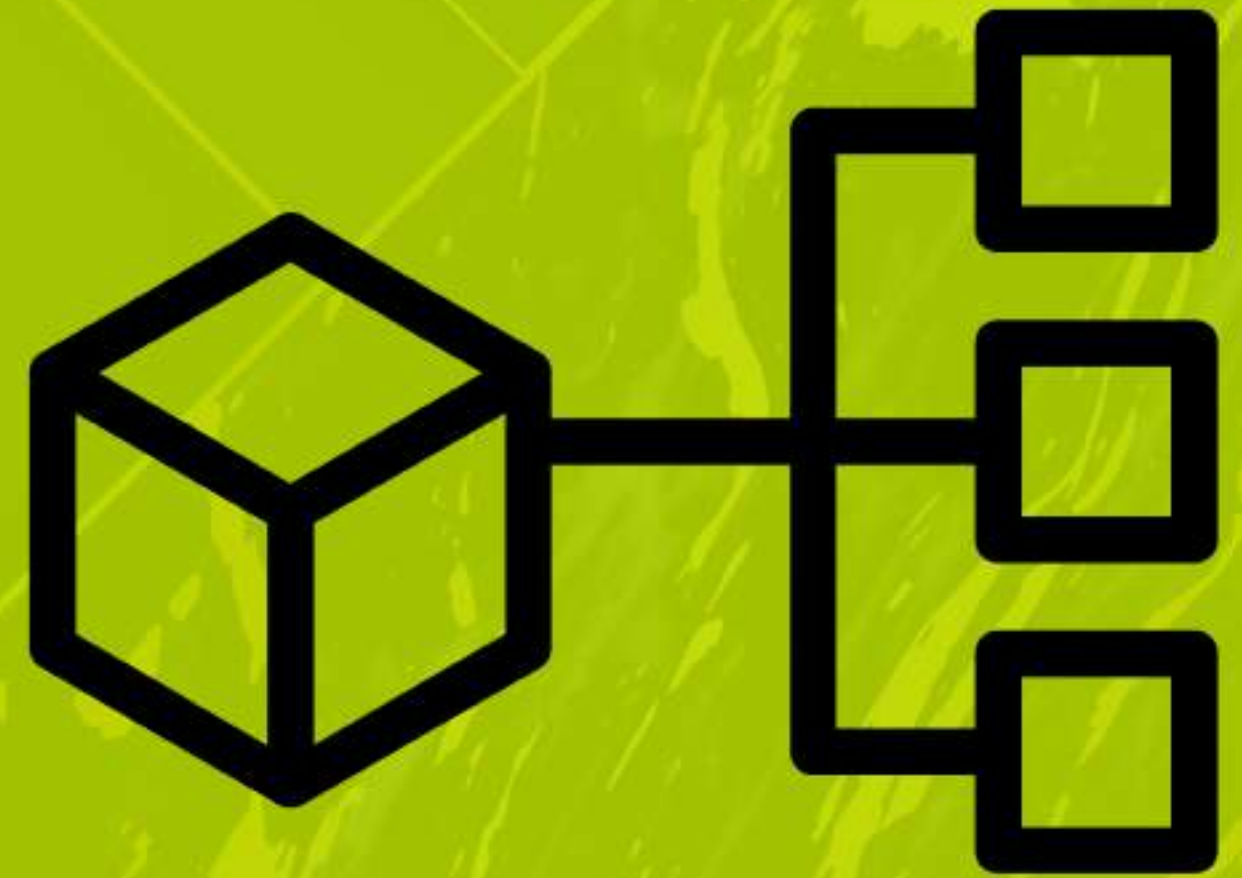
```
public class Pessoa {
```

```
    private String nome;  
    private String endereco;  
    private String telefone;  
    private Date dataNascimento;
```

```
    // Relacionamento com a classe Carro  
    private List<Carro> carros = new ArrayList<Carro>();
```

```
}
```





Arrays em java

Vetores [Arrays]



- Podem armazenar qualquer tipo de dado.
- Podem armazenar objetos (na verdade referências a objetos).
- Seu índice inicial é 0.
- Podem ser multidimensionais (matrizes).

Array - Sintaxe



- O nome de uma variável array segue os mesmos padrões das demais variáveis em Java seguido de um par de colchetes [].
- Declarando um Array:

```
String vetNome[] = new String[5];
```

```
Cliente vetCliente[] = new Cliente[12];
```

```
int idades[] = { 25,39,45,58};
```

```
String casais[][]=new String[3] [2];
```


Array – Atribuição de Valores



```
vetNome[0]="Paula";  
vetCliente[8]=cli;  
Idades[] = {1,5,8,12};  
casais[1][1]="Maria";  
casais[][]={{"José","Maria"},"Barth","Indira"},"Aquira","Paola"}};  
vetCliente[1].setNome("João");
```

Array – Leitura de Valores



```
System.out.println(vetCliente[1].getNome());
```


Array

Percorrer pelo comando for



```
for(int i=0;i<9;i++){  
    System.out.println(vetNome[i]);  
}
```

```
for(int i=0; i <= vetNome.length - 1; i++) {  
    System.out.println(vetNome[i]);  
}
```

Exemplo: ProjetoExemplo

Classe: ExemploVetor



```
public class ExemploVetor {  
    public static void main(String[] args) {  
        //Atribuição direta  
        int[] idade = {21, 18, 16, 24, 19};  
        for (int i = 0; i < idade.length; i++) {  
            System.out.printf("Idade do elemento %d = %d \n", i+1, idade[i]);  
        }  
    }  
}
```

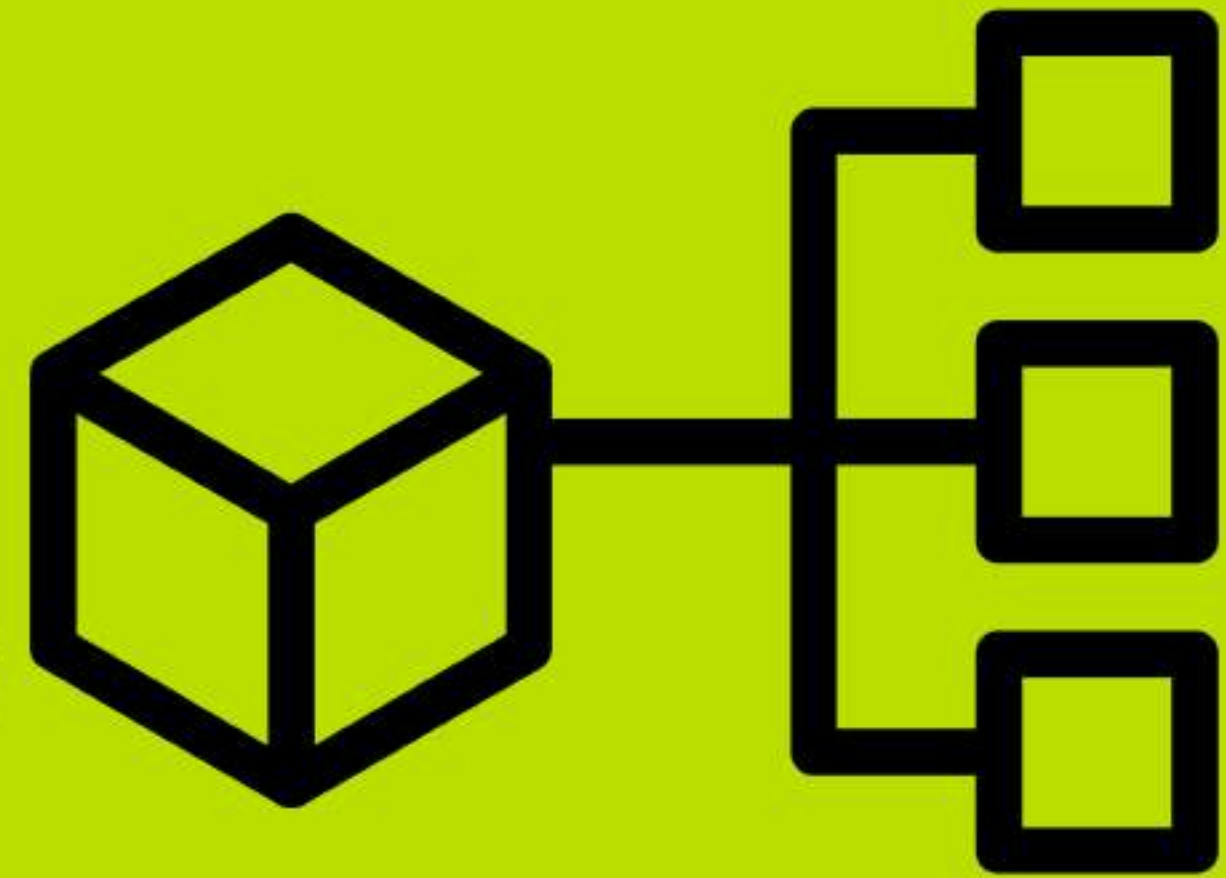
Idade do elemento 1 = 21
Idade do elemento 2 = 18
Idade do elemento 3 = 16
Idade do elemento 4 = 24
Idade do elemento 5 = 19

Array – Atributo length

- Sua função é retornar o número de elementos do vetor.

Array – Representação na memória

```
double[] myList = new double[10];
```



myList

referência

myList[0]
myList[1]
myList[2]
myList[3]
myList[4]
myList[5]
myList[6]
myList[7]
myList[8]
myList[9]


```
import java.util.Scanner;
```

```
public class ExemploVetor2 {
```

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

```
        Scanner sc = new Scanner(System.in);
```

```
        int[] var1 = new int[10];
```

```
        System.out.println("Informe 10 valores: ");
```

```
        for (int i = 0; i < var1.length; i++) {
```

```
            System.out.println("Elemento " + (i+1) + ": ");
```

```
            var1[i] = Integer.parseInt(sc.nextLine());
```

```
        }
```

```
        int soma = 0;
```

```
        for (int elemento : var1){
```

```
            soma+= elemento;
```

```
        }
```

```
        double media = soma / (double) var1.length;
```

```
        System.out.println("Media: " + media);
```

```
    }
```

```
}
```

Exemplo: ExemploVetor2

Informe 10 valores:

Elemento 1:

1

Elemento 2:

2

Elemento 3:

3

Elemento 4:

4

Elemento 5:

5

Elemento 6:

6

Elemento 7:

7

Elemento 8:

8

Elemento 9:

9

Elemento 10:

10

Media: 5.5

Resultado do Exemplo:



Não é possível alocar elementos nesse tipo de vetor além do limite dado na sua criação.

Método System.arraycopy

ExemploArrayCopy

```
public class ExemploArrayCopy {  
    public static void main(String[] args) {  
        int[] vet = {1, 2, 3, 4};  
        int[] vet2 = new int[10];  
        System.arraycopy(vet, 0, vet2, 6, 4);  
        for (int elemento: vet2) {  
            System.out.println(elemento);  
        }  
    }  
}
```

No exemplo acima, na Linha 6 `System.arraycopy(vet, 0, vet2, 6, 4)` temos:
`vet` é o nome do vetor de origem. 0 é posição inicial da cópia do vetor de origem.
`vet2` é o nome do vetor de destino, 6 é a posição inicial da cópia de destino e 4 é o tamanho de elementos a serem copiados da origem para o destino.
Resultado, no próximo slide.

Método System.arraycopy



0

0

0

0

0

0

1

2

3

4

Método Arrays.fill

Tem a finalidade de preencher os elementos de um vetor com um determinado valor. Para isso, deve-se importar o pacote `java.util.Arrays`.

```
import java.util.Arrays;

public class ExemploArraysFill {
    public static void main(String[] args) {
        int[] vetor = new int[5];
        Arrays.fill(vetor, 31);
        for (int valor: vetor) {
            System.out.println(valor);
        }
    }
}
```


Arrays de Referência



- Quando criamos um array de uma classe, ela possui referências. O objeto, está na memória principal e, no array ficam guardadas as referências. [endereços]

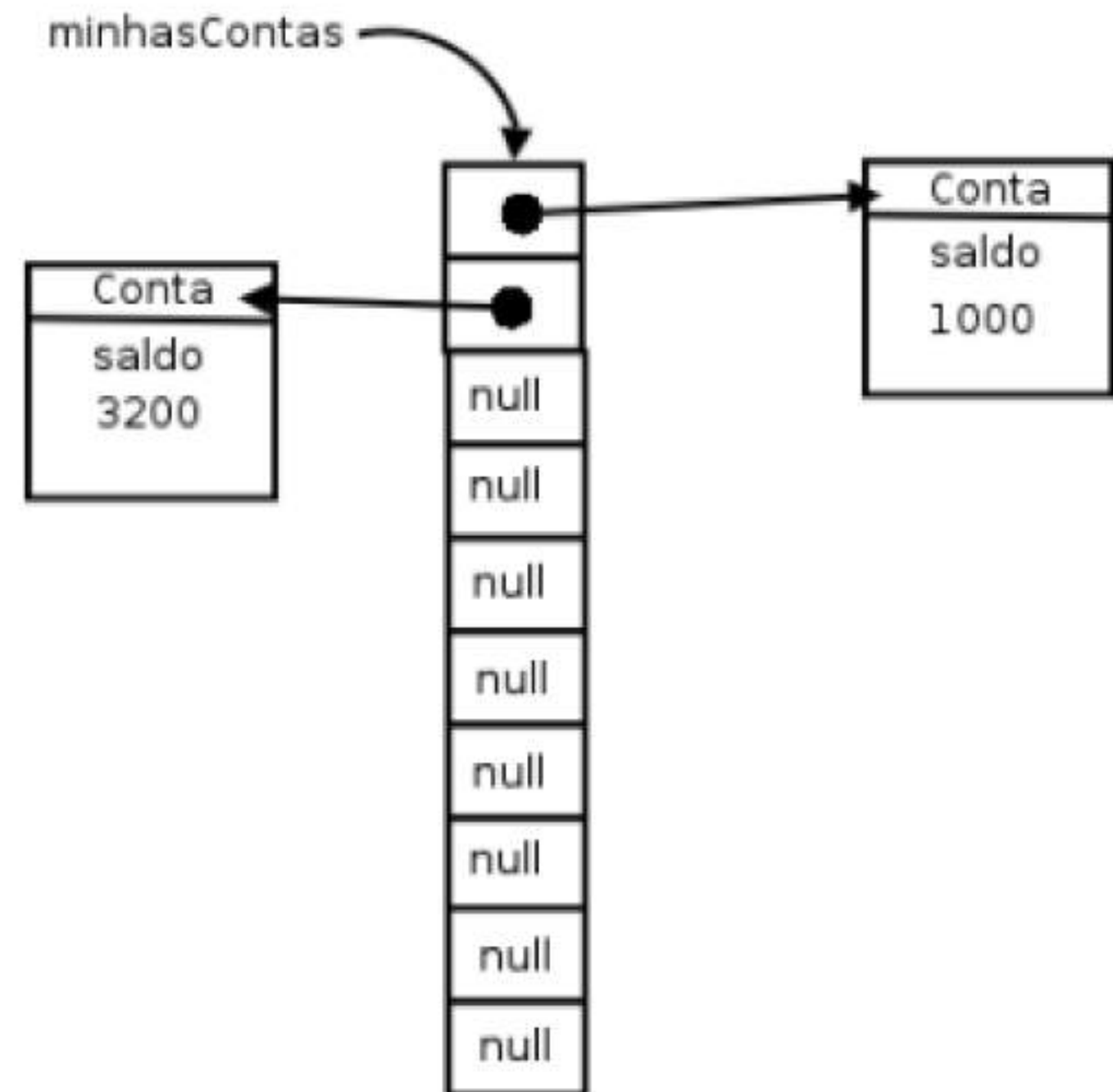
```
Conta[] minhasContas;  
minhasContas = new Conta[10];
```

No exemplo acima temos 10 espaços criados para guardar a referência a uma Conta. No momento elas referenciam para lugar algum (null).

Popular antes de usar



```
Conta[] minhasContas;  
minhasContas = new Conta[10];  
minhasContas[0].saldo = 1000.0;  
minhasContas[1].saldo = 3200.0;
```





MATRIZES


Matrizes



- Array com várias dimensões.
- Declaração:
double[][] matriz = new double[3][4];
ou
double matriz[][] = new double[3][4];
- Matriz de 3 linhas por 4 colunas.
- Exemplo:

ExemploMatriz

```
package modulo03exemplomatriz;
import java.util.Scanner;
public class ExemploMatriz {
    public static void main(String[] args) {
        double[][] matriz = new double[3][4];
        Scanner sc = new Scanner(System.in);
        //Carregando a matriz com dados do usuario
        for(int l=0; l<matriz.length;l++){
            for(int c=0;c<matriz[l].length; c++){
                System.out.println("matriz["+ l +"]["+ c +"]: ");
                matriz[l][c]=sc.nextDouble();
            }
        }
        //Exibindo a matriz
        for(int l=0; l<matriz.length;l++){
            for(int c=0;c<matriz[l].length; c++){
                System.out.printf(matriz[l][c] + "\t");
            }
            System.out.println(); //pular linha
        }
    }
}
```



Passagem de Parâmetros

Tipos primitivos

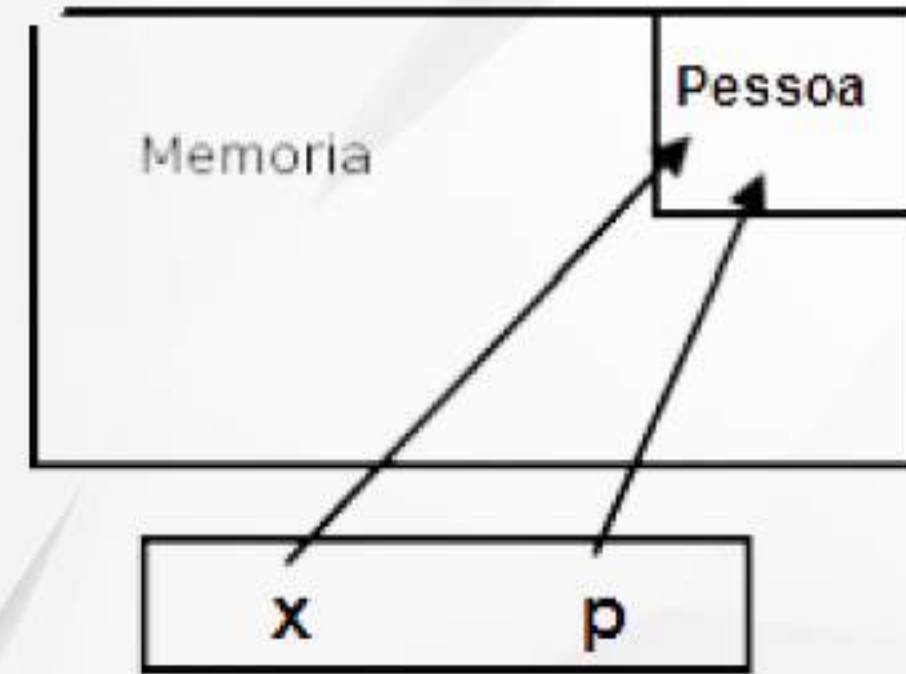


- Java tem apenas o meio de passagem **por valor**.

```
public class PassagemValor {  
    public void alterar(int a)  
    {  
        a = 50;  
        System.out.println("Valor interno na função alterar: " + a);  
    }  
    public static void main(String[] args) {  
        PassagemValor obj = new PassagemValor();  
        int a = 28;  
        obj.alterar(a);  
        System.out.println("Valor de A: " + a);  
    }  
}
```


Os tipos referência, veja a passagem

```
public class PassagemRef {  
  
    public static void meuMetodo(Pessoas p) {  
        p.nome = "Oscar";  
        p.diaNasc = 7;  
        System.out.println("Objeto: " + p);  
    }  
  
    public static void main(String[] args) {  
        Pessoas x = new Pessoas();  
        x.nome = "Lucas";  
        x.diaNasc = 5;  
        meuMetodo(x);  
        System.out.println("Objeto: " + x);  
        System.out.println("Nome: " + x.nome);  
        System.out.println("Dia nasc: " + x.diaNasc);  
    }  
}
```



Nesse código, quando utilizamos x ou p estamos nos referindo exatamente ao mesmo objeto! Elas são duas referências distintas, porém apontam para o mesmo objeto!



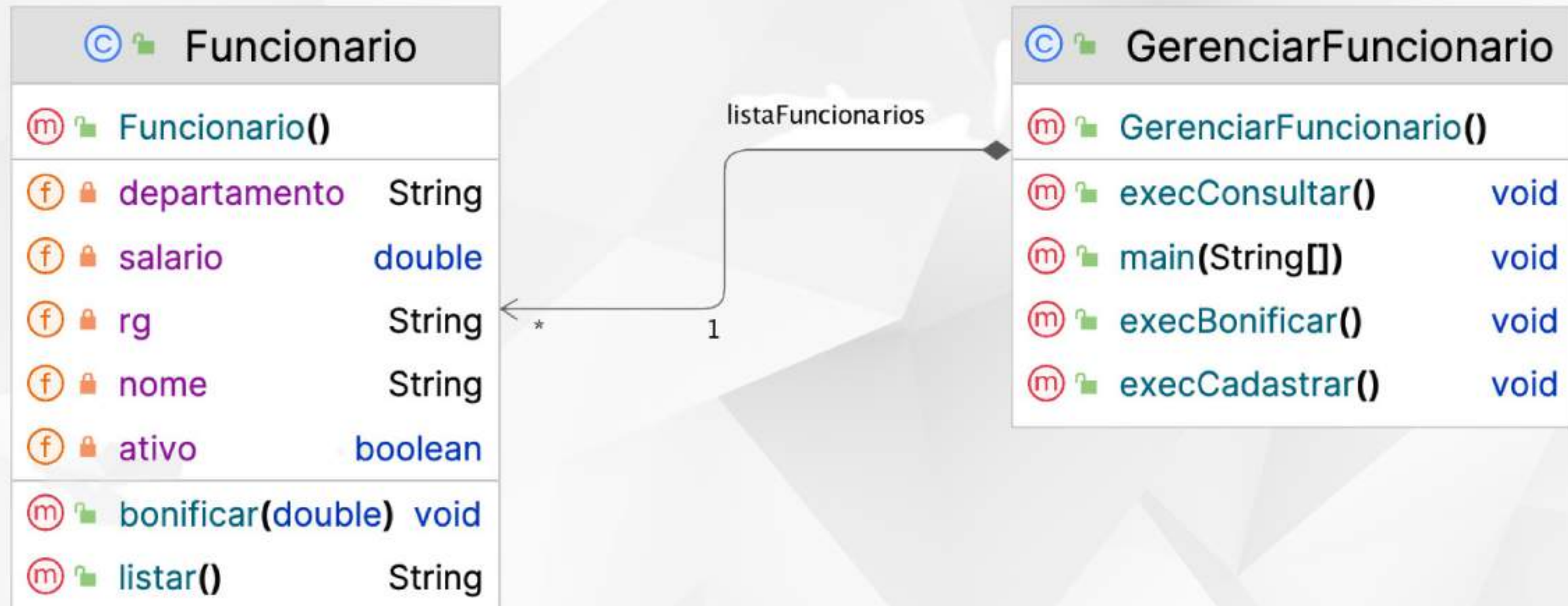
DESAFIO

Exercício

Programa: ProjetoFuncionarios



Objetivo: Uso de associação de objetos instanciados. Projeto a ser modelado



- **Classe: Funcionario**
 - **Atributos: nome, departamento, salario, rg, estaNaEmpresa**



Métodos	Descrição
bonificar()	Deverá receber como parâmetro o valor a ser acrescentado ao salário, e em seguida efetuar o devido acréscimo. Esse método não retorna nada.
listar()	Devolve uma String com os atributos da classe.

Classe: PrincipalFuncionarios



Método	Descrição
main()	Implementá-lo conforme padrão da linguagem Java. O método main() deverá criar um loop para o usuário escolher entre as opções cadastrar, consultar, bonificar. Se for selecionada a opção cadastrar, executar o método execCadastro(). Se for selecionado consultar, executar o método execConsulta(). Para a opção bonificar, executar o método execBonificacao().
execBonificacao()	Verificar primeiro a conta a ser bonificada, realizando um teste para verificar a existência. Em seguida solicitar ao usuário que posição (funcionário) deseja ler. Finalmente, solicitar ao usuário que digite um valor e executar o método bonificar() da classe Funcionarios usando o atributo criado. Ao final, mostre a mensagem "Aumento Concedido".
execConsulta()	Verificar se algum cadastro já foi criado, realizando um teste. Em seguida solicitar ao usuário que posição (funcionário) deseja ler. Finalmente, apresentar os atributos na tela executando o método imprimir() da classe Funcionarios.
execCadastro()	Solicitar que o usuário realize a leitura dos dados via teclado e em seguida realize a atribuição dos valores lidos do teclado aos atributos do objeto classe Funcionarios, criado como atributo dessa classe. Como estamos usando vetores neste exercício, precisamos criar um novo objeto para cada cadastro realizado.



Obrigado

fim



Até a próxima aula





Referências Bibliográficas

- ❑ Mendes; **Java com Ênfase em Orientação a Objetos**, Novatec.
- ❑ Deitel; **Java, como programar** – 10º edição. Java SE 7 e 8
- ❑ Arnold, Gosling, Holmes; **A linguagem de programação Java** – 4º edição.
- ❑ Apostilas da Caelum.