Fatec
Mogi Mirim
Arthur de Azevedo

PROGRAMAÇÃO ORIENTADA A OBJETOS [POO]

Material 04 – POO_04

Prof. Mestre Marcos Roberto de Moraes [Maromo]

Mais Recursos da Linguagem Java

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 c**omo 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

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

```
2   public class ExemploVetor {
3       public static void main(String[] args) {
4           //Definindo o vetor, sendo apenas uma referência.
5           int[] idade;
6           idade = new int[5]; // Alocando cinco posições inteiras para o vetor
7           idade[0] = 21;
8           idade[1] = 18;
9           idade[2] = 16;
10          idade[3] = 24;
11          idade[4] = 19;
12          for(int i=0; i<idade.length;i++){
13              System.out.printf("Elemento %d - idade %d \n", i,idade[i]);
14          }
15          int[] idadeGrupo2 = {14,21,35};
16           for(int i=0; i<idadeGrupo2.length;i++){
17              System.out.printf("grupo 2, Elemento %d - idade %d \n", i,idadeGrupo2[i]);
18          }
19      }
20  }
```

Exemplo:

- Resultado do Exemplo:

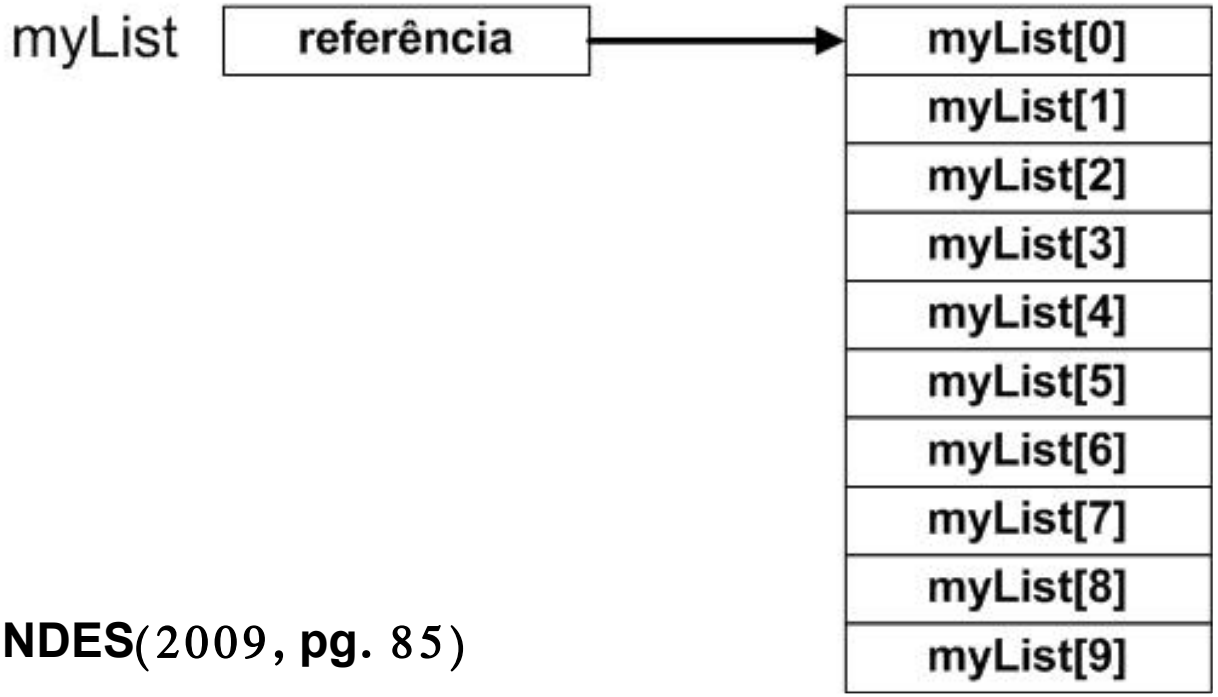
```
Elemento 0 - idade 21
Elemento 1 - idade 18
Elemento 2 - idade 16
Elemento 3 - idade 24
Elemento 4 - idade 19
grupo 2, Elemento 0 - idade 14
grupo 2, Elemento 1 - idade 21
grupo 2, Elemento 2 - idade 35
```

Array – Atributo length

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

Array – Representação na memória

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



MENDES(2009, **pg.** 85)

Exemplo:

ExemploVetor2

```
import java.util.Scanner;
public class ExemploVetor2 {
    public static void main(String[] args) {
        int[] var1 = new int[10];
        Scanner sc = new Scanner(System.in);
        System.out.print("Informe os 10 valores \n");
        //Percorrendo o vetor
        for(int i=0; i<var1.length; i++){
            System.out.print("var1[" + i + "] ? ");
            var1[i] = sc.nextInt();
        }
        int soma=0;
        int cont=0;
        for(cont=0;cont<var1.length; cont++){
            soma+=var1[cont];
        }
        float media = (float)soma/(float)(cont);
        System.out.printf("Média = %2.2f \n", media);
    }
}
```

Resultado do Exemplo:

```
run:
Informe os 10 valores
var1[0] ? 1
var1[1] ? 2
var1[2] ? 3
var1[3] ? 4
var1[4] ? 5
var1[5] ? 6
var1[6] ? 7
var1[7] ? 8
var1[8] ? 9
var1[9] ? 10
Média = 5,50
```

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 i=0; i<vet2.length;i++)
            System.out.println("Posicao vet2[" + i + "] = " + vet2[i]);
    }
}
```

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

```
Posicao vet2[0] = 0
Posicao vet2[1] = 0
Posicao vet2[2] = 0
Posicao vet2[3] = 0
Posicao vet2[4] = 0
Posicao vet2[5] = 0
Posicao vet2[6] = 1
Posicao vet2[7] = 2
Posicao vet2[8] = 3
Posicao vet2[9] = 4
```

Método Arrays.fill
ExemploArraysFill

- 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[] vet = new int[5];
        Arrays.fill(vet, 15);
        for(int i=0;i<vet.length;i++){
            System.out.println("vet[" + i + "] = " + vet[i]);
        }

    }
}
```

Resultado da Execução

```
run:
vet[0] = 15
vet[1] = 15
vet[2] = 15
vet[3] = 15
vet[4] = 15
```

Arrays de Referência

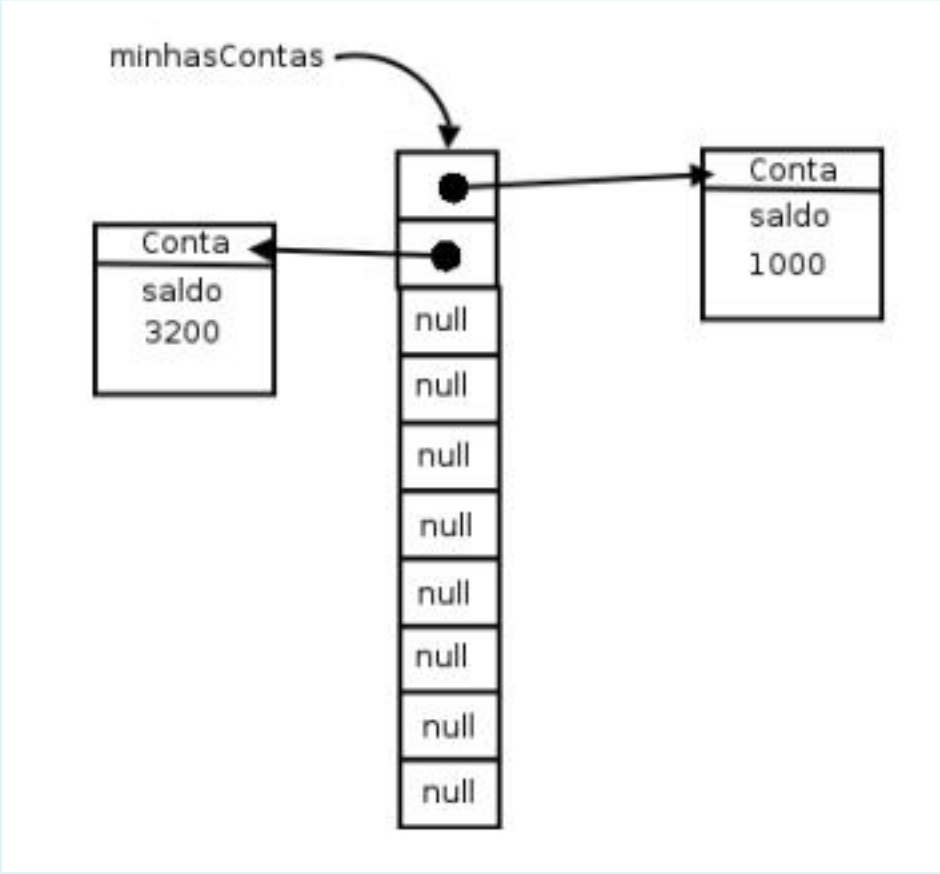
- Quando criamos um array de uma classe, ela possui referências. O objeto, está na memória principal e, no array ficam guardadas as referências. [endereços]

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

No exemplo acima temos 10 espaços criados para guardar a referência a uma Conta. No momentos 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;
```



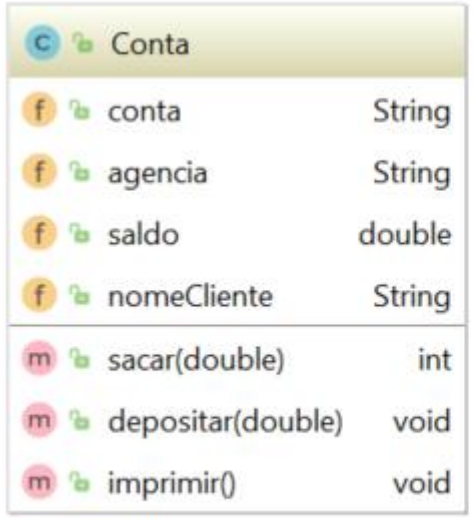
Laboratório

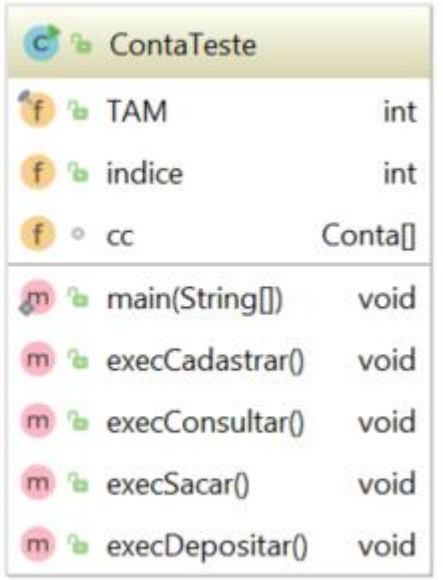
Prática de Vetor em Programa Orientado a Objetos

Objetivo

Programa: Poo04_Laboratorio1

- Uso de vetor de referência a objetos.
- Classes a serem criadas





Objetivo

- Classe: **Conta**
  - Atributos: conta, agencia, saldo, e nomeCliente

Método	Descrição
sacar( )	Retorna o valor 1 caso o saque se realizado ou 0 se não houver saldo suficiente na conta. Deverá receber como parâmetro o valor a ser sacado.
depositar( )	Realizar o depósito do valor recebido como parâmetro. Não retornar valor.
imprimir( )	Exibir na tela os atributos da classe. Esse método não retorna nada

Classe: Conta

```
public class Conta {
    public int conta;
    public int agencia;
    public double saldo;
    public String nome;

    public int sacar(double valor){
        int ret = 0;
        if(valor<=this.saldo){
            this.saldo-=valor;
            ret =1;
        }
        return ret;
    }

    public void depositar(double valor){
        this.saldo+=valor;
    }

    public void imprimir(){
        System.out.println("Conta.....: " + this.conta);
        System.out.println("Agência.....: " + this.agencia);
        System.out.println("Saldo.....: " + this.saldo);
        System.out.println("Nome.....: " + this.nome);
    }
}
```

- **Criar uma constante TAM com o valor 3.**
- **Criar um atributo da classe ContasCorrente do tipo vetor, com o tamanho TAM.**
- **Criar um atributo índice do tipo usando o seguinte comando: public int indice = 0;**

Classe: ContaTeste

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, depositar, sacar, consultar. Se for selecionada a opção sacar, executar o método execSacar( ). Se for selecionado depositar, executar o método execDepositar( ). Para a opção consultar, executar o método execConsultar( ). Para a opção cadastrar, executar o método execCadastrar( ).
execSaque( )	Verificar se alguma conta já foi criada, realizando um teste com o atributo índice. Em seguida solicitar ao usuário que posição (conta) deseja ler. Finalmente, solicitar ao usuário que digite um valor e executar o método sacar( ) da classe ContasCorrentes usando o atributo criado. Testar o retorno do método sacar( ). Se for retornado 1, exibir “Saque realizado”, caso contrário, exibir “Saque não realizado”.
execDeposito( )	Verificar se alguma conta já foi criada, realizando um teste com o atributo índice. Em seguida solicitar ao usuário que posição (conta) deseja ler. Finalmente, solicitar ao usuário que digite um valor e executar o método depositar( ) da classe ContasCorrentes usando o objeto criado anteriormente.
execConsulta( )	Verificar se alguma conta já foi criada, realizando um teste com o atributo índice. Em seguida solicitar ao usuário que posição (conta) deseja ler. Finalmente, apresentar os atributos na tela executando o método imprimir( ) da classe ContasCorrentes.
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 ContasCorrentes, criado como atributo dessa classe. Como estamos usando vetores neste laboratório, precisamos criar um novo objeto para cada cadastro realizado. Para isso, usar o comando: this.cc[índice].conta = SC.nextInt( ). Ao final da leitura dos dados, verificar se o atributo índice

Linhas de 2 a 24

Classe: ContaTeste

```
import java.util.Scanner;
public class  ContaTeste {
    public static int TAM=3;
    public int indice=0;
    ContasCorrentes[] cc = new ContasCorrentes[TAM];

    public void execCadastro(){
        if(indice>=TAM){
            System.out.println("Todos os espaços para contas foram ocupados");
            return;
        }
        Scanner sc = new Scanner(System.in);
        this.cc[indice] = new ContasCorrentes();
        System.out.println("Digitar o nome.....: ");
        this.cc[indice].nome = sc.nextLine();
        System.out.println("Digitar a conta.....: ");
        this.cc[indice].conta = sc.nextInt();
        System.out.println("Digitar a agência.....: ");
        this.cc[indice].agencia = sc.nextInt();
        System.out.println("Digitar o saldo.....: ");
        this.cc[indice].saldo = sc.nextDouble();
        indice++;
    }
```

Linhas de 25 a 37

```
public void execConsulta(){
if(indice>0){
        System.out.println("Digite a posição da conta [0,1 ou 2]: ");
        Scanner sc = new Scanner(System.in);
        int op = sc.nextInt();
        if((op<0)||(op>2))
            System.out.println("Posição inválida");
        else
            this.cc[op].imprimir();
    }else{
        System.out.println("Nenhuma conta cadastrada");
    }
}
```

Linhas de 38 a 59

```
public void execSaque(){
 if(indice>0){
        System.out.println("Digite a posição da conta [0,1 ou 2]: ");
        Scanner sc = new Scanner(System.in);
        int op = sc.nextInt();
        if(op>indice)
            System.out.println("Posição inválida");
        else{
            //efetua o saque
            System.out.println("Digite o valor do saque..:");
            double valor = sc.nextDouble();
            int ret = this.cc[op].sacar(valor);
            if(ret==0){
                System.out.println("Saque não realizado");
            }else{
                System.out.println("Saque realizado com sucesso");
            }
        }
    }else{
        System.out.println("Nenhuma conta cadastrada");
    }
}
```

Linhas de 60 a 77

```
public void execDeposito(){
     if(indice>0){
        System.out.println("Digite a posição da conta [0,1 ou 2]: ");
        Scanner sc = new Scanner(System.in);
        int op = sc.nextInt();
        if(op>indice)
            System.out.println("Posição inválida");
        else{
            //efetua o depósito
            System.out.println("Digite o valor do depósito..:");
            double valor = sc.nextDouble();
            this.cc[op].depositar(valor);
            System.out.println("Deposito realizado com sucesso");
         }
    }else{
        System.out.println("Nenhuma conta cadastrada");
    }
}
```

Linhas de 78 a 107 - *main*

```
public static void main(String[] args) {
    PrincipalContasCorrentes obj = new PrincipalContasCorrentes();
    Scanner sc = new Scanner(System.in);
    int opcMenu = 0;
    while(opcMenu!=9){
        System.out.println("Dados de 03 Contas");
        System.out.println("1) Cadastrar");
        System.out.println("2) Sacar");
        System.out.println("3) Depositar");
        System.out.println("4) Consultar");
        System.out.println("9) Sair");
        System.out.println("Entre com uma opção: ");
        opcMenu =sc.nextInt();
        switch(opcMenu){
            case 1:
                obj.execCadastro();
                break;
            case 2:
                obj.execSaque();
                break;
            case 3:
                obj.execDeposito();
                break;
            case 4:
                obj.execConsulta();
                break;
        }
    }
}
```

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

```java
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

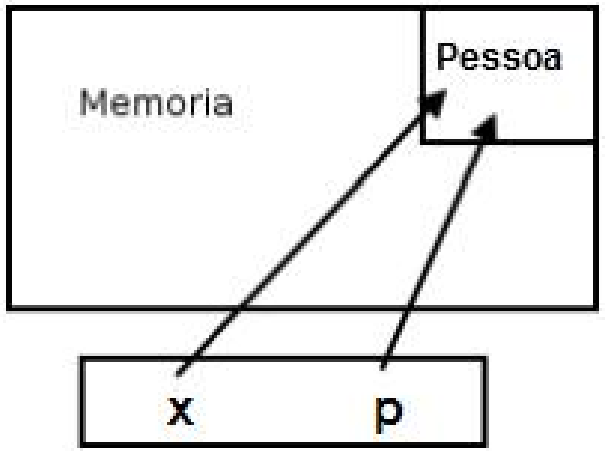
- Java tem apenas o meio de passagem **por valor.**

```
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);
    }
}
```

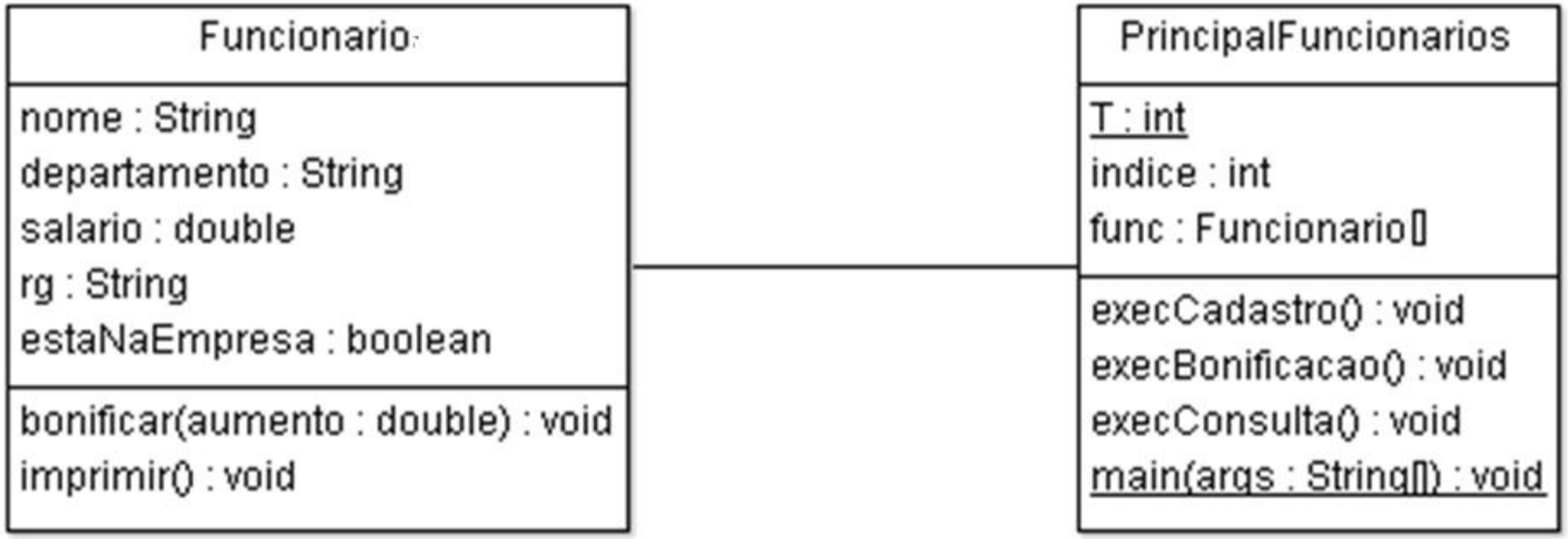


Nesse código, quando utilizamos x ou p estamos nos referindo exatamente ao mesmo objeto! Elas **são duas referências distintas**, porém apontam para o mesmo objeto!

Exercício

Programa: ProjetoFuncionarios

- Objetivo: Uso de vetores 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 acrescido ao salário, e em seguida efetuar o devido acréscimo. Esse método não retorna nada.
imprimir( )	Exibir na tela os atributos da classe. Esse método não retorna nada

Classe: PrincipalFuncionari

- **Criar um atributo constante T com o valor 10. [Tamanho do Vetor]**
- **Criar um atributo da classe Funcionario do tipo vetor, com o tamanho T.**
- **Criar um atributo do tipo estático usando o seguinte comando: public int indice = 0;**

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 se algum conta já foi criado, realizando um teste com o atributo índice. 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 com o atributo índice. 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. Para isso, usar o comando: this.func[indice].conta = SC.nextInt( ). Ao final da leitura dos dados, verificar se o atributo índice alcançou o seu limite. Caso isso ocorra, exibir uma mensagem na tela e reinicializá-lo com 0(Zero).

Referências

- Bibliográficas:
  - Mendes – Java com Ênfase em Orientação a Objetos [Exercícios do Capítulo 1]
  - Deitel – Java, como programar – 6º edição.
  - Arnold, Gosling, Holmes – A linguagem de programação Java – 4º edição.
  - Apostilas Caelum
  - Material do Curso de Capacitação Java do CPS
- Internet
  - http://java.sun.com
  - http://www.guj.com.br
  - http://www.portaljava.com

FIM

Maromo