Solução da Lista de Exercícios nº 5
Usando Herança e Polimorfismo

Prof. Marcos Esteves


```
// Questão 1 (Exercício14)
package geral;

public class Funcionario_4
{
	private static int mat_aux = 0; // mat_aux passa a ser um contador de objetos da classe
	private int mat; // mat passa a ser variável de instancia
	private String nome, funcao;
	private double salario;

	public Funcionario_4() {
		mat = ++mat_aux;
	}

	public Funcionario_4(String nome_aux, String funcao_aux, double sal_aux) {
		this(); // Isto faz com que o construtor acima seja chamado incrementando mat
		nome = nome_aux;
		funcao = funcao_aux;
		salario = sal_aux;
	}

	// Solução da questão 14
	public Funcionario_4(String nome_aux, String funcao_aux) {
		this(nome_aux, funcao_aux, CargosSalarios.buscaSalario(funcao_aux));
	}

	public int getMat() // Não é mais estático devido a mat deixar de ser
	{
		return mat;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getNome() {
		return nome;
	}

	public void setFuncao(String funcao) {
		this.funcao = funcao;
	}

	public String getFuncao() {
		return funcao;
	}

	public void setSalario(double salario) {
```

```
		this.salario = salario;
	}

	public double getSalario() {
		return salario;
	}
}


// Questão 1 - continuação
package geral;

public class CargosSalarios
{
	public static double buscaSalario(String funcao) {
		if (funcao.equals("diretor"))
			return 5000.00;
		else if (funcao.equals("vendedor"))
			return 3000.00;
		else
			return 1000.00;
	}
}

// Questao 1 - continuação
package geral;

public class Vendedor extends Funcionario_4
{
	private double totalVendas, comissao;

	public Vendedor() {
		super();
	}

	public Vendedor(String nome_aux, String funcao_aux, double sal_aux, double totalVendas,
double comissao) {
		super(nome_aux, funcao_aux, sal_aux);
		this.totalVendas = totalVendas;
		this.comissao = comissao;
	}

	public void setTotalVendas(double totalVendas) {
		this.totalVendas = totalVendas;
	}

	public double getTotalVendas() {
		return totalVendas;
	}

	public void setComissao(double comissao) {
```

```
		this.comissao = comissao;
	}

	public double getComissao() {
		return comissao;
	}

	public double calcularSalarioVendedor() {
		return getSalario() + totalVendas * comissao;
	}
}


package geral;

import java.util.*;

public class TestaVendedor // Questao 2
{
	public static void main(String args[]) {
		Vendedor vet[] = new Vendedor[3];
		double totalVendas_aux;
		String nome_aux, funcao_aux;
		double sal_aux, comissao_aux;

		try (Scanner in = new Scanner(System.in);) // Recurso do java 7 para não ter que
fechar ao final
		{
			for (int i = 0; i < 3; i++) {
				System.out.printf("\nEntre com os dados para o Vendedor %d\n", i
+ 1);
				System.out.print("Entre com o nome: ");
				nome_aux = in.nextLine();
				System.out.print("Entre com a funcao: ");
				funcao_aux = in.nextLine();
				System.out.print("Entre com o salario: ");
				sal_aux = in.nextDouble();
				System.out.print("Entre com o total de vendas: ");
				totalVendas_aux = in.nextDouble();
				System.out.print("Entre com a comissao:  ");
				comissao_aux = in.nextDouble();
				in.nextLine();
				vet[i] = new Vendedor(nome_aux, funcao_aux, sal_aux,
totalVendas_aux, comissao_aux);
			}
			for (int i = 0; i < 3; i++) {
				System.out.printf("\n\nOs dados do Vendedor %d sao:\n ", i + 1);
				System.out.printf("matricula: %d\n", vet[i].getMat());
				System.out.printf("nome: %s\n", vet[i].getNome());
				System.out.printf("funcao:%s\n", vet[i].getFuncao());
```

```
                                        System.out.printf("salario a receber: %,.2f\n",
vet[i].calcularSalarioVendedor());
                                        System.out.println();
                                }
                        } catch (InputMismatchException e) {
                                System.out.println("Valor digitado nao corresponde ao esperado !");
                        }
                }
}

// Questão 3

package geral;


import java.time.LocalDate;
import java.time.format.DateTimeFormatter;


class Empregado {
        private int mat;
        private String nome, funcao;
        private double salario;
        private LocalDate data_adm;
        public static int mat_aux = 0;

        DateTimeFormatter formatador = DateTimeFormatter.ofPattern("dd/MM/yyyy");

        public Empregado() {
                super();
                mat = ++mat_aux;
        }

        public Empregado(String nome, String funcao, double salario, LocalDate data_adm) {
                this();
                this.nome = nome;
                this.funcao = funcao;
                this.salario = salario;
                this.data_adm = data_adm;
        }

        public int getMat() {
                return mat;
        }

        public void setNome(String nome) {
                this.nome = nome;
        }

        public String getNome() {
                return nome;
```

```
	}

	public void setFuncao(String funcao) {
		this.funcao = funcao;
	}

	public String getFuncao() {
		return funcao;
	}

	public void setSalario(double salario) {
		this.salario = salario;
	}

	public double getSalario() {
		return salario;
	}

	public void setDataAdm(LocalDate data_adm) {
		this.data_adm = data_adm;
	}

	public LocalDate getDataAdm() {
		return data_adm;
	}

	public void aumentaSalario(double perc) {
		salario = salario + (salario * perc / 100);
	}

	public String toString() {
		return "\nMat: " + mat + "\nNome: " + nome + "\nFunção: " + funcao + "\nSalario:
"+ String.format("%,.2f", salario) + "\nData de Admissão: " + data_adm.format(formatador);
	}
}

// Questão 4

package geral;

import java.time.LocalDate;
import java.time.Period;

class Gerente extends Empregado // Questão 4
{
	private Empregado secretaria;
	private String automovel;

	public Gerente() {
		super();
	}
```

```
	public Gerente(String nome, String funcao, double salario, LocalDate data_adm,
Empregado secretaria, String automovel) {
		super(nome, funcao, salario, data_adm);
		this.secretaria = secretaria;
		this.automovel = automovel;
	}

	public void setSecretaria(Empregado secretaria) {
		this.secretaria = secretaria;
	}

	public Empregado getSecretaria() {
		return secretaria;
	}

	public void setAutomovel(String automovel) {
		this.automovel = automovel;
	}

	public String getAutomovel() {
		return automovel;
	}

	public void aumentaSalario(double perc) {
		double bonus;
		LocalDate dataAux = LocalDate.now();

		// Usando Period do java 8
		Period periodo = Period.between(dataAux, getDataAdm());
		bonus = 0.5 * periodo.getYears();

		super.aumentaSalario(perc + bonus); // Chama o aumentaSalario() de
		// Empregado, pois essa parte é igual e não precisamos repetir o código
	}

	public String toString() {
		return super.toString() + "\nNome da Secretaria: " + secretaria.getNome() +
"\nAutomóvel: " + automovel;
	}
}

// Classe de erro criada para o uso no ex. 5. Ao se criar uma Exception personalizada se define os
seguintes construtores: um default  e um que recebe uma String como argumento. Além disso  a
classe de erro deve estender de Exception.

class VetEmpregadosVazioException extends Exception {
	public VetEmpregadosVazioException() {
		super();
	}
```

```
	public VetEmpregadosVazioException(String s) {
		super(s); // Chama o construtor de Exception
	}
}

// Questão 5

package geral;

import java.time.LocalDate;
import java.util.*;

public class ManipulaEmpregado {
	private Empregado vetEmpregados[] = null;
	private Scanner in = new Scanner(System.in);

	public void criarEmpregados() {
		Empregado empregadoSecretaria;
		try {
			vetEmpregados = new Empregado[2];  // A quantidade foi alterada para 2
para facilitar o teste
			for (int i = 0; i < 2; i++) {
				if (i < 1) {
					System.out.println();
					System.out.println();
					System.out.println("Entre com os dados para um(a)
Empregado(a)");
					vetEmpregados[i] = new Empregado();
					System.out.print("Entre com o nome: ");
					vetEmpregados[i].setNome(in.nextLine());
					System.out.print("Entre com a funcao: ");
					vetEmpregados[i].setFuncao(in.nextLine());
					System.out.print("Entre com o salario: ");
					vetEmpregados[i].setSalario(in.nextDouble());
					in.nextLine();
					vetEmpregados[i].setDataAdm(LocalDate.of(2010, 04,
01));
				} else {
					System.out.println();
					System.out.println();
					System.out.println("Entre com os dados para um Gerente");
					vetEmpregados[i] = new Gerente();
					System.out.print("Entre com o nome: ");
					vetEmpregados[i].setNome(in.nextLine());
					System.out.print("Entre com a função: ");
					vetEmpregados[i].setFuncao(in.nextLine());
					System.out.print("Entre com o salario: ");
					vetEmpregados[i].setSalario(in.nextDouble());
					in.nextLine();
```

```
                                        vetEmpregados[i].setDataAdm(LocalDate.of(2010, 04,
01));

                                        System.out.println("\n--- Entre com a secretária ---");
                                        empregadoSecretaria = new Empregado();
                                        System.out.print("Entre com o nome: ");
                                        empregadoSecretaria.setNome(in.nextLine());
                                        empregadoSecretaria.setFuncao("Secretária");
                                        System.out.print("Entre com o salario: ");
                                        empregadoSecretaria.setSalario(in.nextDouble());
                                        empregadoSecretaria.setDataAdm(LocalDate.now());

                                        // Completando a criação do Gerente com secretária e
automóvel
                                        ((Gerente)
vetEmpregados[i]).setSecretaria(empregadoSecretaria);
                                        System.out.print("\nEntre com o Automovel do gerente: ");
                                        ((Gerente) vetEmpregados[i]).setAutomovel(in.next());
                                        in.nextLine();
                                }
                        }

                } catch (InputMismatchException e) {
                        System.out.println("Valor digitado nao corresponde ao esperado !");
                        System.exit(0);
                }
//                      return vetEmpregados;
        }

        public void listarEmpregados() {
                try {
                        if (vetEmpregados == null)
                                throw new VetEmpregadosVazioException("Criar antes o vetor de
Empregados !");

                        for (int i = 0; i < 2; i++) {
                                System.out.println(vetEmpregados[i].toString());
                        }
                } catch (VetEmpregadosVazioException e) {
                        System.out.println(e.getMessage());
                        System.exit(0);
                }
        }

        public void aumentarEmpregados(double perc) throws VetEmpregadosVazioException //
Colocado aqui para não ter que tratar e poder compilar
        // Sem o try mas compila devido ao throws acima


        {
                if (vetEmpregados == null)
```

```
                        throw new VetEmpregadosVazioException("Criar antes o vetor de
Empregados !");

                for (int i = 0; i < 2; i++) {
                        vetEmpregados[i].aumentaSalario(perc);
                }
        }

        public String localizarEmpregado(int matAux) throws VetEmpregadosVazioException {
                if (vetEmpregados == null)
                        throw new VetEmpregadosVazioException("Criar antes o vetor de
Empregados !");
                String stringRetorno = null;
                int i;
                for (i = 0; i < 2; i++) {
                        if (vetEmpregados[i].getMat() == matAux) {
                                stringRetorno = vetEmpregados[i].getNome();
                                break;
                        }
                }
                if (i == 2) {
                        System.out.println();
                        stringRetorno = "Empregado nao Localizado !";
                }
                return stringRetorno;
        }

        /*
         * Para testar os métodos acima
         *
         * public static void main(String args[]) throws VetEmpregadosVazioException {
         *      ManipulaEmpregado me = new ManipulaEmpregado();
         *      me.criarEmpregados();
         *      me.aumentarEmpregados(10);
         *  me.listarEmpregados(); }
         */
}
// Questao 6

package geral;

import java.util.*;

class TestaExercicio5 {
        public static void main(String args[]) {
                int opcao = 0, matAux = 0;
                double perc = 0;
                ManipulaEmpregado me = new ManipulaEmpregado();

                try (Scanner in = new Scanner(System.in);) // Usando o try do java 7 para fechar o
Scanner automaticamente
```

```
		{
			while (opcao != 5) {
				opcao = 0;
				while ((opcao != 1) && (opcao != 2) && (opcao != 3) && (opcao
!= 4) && (opcao != 5)) {
					System.out.println();
					System.out.println("----------------------");
					System.out.println("Entre com uma opcao");
					System.out.println("1-Criar Vetor de Empregados");
					System.out.println("2-Aumentar Empregados");
					System.out.println("3-Listar Empregados");
					System.out.println("4-Localizar Empregado");
					System.out.println("5-Sair");
					System.out.print("opcao: ");
					opcao = in.nextInt();
				}
				switch (opcao) {
				case 1:
					me.criarEmpregados();
					break;
				case 2:
					System.out.print("Entre com o percentual de Aumento: ");
					perc = in.nextDouble();
					me.aumentarEmpregados(perc);
					break;
				case 3:
					me.listarEmpregados();
					break;
				case 4:
					System.out.print("Entre com a matricula do Empregado: ");
					matAux = in.nextInt();
					System.out.println();
					System.out.println("Nome: " +
me.localizarEmpregado(matAux));
					break;
				}
			}

		} catch (InputMismatchException e) {
			System.out.println("Erro de I/O !");
			System.exit(0);
		} catch (VetEmpregadosVazioException e) {
			System.out.println(e.getMessage());
			System.exit(0);
		}
	}
}
```