

Programação de computadores II

Trabalho 2: Sistema RHTec

Carolina Santos Lins¹, Gustavo Correa Rodrigues¹,
Livia Emanuelle Carrera Soares da Silva¹, Thiago Calado Sosinho¹

¹Faculdade de Computação – Instituto de Ciências Exatas e Naturais
Universidade Federal do Pará (UFPA)
Av. Augusto Correa 01, 66075-090 – Belém – PA – Brasil

{carolina.lins, gustavo.rodrigues, livia.silva, thiago.sosinho}@icen.ufpa.br

1. Introdução

Recursos Humanos é um requisito que sem ele não há empresa ou instituição. Sendo assim, o relatório tem como objetivo mostrar o desenvolvimento do sistema de recursos humanos para uma empresa, o RHTec, ele visa simplificar e melhorar a vida dos gestores de recursos no que tange a distribuição das várias áreas e funcionalidades de uma determinada empresa.



Figura 1. Sistema RHTec, logo.

O sistema RHTec facilita atividades como listagem de funcionários, adição de novos funcionários, análises e avaliações de desempenho, entre outros. A companhia que conta com essa tecnologia se torna mais competitiva, impactando em estratégia de desenvolvimento.

2. Modelagem do Software

O primeiro passo no processo de desenvolvimento do sistema foi fazer a análise e o levantamento de requisitos. Inicialmente, as necessidades se resumiam a uma ferramenta que gerenciasse os dados de funcionários e setores, de forma prática e com a usabilidade

orientada aos funcionários que realmente iriam manusear com o sistema. Com esses requisitos, também foi apresentada a necessidade de registrar os funcionários, sendo essas ações importantes para a empresa, e por isso teria de estudar a melhor forma possível de integra-las ao sistema, para acontecer a demissão, era necessário ter anteriormente o cadastro do funcionário, quanto aos setores é analisado, a quantidade total de funcionários e gastos com salários. A partir desse momento, e os diversos requisitos apontados tem-se criação de UML's que possibilitam a ideação do "problema" até então abstrato. Logo, a seguir têm-se os diagramas dos tipos: Classes, Casos de uso e Atividades para o sistema e algumas funcionalidades dele.

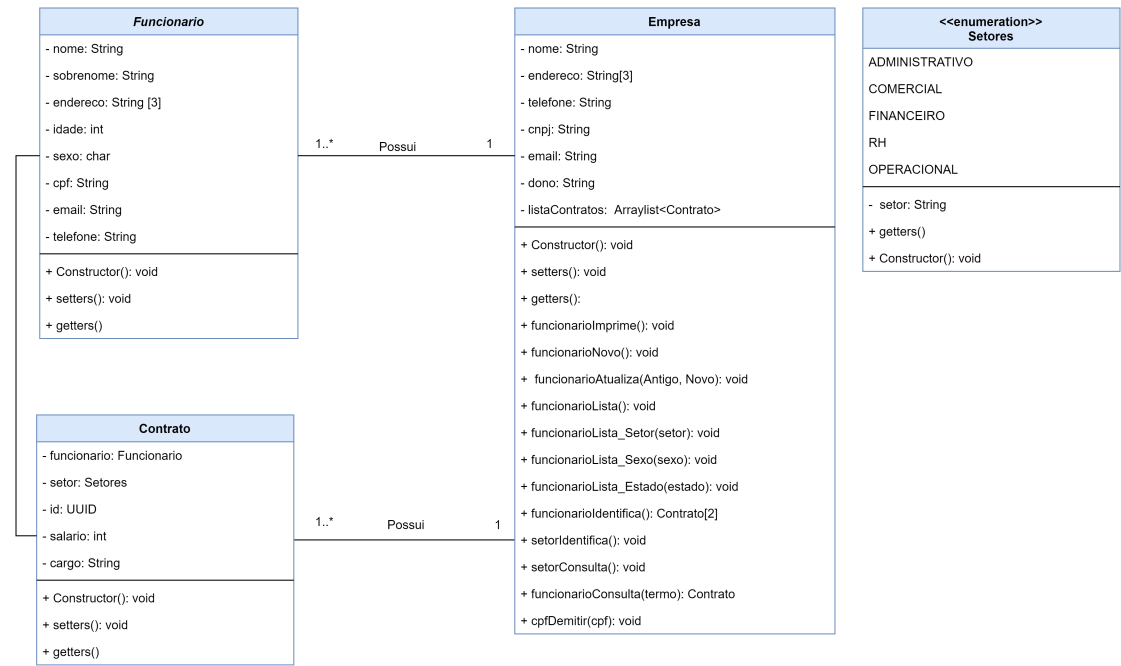


Figura 2. Diagrama de Classes do Sistema RHTEC

O Diagrama de Classes, como o próprio nome sugere representa as classes, sendo assim, retrata uma visualização específica que talvez o cliente leigo sinta dificuldade em entendimento, no entanto, é uma importante ferramenta utilizada na orientação de tarefas ao grupo desenvolvedor. Portanto, tendo em vista essas limitações é feito também o diagrama de casos uso mostrado a seguir:



Figura 3. Diagrama de Casos de Uso do Sistema RHTec

No diagrama de Casos de Uso, há melhor visualização das funcionalidades do programa, o gerenciador se por exemplo escolher dentre as funções de demitir, atualizar dados ou adicionar novo funcionário, terá acesso a tais e que podem ser representadas da seguinte forma nos diagramas de atividades propostos:

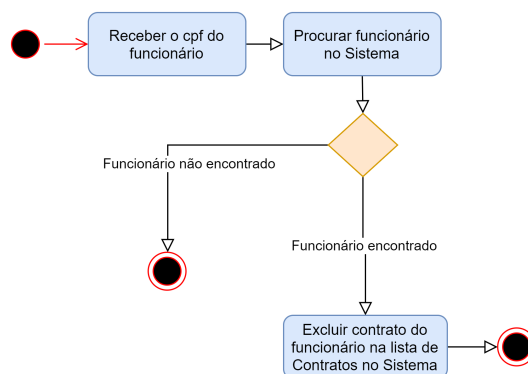


Figura 4. Diagrama de Atividades do método `cpfDemissao`

Na figura 4, temos a representação das ações que serão feitas pelo gerenciador ao querer demitir um funcionário.

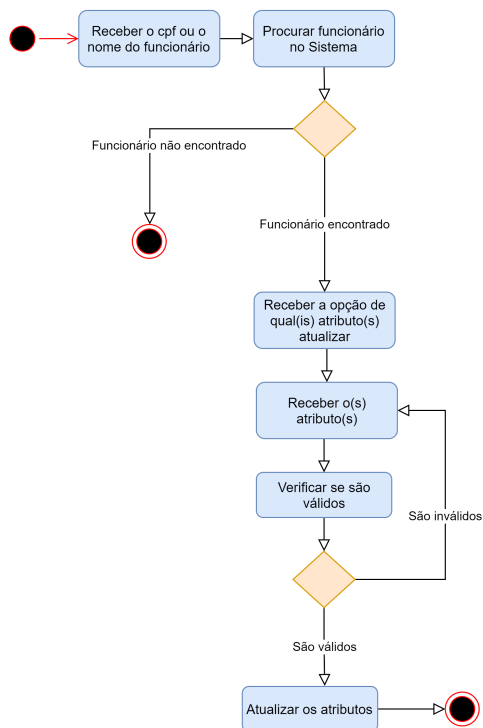


Figura 5. Diagrama de Atividades do método `funcionarioAtualiza`

Na figura 5, temos a representação das ações que serão feitas pelo gerenciador ao querer atualizar os dados do funcionário.

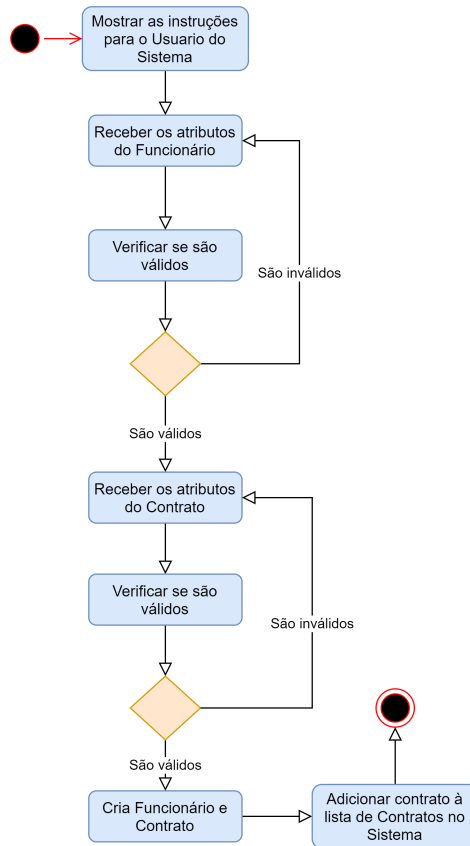


Figura 6. Diagrama de Atividades do método funcionarioNovo

Na figura 6, temos a representação das ações que serão feitas pelo gerenciador ao querer adicionar um novo funcionário ao sistema.

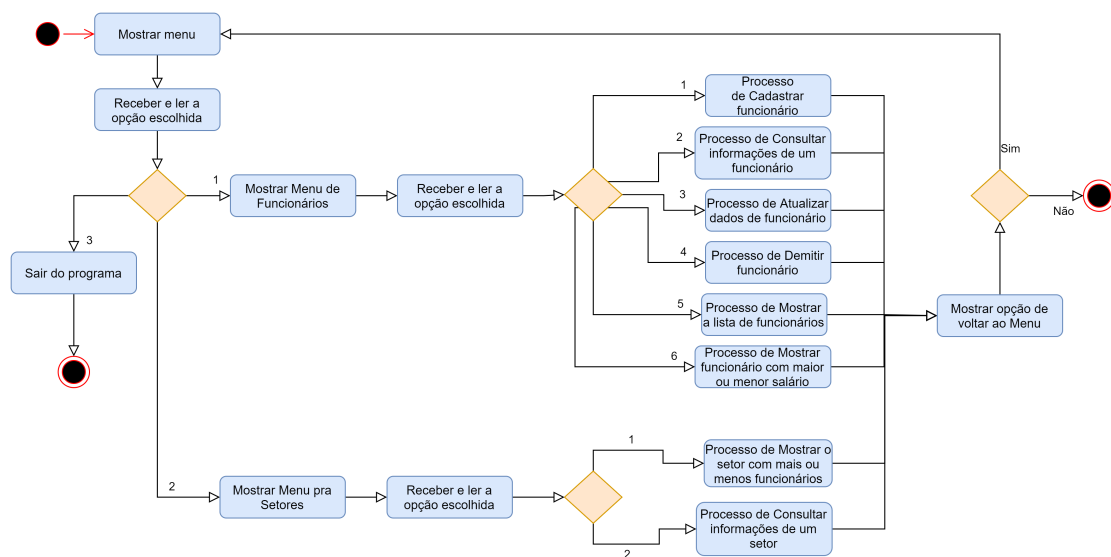


Figura 7. Diagrama de Atividades do Sistema RHTec

Contudo, no diagrama acima, há representação das atividades que podem ser feitas pelo gerenciador do programa. Por meio de switch case tem a seleção da função que se deseja e a realização da mesma posteriormente.

3. Aspectos gerais da codificação

No desenvolvimento do código é utilizada o IDE (Ambiente de Desenvolvimento Integrado) chamada IntelliJ IDEA Community Edition. Escrito em Java para o desenvolvimento de software de computador, é desenvolvido pela JetBrains.

Dentre as máquinas utilizadas para o desenvolvimento da aplicação temos:

Sistema operacional: Windows 10 Home Edition

CPU: Intel i5 8ª geração 1.6 GHz

RAM: 8 Gbs RAM com vídeo integrado

GPU: Intel graphics UHD 320

Sistema operacional: Windows 10 Home Edition

CPU: Intel i5 9ª geração 2.9 GHz

RAM: 8 Gbs RAM

GPU: RTX 2060

Sistema operacional: Windows 10 Home Edition

CPU: Intel i7 7ª geração 2.7 GHz

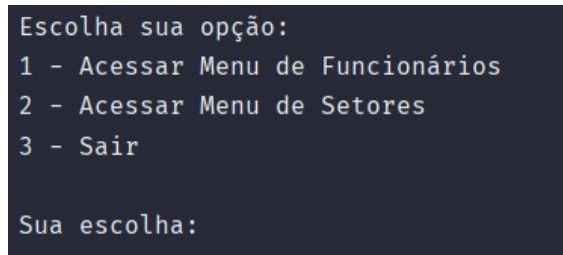
RAM: 8 Gbs RAM

GPU: Intel HD Graphics 620

4. Testes Computacionais

Nos testes será feito uma breve simulação do funcionamento do nosso sistema mostrando que ele cumpre com o proposto, seguindo caminhos que o gerenciador pode fazer ao utilizar a aplicação. Na condição de teste, foram inseridos uma Empresa criada e funcionários fictícios para demonstrar a funcionalidade dos métodos apresentados ao decorrer desta sessão.

A imagens a seguir são do console ao compilar o código:



```
Escolha sua opção:
1 - Acessar Menu de Funcionários
2 - Acessar Menu de Setores
3 - Sair

Sua escolha:
```

Figura 8. Menu Principal, Sistema RHTec.

Na figura 8, temos a primeira interação onde o gerenciador se depara com a seleção do que pretende acessar, funcionários ou setores, vamos supor primeiramente que irá gerenciar os funcionários e seleciona a opção "1".

4.1. Menu dos Funcionários

No menu dos funcionários, o programa apresenta diversas opções para a gestão de funcionários, aos quais são de suma importância para o funcionamento pleno do manejo de pessoal. A partir do momento ao qual é feita o cadastro de um funcionário novo, consulta de algum funcionário existente, atualização dos dados cadastrais deste funcionário, o processo de demissão, a listagem de funcionários cadastrados e uma opção que mostra o funcionário que recebe mais e o que recebe menos.

```
Escolha sua opção:
1 - Cadastrar um novo funcionário
2 - Consultar dados de um funcionário
3 - Atualizar dados de um funcionário
4 - Demitir um funcionário
5 - Mostrar lista de funcionários
6 - Mostrar os funcionários com o maior e o menor salário
```

Figura 9. Menu dos funcionários RHTec.

4.1.1. Cadastrar novo funcionário

Nesta sessão é feita o processo de cadastro do usuário, aos quais suas informações pessoais serão armazenadas na base de dados do programa. Este processo de admissão, também engloba o setor ao qual o funcionário vai estar vinculado, o valor da remuneração, e se ele vai estar na condição de colaborador ou chefe. Entende-se por colaborador, o profissional subordinado e remunerado pela empresa e seu setor responsável. Quanto ao chefe, trata-se do funcionário ao qual coordena o setor alocado pelo gerenciador de recursos humanos e está na condição de superior ao colaborador. Vale ressaltar que cada setor é necessário de um chefe e não há limitação de número de colaboradores.

Ao completar a ficha de inscrição do novo funcionário o sistema retorna ao seu menu principal, caso o usuário opte por isso. Caso contrário o programa terminará com sucesso.

```
Bem vind@ ao assistente de cadastro de funcionário
Insira o nome do funcionário a ser cadastrado
Thiago
Insira o sobrenome do funcionário a ser cadastrado
Silva
Insira o estado o funcionário mora
Para
Insira a cidade o funcionário mora
Belém
Insira o bairro o funcionário mora
Reduto
Insira a idade do funcionário
29
```

Figura 10. Processo de cadastro de um novo funcionário - parte 1.

```

Insira o cpf do funcionário (somente números)
0001100011
Insira o email do funcionário
email@email.com
Insira o telefone do funcionário (somente números)
9199999999
Insira o setor do funcionário
1. ADMINISTRATIVO
2. COMERCIAL
3. FINANCEIRO
4. RH
5. OPERACIONAL
3
Digite o salário do funcionário
1234
Escolha o cargo do funcionário:
1. Colaborador
2. Chefe
1
Deseja voltar ao menu principal? (S/N)

```

Figura 11. Processo de cadastro de um novo funcionário - parte 2.

4.1.2. Consultar dados de um funcionário

Ao realizar o processo de cadastro, em um ambiente de gestão de pessoas, a necessidade de confirmar dados cadastrais principalmente para fins de previdência, aposentadoria ou benefícios aos quais o trabalhador tem direito, se torna essencial na vida do setor de Recursos Humanos. Logo a importância da consulta dos dados cadastrais do funcionário torna-se um agente facilitador para a realização dessas atividades.

O sistema faz a busca pelo seu CPF e em seguida imprime o resumo contratual do funcionário com as informações pertinentes a ele.

```

Digite o nome ou cpf do funcionário que deseja consultar.
0001100011

Nome: Thiago Calado
Cpf: 0001100011
Idade: 29
Sexo: M
Endereço: Pará, Belém, Reduto
Telefone: 9199999999
Email: email@email.com
Id: 15702ac8-4da0-4d65-a6fe-8ad2a8154fb9
Setor: Comercial
Salário: R$1234,00
Cargo: Colaborador

```

Figura 12. Realização de consulta de funcionários RHTec.

4.1.3. Atualizar dados de um funcionário

O funcionário pode ter seus dados atualizados se houver necessidade pelo RH ou pelo mesmo. Nesse procedimento entra no processo de revisão cadastral do usuário, troca de setor, faixa salarial e uma possível mudança de atribuição dada, no caso chefe de setor ou colaborador, conforme a circunstância ocorra essa necessidade.


```
Digite o cpf do funcionário que deseja atualizar.  
93399288201  
  
Escolha o dado a ser atualizado:  
1. Nome e sobrenome  
2. Endereço  
3. Idade  
4. Sexo  
5. Cpf  
6. Email  
7. Telefone  
8. Setor  
9. Salário  
10. Cargo  
11. Sair  
9  
Insira o salário do funcionário  
6444
```

Figura 13. Processo de atualização de dados de um funcionário RHTec.

Ao realizar a operação, o usuário poderá fazer mais alterações ou retornar ao menu principal do programa.

4.1.4. Demitir um funcionário

O processo de desligamento de um funcionário se dá por diversos motivos: justa causa, sem justa causa, ou demissão por parte do empregado. O programa realiza o descadastro do pessoal utilizando o cadastro de pessoa física (CPF). O CPF cadastrado é encontrado e as informações vinculadas a este CPF também serão excluídas do sistema, realizando-se por completo o processo de desvínculo do trabalhador no sistema.

```
Digite o cpf do funcionário que deseja demitir.  
93399288201  
  
Deseja voltar ao menu principal? (S/N)  
|
```

Figura 14. Processo de demissão de um funcionário RHTec.

4.1.5. Mostrar lista de funcionários

Como citado no começo da sessão de testes, o programa conta com funcionários fictícios para que certas funcionalidades sejam realizadas. O método de consulta de funcionários utiliza 4 critérios: a busca por todos os funcionários, listar por Estado, por Setor cadastrado e por Sexo. Escolhido o critério, o sistema imprime o resumo contratual dos respectivos funcionários cadastrados.

```
3. FINANCEIRO
4. RH
5. OPERACIONAL
█
Nome: Paulo Souza
Cpf: 73848878783
Idade: 20
Sexo: M
Endereço: Pará, Belém, Marco
Telefone: 09198934984
Email: psouza@gmail.com
Id: 6e7bbaaa-cef5-4fb3-8a9c-6607fa41ed20
Setor: Comercial
Salário: R$23000,00
Cargo: Chefe

Nome: Julia Ramos
Cpf: 66382782882
Idade: 20
Sexo: F
Endereço: Pará, Belém, Guamá
Telefone: 09198888883
Email: jramos@gmail.com
Id: 7330c86d-8b25-4176-bcfb-72c578eddb68
Setor: Comercial
Salário: R$1000,00
Cargo: Colaborador
```

Figura 15. Exemplo de consulta de funcionários por setor RHTec.

4.1.6. Mostrar os funcionários com o maior e menor salário

A funcionalidade a seguir, com o critério estatístico ou de relatório, é realizado uma consulta do funcionário que é menor remunerado em contraste com o mais bem remunerado. Esta função tão sensível quanto as outras e o gerente de RH deverá ter cautela ao realizar a consulta.

```

Funcionário com menor salário:
Nome: Julia Ramos
Cpf: 66382782882
Idade: 20
Sexo: F
Endereço: Pará, Belém, Guamá
Telefone: 09198888883
Email: jramos@gamail.com
Id: 7330c86d-8b25-4176-bcfb-72c578eddb68
Setor: Comercial
Salário: R$1000,00
Cargo: Colaborador

Funcionário com maior salário:
Nome: Paulo Souza
Cpf: 73848878783
Idade: 20
Sexo: M
Endereço: Pará, Belém, Marco
Telefone: 09198934984
Email: psouza@gamail.com
Id: 6e7bbaaa-cef5-4fb3-8a9c-6607fa41ed20
Setor: Comercial
Salário: R$23000,00
Cargo: Chefe

```

Figura 16. Consulta do menor e maior salário da equipe.

4.2. Menu dos Setores

Este menu tem seu objetivo fazer demonstrativos e consultas dos setores existentes verificando qual setor tem mais ou menos funcionários dentro da Empresa. Também, nesta sessão, é feita a consulta das sessões existentes com as suas respectivas informações, assim como o custo de pessoal no setor com base na soma de todos os salários dos funcionários. Também mostra a média salarial do setor e por fim o nome do chefe de setor.

Para garantir o funcionamento no período de testes, o programa já tem em sua base de dados setores cadastrados.

```

Escolha sua opção:
1 - Mostrar o setor com mais e o setor com menos funcionários
2 - Consultar informações de um setor

Sua escolha: 1

Departamento com maior número de funcionários: Administrativo
Número de funcionários: 1
Departamento com maior numero de funcionários: Comercial
Número de funcionários: 2

Deseja voltar ao menu principal? (S/N)

```

Figura 17. Menu de Setores - Mostrar o setor com mais e o setor com menos funcionários RHTec.

```
Escolha sua opção:
1 - Mostrar o setor com mais e o setor com menos funcionários
2 - Consultar informações de um setor

Sua escolha: 2

Escolha o setor a ser consultado
1. ADMINISTRATIVO
2. COMERCIAL
3. FINANCEIRO
4. RH
5. OPERACIONAL
2

Setor: Comercial
Número de funcionários: 2
Salário total: R$24000,00
Média de salário no setor: R$12000,00
Chefe do setor: Paulo Souza
|
Deseja voltar ao menu principal? (S/N)
```

Figura 18. Menu de Setores - Consultar informações de um setor. Exemplo: Setor Comercial .

Durante o processo de testes do sistema de gerenciamento de pessoas, as suas respectivas funções atenderam as expectativas planejadas garantindo o pleno funcionamento em caso concreto.

5. Precificação

Fazer um software é caro, demanda esforço e disponibilidade da equipe para entregar o melhor ao cliente, logo, nossa prática de desenvolvimento leva a questão base de que não daremos um preço final ao cliente, mas junto a ele deixaremos claro que iremos desenvolver e ao decorrer da produção os valores irão aumentando gradualmente e assim, ao conseguir ultrapassar os possíveis desafios que podem surgir em meio a sua construção, ele estará a par da maioria do esforço tido pela equipe para seu produto.

A determinação do preço do software é analisada com base no seu custo de produção e manutenção do produto, bem como as demais despesas fixas e variáveis agregadas em geral a equipe desenvolvedora. No entanto, tendo em vista que o desenvolvimento mostrado em questão trata-se de um MVP, podemos por um valor inicial de cinco mil reais, posto todos os benefícios e importantes mudanças dispostas a agregar na empresa de nosso cliente.

6. Conclusão

Com as tecnologias desenvolvidas torna-se possível a criação do RHTec, sendo este projeto passível de melhoramentos futuros, como adição de interface gráfica, de forma a tornar a experiência do usuário muito mais agradável e um sistema cada vez mais completo. Também pode se pensar em acrescentar análise mais complexa de questões da empresa, dentre outras funcionalidades.

Apêndice A – Código Fonte

Conclui-se, com base no código fonte (apêndice A), que o sistema proposto tem as Classes Main, Empresa, Funcionario, Contrato, enum Setor mostrados anteriormente no diagrama de classes.

Classe Main

```
import java.util.Scanner;
import java.util.Locale;
import java.util.ArrayList;

public class Main {

    //o menu geral chama dois outros menus (funcionários e setores) de acordo com a
    //escolha do usuário
    private static void menuGeral(Empresa empresa) {

        Scanner scn = new Scanner(System.in);
        int opc;
        String voltar;
        boolean sair = false;

        while(!sair) {

            System.out.println("Escolha sua opção:");
            System.out.println("1 - Acessar Menu de Funcionários");
            System.out.println("2 - Acessar Menu de Setores");
            System.out.println("3 - Sair");

            System.out.printf("%nSua escolha: ");
            opc = Integer.parseInt(scn.nextLine());

            switch (opc) {
                case 1:
                    menuFuncionarios(empresa);

                    System.out.println("\nDeseja voltar ao menu principal? (S/N)");

                    while (true) {
                        voltar = scn.nextLine().toUpperCase(Locale.ROOT);
                        if (voltar.equals("S")) {
                            break;
                        } else if (voltar.equals("N")) {
                            sair = true;
                            break;
                        } else {
                            System.out.println("Valor inválido, insira novamente");
                        }
                    }
                    break;

                case 2:
                    menuSetores(empresa);

                    System.out.println("\nDeseja voltar ao menu principal? (S/N)");

                    while (true) {
                        voltar = scn.nextLine().toUpperCase(Locale.ROOT);
                        if (voltar.equals("S")) {
                            break;
                        } else if (voltar.equals("N")) {
                            sair = true;
                            break;
                        } else {
                            System.out.println("Valor inválido, insira novamente");
                        }
                    }

                    break;

                case 3:
                    sair = true;
            }
        }
    }
}
```

```

        break;

        default:
            System.out.println("Opção inválida, selecione novamente\n");
    }

}

}

//contém as opções para funcionários como contratar, demitir, etc.
private static void menuFuncionarios(Empresa empresa) {
    Scanner sc = new Scanner(System.in);
    int opc;
    String termo, cpf;

    System.out.println("\n\nEscolha sua opção:");
    System.out.println("1 - Cadastrar um novo funcionário");
    System.out.println("2 - Consultar dados de um funcionário");
    System.out.println("3 - Atualizar dados de um funcionário");
    System.out.println("4 - Demitir um funcionário");
    System.out.println("5 - Mostrar lista de funcionários");
    System.out.println("6 - Mostrar os funcionários com o maior e o menor salário");

    System.out.printf("%nSua escolha: ");
    opc = Integer.parseInt(sc.nextLine());
    System.out.println("\n");

    switch (opc) {
        case 1:
            empresa.funcionarioNovo();
            break;

        case 2:
            System.out.println("Digite o nome ou cpf do funcionário que deseja consultar.");
            termo = sc.nextLine();
            Contrato contr = empresa.funcionarioConsulta(termo);

            System.out.println();

            if (contr.getFuncionario() == null){
                System.out.println("Funcionário não encontrado");
            }else{
                empresa.funcionarioImprime(contr);
            }

            break;

        case 3:
            System.out.println("Digite o cpf do funcionário que deseja atualizar.");
            cpf = sc.nextLine();
            System.out.println();
            empresa.funcionarioAtualiza(cpf);
            break;

        case 4:
            System.out.println("Digite o cpf do funcionário que deseja demitir.");
            cpf = sc.nextLine();
            empresa.cpfDemitir(cpf);
            break;

        case 5:
            menuLista(empresa);
            break;

        case 6:

```

```

        Contrato[] MenorMaior = empresa.funcionarioIdentifica();
        System.out.println("Funcionário com menor salário:");
        empresa.funcionarioImprime(MenorMaior[0]);

        System.out.println("\nFuncionário com maior salário:");
        empresa.funcionarioImprime(MenorMaior[1]);
        break;

    default:
        System.out.println("Opção inválida");
    }
}

//é chamado pelo menu de Funcionários
//serve para visualizar a lista geral de funcionários ou as listas por sexo, estado
ou setor
private static void menuLista(Empresa empresa) {
    Scanner sc = new Scanner(System.in);
    int opc;
    boolean valido = true;

    do {

        System.out.println("Escolha sua opção:");
        System.out.println("1 - Mostrar lista de todos os funcionários");
        System.out.println("2 - Mostrar lista de funcionários por estado");
        System.out.println("3 - Mostrar lista de funcionários por setor");
        System.out.println("4 - Mostrar lista de funcionários por sexo");

        System.out.printf("%nSua escolha: ");
        opc = Integer.parseInt(sc.nextLine());

        switch (opc) {
            case 1:
                empresa.funcionarioLista();
                break;

            case 2:
                System.out.println("Insira o estado");
                String estado = new String();
                estado = sc.nextLine();

                empresa.funcionarioLista(estado);
                break;

            case 3:

                System.out.println("Insira o setor\n1. ADMINISTRATIVO\n2. COMERCIAL\n3. FINANCEIRO\n4. RH\n5. OPERACIONAL");
                Setores setor = null; // para evitar erro de runtime

                while(true) {
                    int escolha = Integer.parseInt(sc.nextLine());
                    switch (escolha) {
                        case 1:
                            setor = Setores.ADMINISTRATIVO;
                            break;

                        case 2:
                            setor = Setores.COMERCIAL;
                            break;

                        case 3:
                            setor = Setores.FINANCEIRO;
                            break;

                        case 4:

```

```

        setor = Setores.RH;
        break;

    case 5:
        setor = Setores.OPERACIONAL;
        break;

    default:
        System.out.println("Setor inválido, selecione novamente"
            );
    }

    if (setor != null) {
        break;
    }
}

empresa.funcionarioLista(setor);
break;

case 4:

    System.out.println("Insira o sexo (M/F)");
    String sexo;
    while (true) {
        sexo = sc.nextLine().toUpperCase(Locale.ROOT);
        if (sexo.equals("F") || sexo.equals("M")) {
            break;
        } else {
            System.out.println("Valor inválido, insira novamente");
        }
    }
    empresa.funcionarioLista(sexo);
    break;

    default:
        valido = false;
        System.out.println("Opção inválida, selecione novamente\n");
    }

}while(!valido);
}

//contém as ações para ver informações de setores
private static void menuSetores(Empresa empresa) {

    Scanner sc = new Scanner(System.in);
    int opc;

    System.out.println("\n\nEscolha sua opção:");
    System.out.println("1 - Mostrar o setor com mais e o setor com menos funcioná
        rios");
    System.out.println("2 - Consultar informações de um setor");

    System.out.printf("%nSua escolha: ");
    opc = Integer.parseInt(sc.nextLine());
    System.out.println("\n");

    switch (opc) {
        case 1:
            empresa.setorIdentifica();
            break;

        case 2:
            empresa.setorConsulta();
            break;

        default:

```



```

        System.out.println("Opção inválida");
    }
}

public static void main(String[] args) {
    String[] enderecoEmp = {"Pará", "Belém", "Umarizal"};
    ArrayList<Contrato> listaContratos = new ArrayList<Contrato>();
    Empresa empresa = new Empresa("Empresa", enderecoEmp, "09199965222", "
        09203983777", "empresa@gmail.com", "João Alberto", listaContratos);

    //criação de indereços pra testes
    String[] teste1 = {"Pará", "Belém", "Umarizal"};
    String[] teste2 = {"Rio Janeiro", "Rio Janeiro", "Copacabana"};
    String[] teste3 = {"São Paulo", "São Paulo", "Jardins"};
    String[] teste4 = {"Pará", "Belém", "Marambaia"};
    String[] teste5 = {"Pará", "Belém", "Marco"};
    String[] teste6 = {"Pará", "Belém", "Guamá"};
    String[] teste7 = {"Pará", "Belém", "Cremação"};
    String[] teste8 = {"Pará", "Belém", "Pedreira"};
    String[] teste9 = {"Pará", "Belém", "Nazaré"};
    String[] teste10 = {"Pará", "Belém", "Reduto"};

    //criação de funcionários pra testes
    Funcionario teste1Func = new Funcionario("Albert", "Souza", teste1, 20, 'M', "
        03399288201", "asouza@gmail.com", "09198867673");
    Funcionario teste2Func = new Funcionario("Luciano", "Lima", teste2, 20, 'M', "
        54284839497", "l1limao@gmail.com", "09198392989");
    Funcionario teste3Func = new Funcionario("Gabriel", "Sales", teste3, 20, 'M', "
        27483903006", "gsales@gmail.com", "09199999993");
    Funcionario teste4Func = new Funcionario("Lucas", "Oliveira", teste4, 20, 'M', "
        98948938499", "loliveira@gmail.com", "09192442473");
    Funcionario teste5Func = new Funcionario("Paulo", "Souza", teste5, 20, 'M', "
        73848878783", "psouza@gmail.com", "09198934984");
    Funcionario teste6Func = new Funcionario("Julia", "Ramos", teste6, 20, 'F', "
        66382782882", "jramos@gmail.com", "09198888883");
    Funcionario teste7Func = new Funcionario("Gabriely", "Santos", teste7, 20, 'F', "
        98398928393", "gsantosb@gmail.com", "09189928292");
    Funcionario teste8Func = new Funcionario("Bruna", "Rodrigues", teste8, 20, 'F', "
        74783478378", "brodrigues@gmail.com", "09199993333");
    Funcionario teste9Func = new Funcionario("Ana", "Ferreira", teste9, 20, 'F', "
        56233665211", "aferreira@gmail.com", "09198348378");
    Funcionario teste10Func = new Funcionario("Clara", "Alves", teste10, 20, 'F', "
        52899993-22", "calves@gmail.com", "09199383383");

    //criação de contratos pra testes
    Contrato teste1Cont = new Contrato(teste1Func, Setores.ADMINISTRATIVO, 1700000, "
        Chefe");
    Contrato teste2Cont = new Contrato(teste2Func, Setores.ADMINISTRATIVO, 200000, "
        Colaborador");
    Contrato teste3Cont = new Contrato(teste3Func, Setores.RH, 1200000, "Chefe");
    Contrato teste4Cont = new Contrato(teste4Func, Setores.RH, 170000, "Colaborador"
    );
    Contrato teste5Cont = new Contrato(teste5Func, Setores.COMERCIAL, 2300000, "
        Chefe");
    Contrato teste6Cont = new Contrato(teste6Func, Setores.COMERCIAL, 100000, "
        Colaborador");
    Contrato teste7Cont = new Contrato(teste7Func, Setores.FINANCEIRO, 1500000, "
        Chefe");
    Contrato teste8Cont = new Contrato(teste8Func, Setores.FINANCEIRO, 120000, "
        Colaborador");
    Contrato teste9Cont = new Contrato(teste9Func, Setores.OPERACIONAL, 1900000, "
        Chefe");
    Contrato teste10Cont = new Contrato(teste10Func, Setores.OPERACIONAL, 130000, "
        Colaborador");

    //adicionar os contratos na lista de contratos da empresa
    empresa.funcionarioNovo(teste1Cont);
}

```

```

        empresa.funcionarioNovo(teste2Cont);
        empresa.funcionarioNovo(teste3Cont);
        empresa.funcionarioNovo(teste4Cont);
        empresa.funcionarioNovo(teste5Cont);
        empresa.funcionarioNovo(teste6Cont);
        empresa.funcionarioNovo(teste7Cont);
        empresa.funcionarioNovo(teste8Cont);
        empresa.funcionarioNovo(teste9Cont);
        empresa.funcionarioNovo(teste10Cont);

        menuGeral(empresa);
    }
}

```

Classe Empresa

```

import java.util.ArrayList;
import java.util.Locale;
import java.util.Scanner;

public class Empresa {
    private String nome;
    private String[] endereco;
    private String telefone;
    private String cnpj;
    private String email;
    private String dono;
    private ArrayList<Contrato> listaContratos = new ArrayList<Contrato>();

    //construtor padrão
    public Empresa(String nome, String[] endereco, String telefone, String cnpj, String
        email, String dono, ArrayList<Contrato> listaContratos) {
        this.nome = nome;
        this.endereco = endereco;
        this.telefone = telefone;
        this.cnpj = cnpj;
        this.email = email;
        this.dono = dono;
        this.listaContratos = listaContratos;
    }

    //construtor genérico
    public Empresa() {
    }

    //pede os dados do funcionário para o usuário do sistema, instancia Funcionario e
    Contrato e depois adiciona na lista de contratos
    public void funcionarioNovo() {
        Scanner input = new Scanner(System.in);

        System.out.println("Bem vind@ ao assistente de cadastro de funcionário");
        System.out.println("Insira o nome do funcionário a ser cadastrado");
        String[] nome = new String[2];
        nome[0] = input.nextLine();

        System.out.println("Insira o sobrenome do funcionário a ser cadastrado");
        nome[1] = input.nextLine();

        System.out.println("Insira o estado o funcionário mora");
        String[] endereco = new String[3];
        endereco[0] = input.nextLine();
        System.out.println("Insira a cidade o funcionário mora");
        endereco[1] = input.nextLine();
        System.out.println("Insira o bairro o funcionário mora");
    }
}

```

```

endereco[2] = input.nextLine();

System.out.println("Insira a idade do funcionário");
int idade;
while (true) { // uso de loops while(true) para receber inputs até que um input
    válido seja recebido
    idade = Integer.parseInt(input.nextLine());
    if (idade >= 18 && idade < 100) {
        break;
    } else {
        System.out.println("Valor inválido, insira novamente");
    }
}

System.out.println("Insira o sexo do funcionário (M/F)");
String sexo;
while (true) {
    sexo = input.nextLine().toUpperCase(Locale.ROOT);
    if (sexo.equals("F") || sexo.equals("M")) {
        break;
    } else {
        System.out.println("Valor inválido, insira novamente");
    }
}

System.out.println("Insira o cpf do funcionário (somente números)");
String cpf;
while (true) {
    cpf = input.nextLine();
    if (cpf.length() == 11 && cpf.matches("[0-9]+")) {
        break;
    } else {
        System.out.println("Cpf inválido, digite 11 números");
    }
}

System.out.println("Insira o email do funcionário");
String email = input.nextLine();

System.out.println("Insira o telefone do funcionário (somente números)");
String telefone;
while (true) {
    telefone = input.nextLine();
    if (telefone.length() == 11 && telefone.matches("[0-9]+")) {
        break;
    } else {
        System.out.println("Telefone inválido, digite 11 números");
    }
}

Funcionario funcionario = new Funcionario(nome[0], nome[1], endereco, idade,
    sexo.charAt(0), cpf, email, telefone);

System.out.println("Insira o setor do funcionário\n1. ADMINISTRATIVO\n2.
    COMERCIAL\n3. FINANCEIRO\n4. RH\n5. OPERACIONAL");
Setores setor = null; // para evitar erro de runtime

while(true) {
    int escolha = Integer.parseInt(input.nextLine());
    switch (escolha) {
        case 1:
            setor = Setores.ADMINISTRATIVO;
            break;

        case 2:
            setor = Setores.COMERCIAL;
            break;

        case 3:
            setor = Setores.FINANCEIRO;

```

```

        break;

    case 4:
        setor = Setores.RH;
        break;

    case 5:
        setor = Setores.OPERACIONAL;
        break;

    default:
        System.out.println("Setor inválido, selecione novamente");
    }

    if (setor != null) {
        break;
    }
}

System.out.println("Digite o salário do funcionário");
double salario = Double.parseDouble(input.nextLine());

System.out.println("Escolha o cargo do funcionário:\n1. Colaborador\n2. Chefe");
int escolha;
String cargo;
while (true) {
    escolha = Integer.parseInt(input.nextLine());

    if (escolha == 2) {
        boolean valido = true;
        for (Contrato c : this.listaContratos) {
            if (c.getSetor().equals(setor) && c.getCargo().equals("Chefe")) {
                valido = false;
                break;
            }
        }

        if (valido) {
            cargo = "Chefe";
            break;
        } else {
            System.out.println("Cargo inválido, já existe um chefe neste setor");
        }
    } else if (escolha == 1) {
        cargo = "Colaborador";
        break;
    } else {
        System.out.println("Opção inválida, escolha novamente");
    }
}

Contrato contrato = new Contrato(funcionario, setor, (int) salario*100, cargo);
this.listaContratos.add(contrato);
}

//similar à funcionarioNovo(), porém com a opção de escolher quais dados atualizar
public void funcionarioAtualiza(String cpf) {
    Scanner input = new Scanner(System.in);
    Contrato contrato = new Contrato();

    for (Contrato c : this.listaContratos) {
        if (c.getFuncionario().getCpf().equals(cpf)) {
            contrato = c;
        }
    }

    if (contrato.getFuncionario() == null) {
        System.out.println("Funcionário não encontrado");
    }
}

```

```

    } else {
        while (true) {
            System.out.println("Escolha o dado a ser atualizado: ");
            System.out.println("1. Nome e sobrenome");
            System.out.println("2. Endereço");
            System.out.println("3. Idade");
            System.out.println("4. Sexo");
            System.out.println("5. Cpf");
            System.out.println("6. Email");
            System.out.println("7. Telefone");
            System.out.println("8. Setor");
            System.out.println("9. Salário");
            System.out.println("10. Cargo");
            System.out.println("11. Sair");

            int escolha = Integer.parseInt(input.nextLine());

            switch (escolha) {
                case 1:
                    System.out.println("Insira o nome atualizado do funcionário");
                    contrato.getFuncionario().setNome(input.nextLine());
                    System.out.println("Insira o sobrenome atualizado do funcionário");
                    contrato.getFuncionario().setSobrenome(input.nextLine());
                    break;

                case 2:
                    String[] endereco = new String[3];
                    System.out.println("Insira o estado onde o funcionário mora");
                    endereco[0] = input.nextLine();
                    System.out.println("Insira a cidade onde o funcionário mora");
                    endereco[1] = input.nextLine();
                    System.out.println("Insira o bairro onde o funcionário mora");
                    endereco[2] = input.nextLine();
                    contrato.getFuncionario().setEndereco(endereco[0], endereco[1],
                        endereco[2]);
                    break;

                case 3:
                    System.out.println("Insira a idade do funcionário");
                    int idade = Integer.parseInt(input.nextLine());
                    if (idade > 18 && idade < 100) {
                        contrato.getFuncionario().setIdade(idade);
                    } else {
                        System.out.println("Idade invalida, tente novamente");
                    }
                    break;

                case 4:
                    System.out.println("Insira o sexo do funcionário (M ou F)");
                    String sexo = input.nextLine().toUpperCase(Locale.ROOT);
                    if (sexo.equals("F") || sexo.equals("M")) {
                        contrato.getFuncionario().setSexo(sexo.charAt(0));
                    } else {
                        System.out.println("Sexo invalido, tente novamente");
                    }
                    break;

                case 5:
                    System.out.println("Insira o cpf do funcionário (11 números)");
                    String newCpf = input.nextLine();
                    if (newCpf.matches("[0-9]+" ) && newCpf.length() == 11) {
                        contrato.getFuncionario().setCpf(newCpf);
                    } else {
                        System.out.println("Cpf invalido, tente novamente");
                    }
                    break;

                case 6:
                    System.out.println("Insira o email do funcionario");

```

```

        contrato.getFuncionario().setEmail(input.nextLine());
        break;

    case 7:
        System.out.println("Insira o telefone do funcionário (11 números)");
        String telefone = input.nextLine();
        if (telefone.matches("[0-9]+") && telefone.length() == 11) {
            contrato.getFuncionario().setTelefone(telefone);
        } else {
            System.out.println("Telefone invalido, tente novamente");
        }
        break;

    case 8:
        System.out.println("Insira o setor do funcionário\n1. ADMINISTRATIVO\n2. COMERCIAL\n3. FINANCEIRO\n4. RH\n5. OPERACIONAL");
        int setor = Integer.parseInt(input.nextLine());
        switch (setor) {
            case 1:
                contrato.setSetor(Setores.ADMINISTRATIVO);
                break;

            case 2:
                contrato.setSetor(Setores.COMERCIAL);
                break;

            case 3:
                contrato.setSetor(Setores.FINANCEIRO);
                break;

            case 4:
                contrato.setSetor(Setores.RH);
                break;

            case 5:
                contrato.setSetor(Setores.OPERACIONAL);
                break;

            default:
                System.out.println("Setor inválido, selecione novamente");
        }

    case 9:
        System.out.println("Insira o salário do funcionário");
        int salario = (int) Double.parseDouble(input.nextLine());
        salario = salario * 100;
        contrato.setSalario(salario);
        break;

    case 10:
        System.out.println("Escolha o cargo do funcionário:\n1. Colaborador\n2. Chefe");
        int cargo = Integer.parseInt(input.nextLine());
        String novoCargo;

        if (cargo == 2) {
            boolean valido = true;
            for (Contrato c : this.listaContratos) {
                if (c.getSetor().equals(contrato.getSetor()) && c.getCargo().equals("Chefe")) {
                    valido = false;
                    break;
                }
            }

            if (valido) {
                contrato.setCargo("Chefe");
            }
        }
    }
}

```

```

        } else {
            System.out.println("Cargo inválido, já existe um chefe
                               neste setor");
        }

        } else if (cargo == 1) {
            contrato.setCargo("Colaborador");
        } else {
            System.out.println("Opção inválida, escolha novamente");
        }

        break;

    case 11:
        return; //sai do metodo com return pois break faz parte da
                sintaxe do switch-case

    default:
        System.out.println("Opção inválida, escolha novamente\n");
    }
}

//lista todos os funcionários da empresa
public void funcionarioLista() {
    for (Contrato contrato : this.listaContratos) {
        this.funcionarioImprime(contrato);
    }
}

//lista todos os funcionários do setor que recebe
public void funcionarioLista(Setores setor) {
    for (Contrato contrato : this.listaContratos) {
        if (contrato.getSetor().equals(setor)) {
            this.funcionarioImprime(contrato);
        }
    }
}

//lista todos os funcionários do sexo que recebe
public void funcionarioLista(char sexo) {
    for (Contrato contrato : this.listaContratos) {
        if (contrato.getFuncionario().getSexo() == sexo) {
            this.funcionarioImprime(contrato);
        }
    }
}

//lista todos os funcionários do estado que recebe
public void funcionarioLista(String estado) {
    for (Contrato contrato : this.listaContratos) {
        if (contrato.getFuncionario().getEndereco()[0].equals(estado)) {
            this.funcionarioImprime(contrato);
        }
    }
}

//identifica o funcionário com menor e maior salário e os retorna num array Contrato
[2]
public Contrato[] funcionarioIdentifica() {
    Contrato menorSalario = new Contrato();
    Contrato maiorSalario = new Contrato();
    menorSalario.setSalario(Integer.MAX_VALUE);
    maiorSalario.setSalario(-1);
    for (Contrato c : this.listaContratos) {
        if (c.getSalario() < menorSalario.getSalario()) {
            menorSalario = c;
        }
    }
}

```

```

        if (c.getSalario() > maiorSalario.getSalario()) {
            maiorSalario = c;
        }
    }

    return new Contrato[]{menorSalario, maiorSalario};
}

//consulta um funcionário com base no nome ou no cpf
public Contrato funcionarioConsulta(String termo) {
    Contrato resultado = new Contrato();

    if (termo.matches("[A-Za-z]+")) { //se o termo for alfabético, consulta por nome
        for (Contrato c : this.listaContratos) {
            if (c.getFuncionario().getNome().equals(termo)) {
                resultado = c;
            }
        }
    } else if (termo.matches("[0-9]+")) { //se for numérico, consulta por cpf
        for (Contrato c : this.listaContratos) {
            if (c.getFuncionario().getCpf().equals(termo)) {
                resultado = c;
            }
        }
    }

    return resultado;
}

//demite um funcionário dado seu cpf
public void cpfDemitir(String cpf) {
    Contrato aDemitir = funcionarioConsulta(cpf);

    if (aDemitir.getFuncionario() == null) {
        System.out.println("Funcionário não encontrado");
    } else {
        this.listaContratos.remove(aDemitir);
    }
}

//identifica qual setores tem menos e mais funcionários, imprime o salário total dos
setores
public void setorIdentifica() {
    int[] numFuncionarios = {0, 0, 0, 0, 0}; // [0]: administrativo, [1]: comercial,
        [2]: financeiro, [3]: rh, [4]: operacional

    for (Contrato c : this.listaContratos) {
        switch (c.getSetor()) {
            case ADMINISTRATIVO:
                numFuncionarios[0]++;
                break;

            case COMERCIAL:
                numFuncionarios[1]++;
                break;

            case FINANCEIRO:
                numFuncionarios[2]++;
                break;

            case RH:
                numFuncionarios[3]++;
                break;

            case OPERACIONAL:
                numFuncionarios[4]++;
                break;
        }
    }
}

```



```

int menor = Integer.MAX_VALUE;
int maior = -1;
int menorIndex = -1;
int maiorIndex = 5;

for (int i = 0; i < numFuncionarios.length; i++) {
    if (numFuncionarios[i] > maior) {
        maior = numFuncionarios[i];
        maiorIndex = i;
    }

    if (numFuncionarios[i] < menor) {
        menor = numFuncionarios[i];
        menorIndex = i;
    }
}

System.out.print("Departamento com maior número de funcionários: ");
switch (menorIndex) {
    case 0:
        System.out.printf("Administrativo\nNúmero de funcionários: %d\n", menor);
        ;
        break;

    case 1:
        System.out.printf("Comercial\nNúmero de funcionários: %d\n", menor);
        break;

    case 2:
        System.out.printf("Financeiro\nNúmero de funcionários: %d\n", menor);
        break;

    case 3:
        System.out.printf("Rh\nNúmero de funcionários: %d\n", menor);
        break;

    case 4:
        System.out.printf("Operacional\nNúmero de funcionários: %d\n", menor);
        break;
}

System.out.print("Departamento com maior numero de funcionários: ");
switch (maiorIndex) {
    case 0:
        System.out.printf("Administrativo\nNúmero de funcionários: %d\n", maior);
        ;
        break;

    case 1:
        System.out.printf("Comercial\nNúmero de funcionários: %d\n", maior);
        break;

    case 2:
        System.out.printf("Financeiro\nNúmero de funcionários: %d\n", maior);
        break;

    case 3:
        System.out.printf("Rh\nNúmero de funcionários: %d\n", maior);
        break;

    case 4:
        System.out.printf("Operacional\nNúmero de funcionários: %d\n", maior);
        break;
}
}

//imprime todos os dados do funcionário recebido
public void funcionarioImprime(Contrato contrato) {
    System.out.printf("Nome: %s %s\n", contrato.getFuncionario().getNome(), contrato
        .getFuncionario().getSobrenome());
}

```

```

        System.out.printf("Cpf: %s\n", contrato.getFuncionario().getCpf());
        System.out.printf("Idade: %d\n", contrato.getFuncionario().getIdade());
        System.out.printf("Sexo: %c\n", contrato.getFuncionario().getSexo());
        System.out.printf("Endereço: %s, %s, %s\n", contrato.getFuncionario().
            getEndereco()[0], contrato.getFuncionario().getEndereco()[1], contrato.
            getFuncionario().getEndereco()[2]);
        System.out.printf("Telefone: %s\n", contrato.getFuncionario().getTelefone());
        System.out.printf("Email: %s\n", contrato.getFuncionario().getEmail());
        System.out.printf("Id: %s\n", contrato.getId());
        System.out.printf("Setor: %s\n", contrato.getSetor().getSetor());
        System.out.printf("Salário: R$%.2f\n", (double) contrato.getSalario()/100);
        System.out.printf("Cargo: %s\n", contrato.getCargo());
    }

    //consulta um setor baseado na escolha do usuário, imprime o nome, salário total e m
    édio, chefe do setor e numero de funcionários
    public void setorConsulta() {
        Scanner input = new Scanner(System.in);
        Setores setorEscolhido = null;

        System.out.println("Escolha o setor a ser consultado\n1. ADMINISTRATIVO\n2.
            COMERCIAL\n3. FINANCEIRO\n4. RH\n5. OPERACIONAL");
        int escolha;

        while (true) {
            escolha = Integer.parseInt(input.nextLine());

            switch (escolha) {
                case 1:
                    setorEscolhido = Setores.ADMINISTRATIVO;
                    break;

                case 2:
                    setorEscolhido = Setores.COMERCIAL;
                    break;

                case 3:
                    setorEscolhido = Setores.FINANCEIRO;
                    break;

                case 4:
                    setorEscolhido = Setores.RH;
                    break;

                case 5:
                    setorEscolhido = Setores.OPERACIONAL;
                    break;

                default:
                    System.out.println("Setor inválido, escolha novamente");
            }

            if (setorEscolhido != null) {
                break;
            }
        }

        int numFunc = 0;
        int salarioTotal = 0;
        Contrato chefe = new Contrato();

        for (Contrato c : this.listaContratos) { //procura o chefe do setor enquanto
            conta o número de funcionários
            if (c.getSetor().equals(setorEscolhido)) {
                numFunc++;
                salarioTotal += c.getSalario();

                if (c.getCargo().equals("Chefe")) {
                    chefe = c;
                }
            }
        }
    }

```

```

    }

    Funcionario ch = chefe.getFuncionario();

    System.out.printf("Setor: %s\n", setorEscolhido.getSetor());
    System.out.printf("Número de funcionários: %d\n", numFunc);
    System.out.printf("Salário total: R$%.2f\n", (double) salarioTotal/100);
    System.out.printf("Média de salário no setor: R$%.2f\n", (double) (salarioTotal
        /100)/numFunc);
    System.out.printf("Chefe do setor: %s %s\n", ch.getNome(), ch.getSobrenome());

}

public void funcionarioNovo(Contrato contrato) {
    this.listaContratos.add(contrato);
}
}

```

Classe Funcionario

```

public class Funcionario {
    private String nome;
    private String sobrenome;
    private String[] endereco = new String[3];
    private int idade;
    private char sexo;
    private String cpf;
    private String email;
    private String telefone;

    // construtor padrão
    public Funcionario(String nome, String sobrenome, String[] endereco, int idade, char
        sexo, String cpf, String email, String telefone) {
        this.nome = nome;
        this.sobrenome = sobrenome;
        this.endereco = endereco;
        this.idade = idade;
        this.sexo = sexo;
        this.cpf = cpf;
        this.email = email;
        this.telefone = telefone;
    }

    public Funcionario() { //construtor genérico
    }

    //getters e setters

    public String getNome() {
        return nome;
    }

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

    public String getSobrenome() {
        return sobrenome;
    }

    public void setSobrenome(String sobrenome) {
        this.sobrenome = sobrenome;
    }

    public String[] getEndereco() {
        return endereco;
    }
}

```

```

    }

    public void setEndereco(String estado, String cidade, String bairro) {
        this.endereco[0] = estado;
        this.endereco[1] = cidade;
        this.endereco[2] = bairro;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public char getSexo() {
        return sexo;
    }

    public void setSexo(char sexo) {
        this.sexo = sexo;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}

```

Classe Contratos

```

import java.util.UUID;

public class Contrato {
    private Funcionario funcionario;
    private Setores setor;
    private UUID id;
    private int salario; //salário do funcionário guardado em centavos, para evitar
        erros de arredondamento
    private String cargo;

    public Contrato(Funcionario funcionario, Setores setor, int salario, String cargo) {
        // construtor padrão
        this.funcionario = funcionario;
        this.setor = setor;
        this.id = UUID.randomUUID();
        this.salario = salario;
        this.cargo = cargo;
    }
}

```

```

    }

    public Contrato() { //construtor genérico para inicialização de variáveis

    }

    //getters e setters

    public Funcionario getFuncionario() {
        return funcionario;
    }

    public Setores getSetor() {
        return setor;
    }

    public void setSetor(Setores setor) {
        this.setor = setor;
    }

    public String getId() {
        return id.toString();
    }

    public void setId(UUID id) {
        this.id = id;
    }

    public void newId() {
        this.id = UUID.randomUUID();
    }

    public int getSalario() {
        return salario;
    }

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

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }
}

```

Enum Setores

```

public enum Setores { // enumareador usado como controle do setor de cada contrato
    ADMINISTRATIVO("Administrativo"),
    COMERCIAL("Comercial"),
    FINANCEIRO("Financeiro"),
    RH("Rh"),
    OPERACIONAL("Operacional");

    private final String setor; // "conversao" de tipo Setores pra String

    Setores(String setor) { // construtor padrão
        this.setor = setor;
    }

    //getter

    public String getSetor() {
        return this.setor;
    }
}

```

