

1. **SUBMISSÃO DA TAREFA REALIZADA COM SUCESSO. COMPROVANTE DE SUBMISSÃO Nº 3227405.**
2. **SUBMISSÃO DA TAREFA REALIZADA COM SUCESSO. COMPROVANTE DE SUBMISSÃO Nº 3230522.**
- 3.

Cadastramento Eleitoral

Flávio Seabra

Pacote View

- public class CadastramentoDeEleitores

- Main

- MainView view = new MainView();
 - view.startView();

O programa inicia chamando o startView()

- public interface View

- public class MainView implements View

- startView()

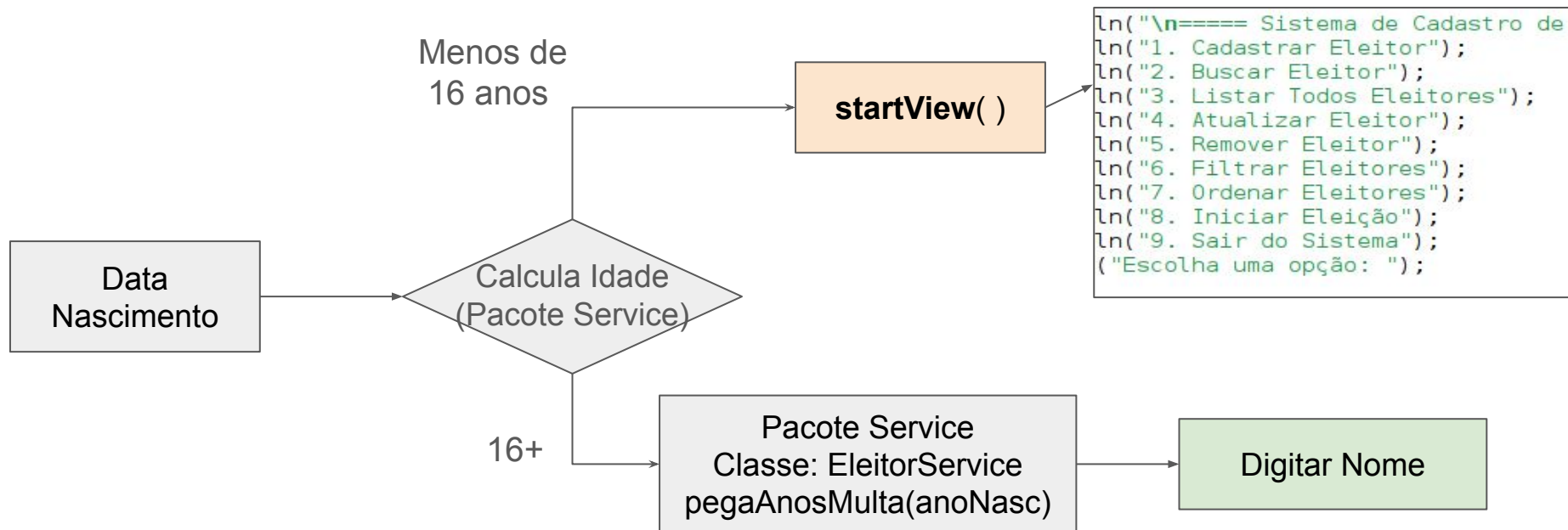
```
System.out.println("\n===== Sistema de Cadastro de  
System.out.println("1. Cadastrar Eleitor");  
System.out.println("2. Buscar Eleitor");  
System.out.println("3. Listar Todos Eleitores");  
System.out.println("4. Atualizar Eleitor");  
System.out.println("5. Remover Eleitor");  
System.out.println("6. Filtrar Eleitores");  
System.out.println("7. Ordenar Eleitores");  
System.out.println("8. Iniciar Eleição");  
System.out.println("9. Sair do Sistema");  
System.out.print("Escolha uma opção: ");
```

Pacote View

- public class MainView implements View
 - **public void startView()**
 - private void cadastrarEleitor()
 - private void buscarEleitorPorId()
 - private void listarTodosEleitores()
 - private void listarEleitoresParaAtualizarRemover()
 - private void atualizarEleitor()
 - private void removerEleitor()
 - private void filtrarEleitores()
 - private void buscarPorTitulo()
 - private void buscarPorCpf()
 - private void ordenarEleitores()
 - private void mostrarEleitores(List<Eleitor> eleitores)
 - public static void imprimeEleitor(Eleitor eleitor)
 - private void iniciarEleicao()

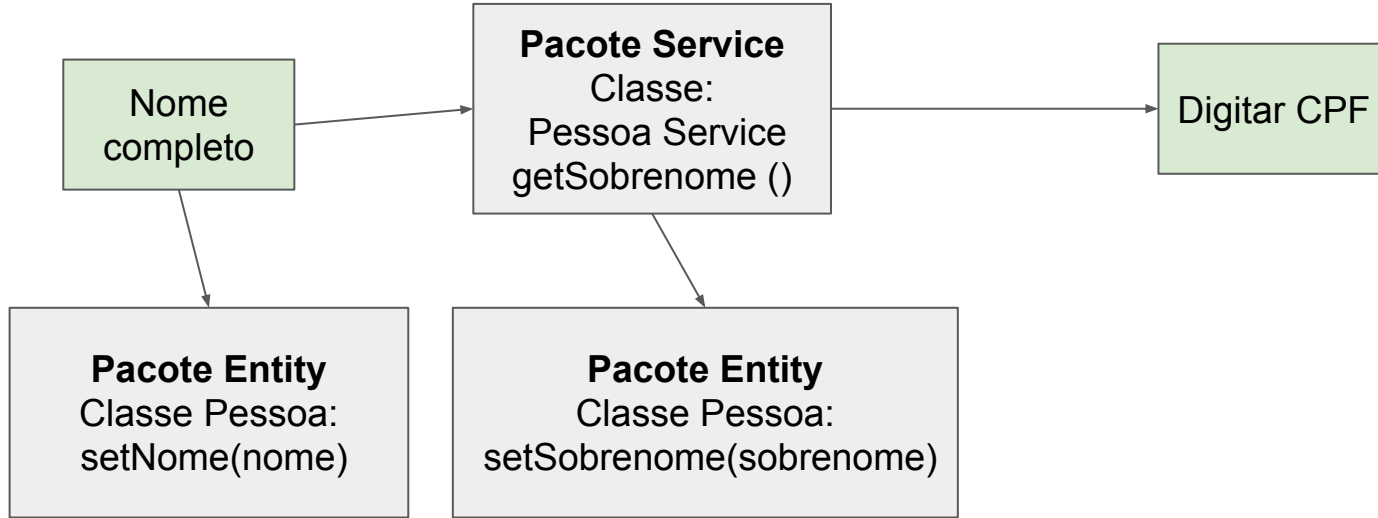
Pacote View

- public class MainView implements View
 - private void cadastrarEleitor()
 - Data de nascimento
 - dd/mm/aaaa - permite inserção com ou sem zeros (d/m/aaaa)



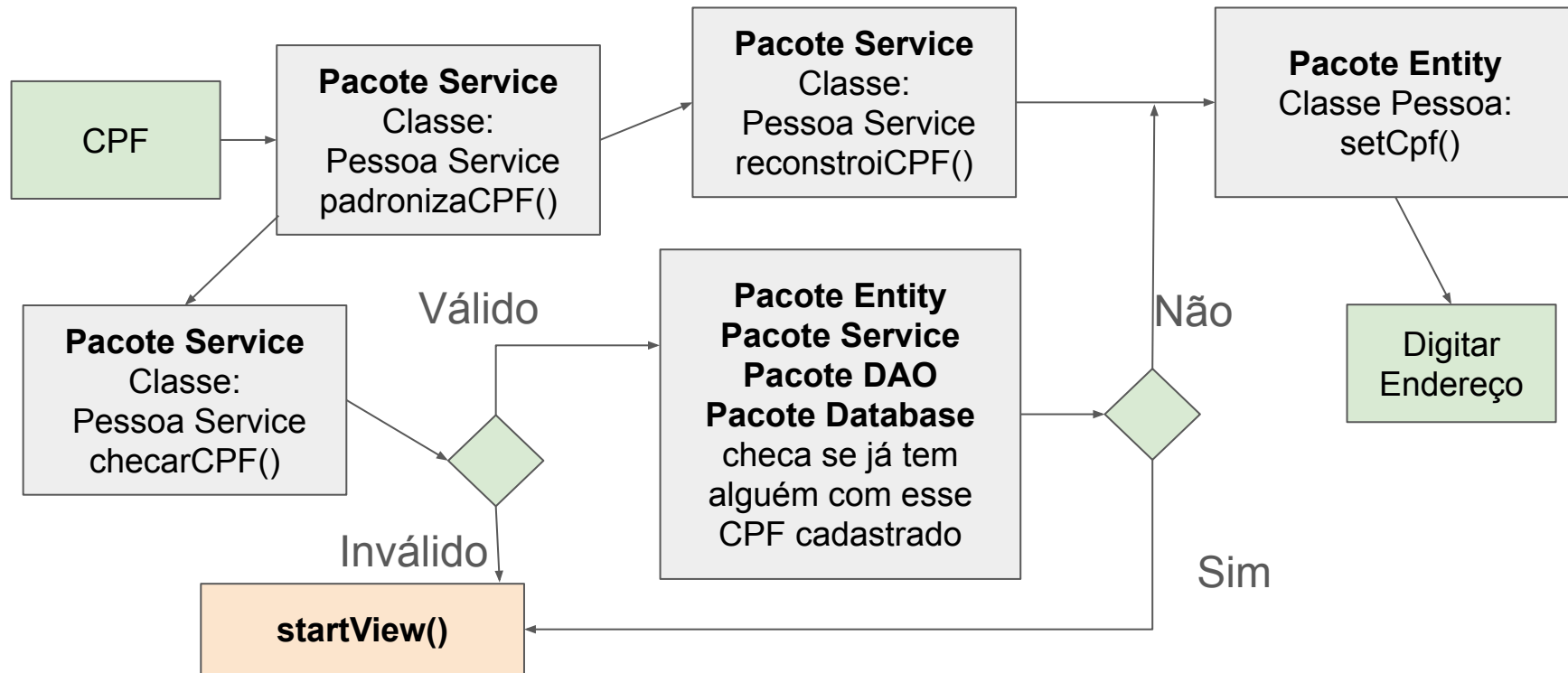
Pacote View

- public class MainView implements View
 - private void cadastrarEleitor()
 - Nome



Pacote View

- public class MainView implements View
 - private void cadastrarEleitor()
 - CPF



Pacote View

- public class MainView implements View
 - private void cadastrarEleitor()
 - Endereço

Pacote Entity
Classe:
Endereco
Rua
Número Bairro
Cidade

Pacote Service
Classe: EleitorService
geradorDeTitulo()
geradorDeSecao()
selecionaZona(cidade)

```
endereco.setRua(rua_digitada);
endereco.setNumero_compl(num_digitado);
endereco.setBairro(bairro_digitado);
endereco.setCidade(cidade_para_cadastrar);
endereco.setEstado("RN");

pessoa.setNome(nome);
pessoa.setSobrenome(sobrenome);
pessoa.setDataNascimento(dn);
pessoa.setCpf(cpf_reconstruido);
pessoa.setEndereco(endereco);

eleitor.setPessoa(pessoa);
eleitor.setZonaEleitoral(zonaEleitoral);
eleitor.setSecaoEleitoral(secacaoEleitoral);
eleitor.setTituloEleitoral(tituloEleitoral);
eleitor.setMultas(0);
eleitor.setSituacao(true);

eleitor.setAnosSemVotar(anosMultas);
float valor_multa = (float) 3.51 * anosMultas.size();

eleitor.setSituacao(false);
eleitor.setMultas(valor_multa);
```


Inserir eleitores

- Checagens ao inserir
 - Idade - o sistema deve checar se o eleitor tem a idade mínima
 - Ao inserir a data de nascimento o sistema corrige se o usuário inserir sem os zeros nos dias e meses menores que 10
 - A idade é calculada e é feita a checagem. Se a pessoa não tiver pelo menos 16 anos o sistema retorna ao menu inicial
 - CPF
 - O sistema corrige caso seja digitado com pontos e traços ou apenas os números
 - Validade - faz o teste de validade do número do CPF
 - Duplicidade - deve checar se já existe alguém cadastrado com aquele mesmo CPF

Inserir eleitores

- Inserção do nome
 - O nome completo é inserido normalmente e o sistema pega o último nome e seta no atributo “String sobrenome” da classe pessoa para que esse sobrenome seja usado nas opções de filtragem depois

Inserir eleitores

- Atribui
 - Título - gera um título de eleitor no formato 0000 0000 0000
 - São gerados 3 números aleatórios de 4 dígitos e unidos em uma String separada por espaço. Caso um dos números aleatórios tenha menos de 4 dígitos, o número 0 é adicionado na frente
 - Seção - atribui uma seção aleatória entre 1 e 99
 - Zona Eleitoral- escolhe a zona eleitoral de acordo com a cidade informada no endereço
 - Zona Eleitoral 10 para endereço de Natal
 - Zona Eleitoral 20 para endereço de Parnamirim
 - Zona Eleitoral 30 para endereço de Macaíba
 - O sistema corrige digitação da cidade apenas com minúscula ou sem o acento de macaíba.

Inserir eleitores

- Gera multa
 - O sistema verifica se o eleitor cadastrado já tinha 18 anos nas datas das últimas eleições (2022, 2020, 2018, etc.) e se sim gera uma multa por alistamento tardio no valor de R\$3,51 x N. de eleições em atraso avisando em quais eleições ele não votou
- Quitação
 - Caso o sistema não gere multa, o Eleitor será cadastrado como “Quite”, caso o sistema gere multa o Eleitor será cadastrado como “INATIVO” e assim permanecerá até o pagamento da multa

Filtrar Eleitor

- Filtrar pelo sobrenome
- Filtrar pela Zona Eleitoral

EleitorService

```
public List<Eleitor> filtrarEleitores(Predicate<Eleitor> predicate) {  
    return listarTodosEleitores().stream().filter(predicate).collect(Collectors.toList());  
}
```

MainView

```
case 1:  
    System.out.print("Sobrenome: ");  
    String sobrenome = scanner.nextLine();  
    List<Eleitor> eleitoresPorNome = eleitorService.filtrarEleitores(  
        eleitor -> eleitor.getPessoa().getSobrenome().equalsIgnoreCase(sobrenome));  
    mostrarEleitores(eleitoresPorNome);  
    break;  
case 2:  
    System.out.print("Zona Eleitoral (10, 20 ou 30): ");  
    int zonaEleitoral = scanner.nextInt();  
    scanner.nextLine(); // Consumir nova linha  
    List<Eleitor> eleitoresPorZona = eleitorService.filtrarEleitores(  
        eleitor -> eleitor.getZonaEleitoral() == zonaEleitoral);  
    mostrarEleitores(eleitoresPorZona);  
    break;
```

Buscar Eleitor

- Pelo ID
- Pelo CPF
- Pelo número do título

Se o eleitor buscado estiver com situação INATIVO o sistema avisa o valor da multa e oferece a opção de quitar a multa.

Caso o eleitor deseje quitar a multa a situação do eleitor é setada para Quite

Pacote View

- public class MainView implements View
 - private void buscarEleitorPorId()

Pacote Service

Classe: EleitorService

```
public Eleitor buscarEleitorPorId(int id) {  
    Eleitor eleitor = eleitorDAO.findById(id);  
    if (eleitor == null) {  
        throw new EleitorNotFoundException("Eleitor com ID " + id + " não encontrado.");  
    }  
    return eleitor;  
}
```

Pacote View

- public class MainView implements View
 - private void buscarEleitorPorId()

```
private void buscarEleitorPorId() {  
    System.out.print("ID do Eleitor: ");  
    int id = scanner.nextInt();  
    scanner.nextLine(); // consumir nova linha  
  
    Eleitor eleitor = eleitorService.buscarEleitorPorId(id);  
    if (eleitor != null) {  
        System.out.println("Eleitor encontrado:");  
        imprimeEleitor(eleitor);  
    }  
}
```

Pacote Service
Classe: EleitorService

```
public Eleitor buscarEleitorPorId(int id) {  
    Eleitor eleitor = eleitorDAO.findById(id);  
    if (eleitor == null) {  
        throw new EleitorNotFoundException("Eleitor com ID " + id + " não encontrado.");  
    }  
    return eleitor;  
}
```

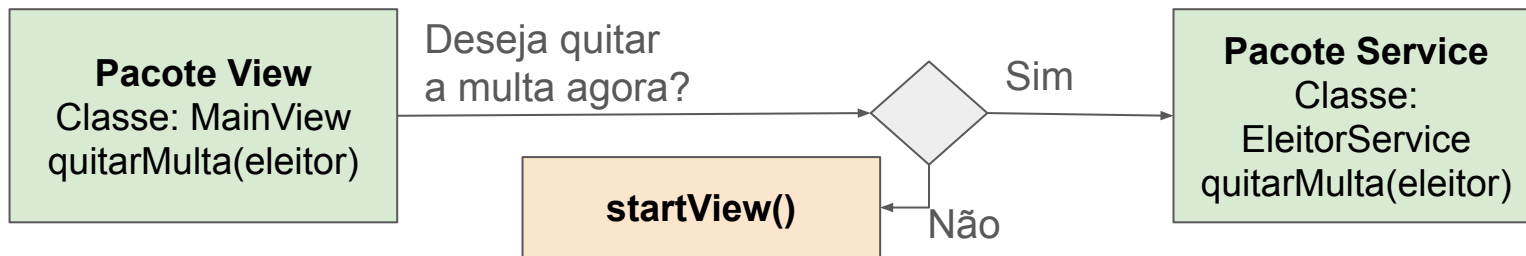

Pacote View

- public class MainView implements View
 - private void buscarPorTitulo() e buscarPorCpf()

```
//Separa o CPF em uma lista de char para testar validade
List<Character> elementos_CPF = PessoaService.padronizaCPF(cpf_buscar);
//Reconstrói CPF no formato correto para inserir no cadastro caso passe no teste de validade
String cpf_reconstruido = PessoaService.reconstróiCPF(elementos_CPF);

List<Eleitor> eleitoresPorNome = eleitorService.filtrarEleitores(
    eleitor -> eleitor.getPessoa().getCpf().equalsIgnoreCase(cpf_reconstruido));
mostrarEleitores(eleitoresPorNome);

if(eleitoresPorNome.get(0).getMultas() > 0){
    quitarMulta(eleitoresPorNome.get(0));
}
```



Listar eleitores

- Listar todos

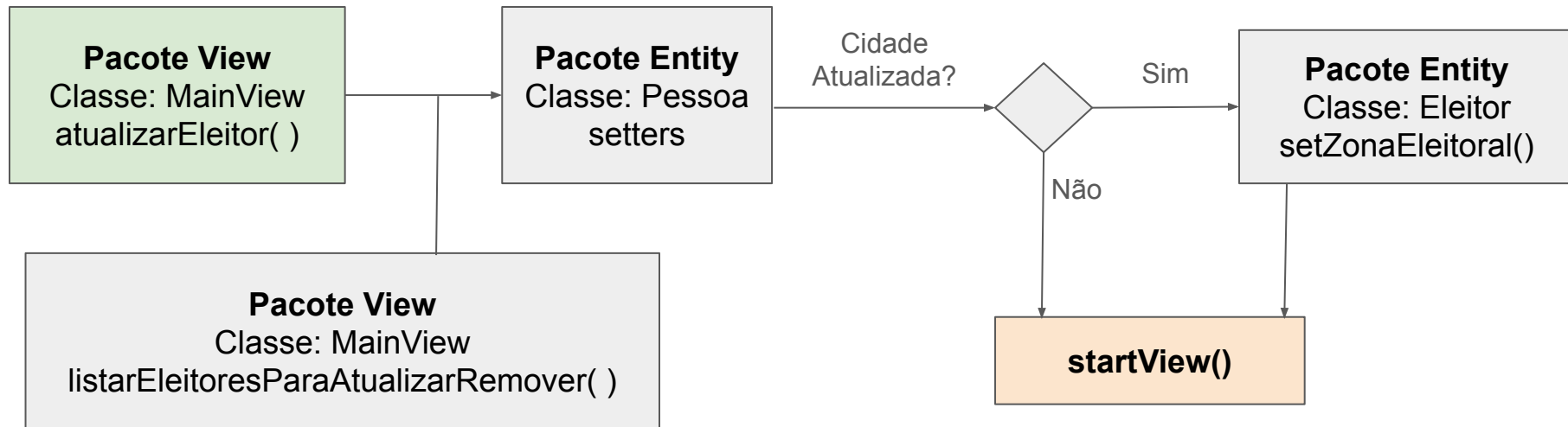
Pacote View

- public class MainView implements View
 - private void listarTodosEleitores()



Atualizar Eleitor

- Atualiza informações de eleitor já cadastrado
 - Pelo ID
 - É oferecida apenas a opção de atualizar o **nome** e o **endereço** já que uma vez cadastrado, o número do título, o CPF e o ID não devem mudar
 - Na mudança de endereço um novo número de Seção Eleitoral é gerado e, caso a mudança de endereço envolva mudança de município, uma nova Zona Eleitoral é setada.



Remover eleitor

- Remove eleitor do cadastro pelo ID

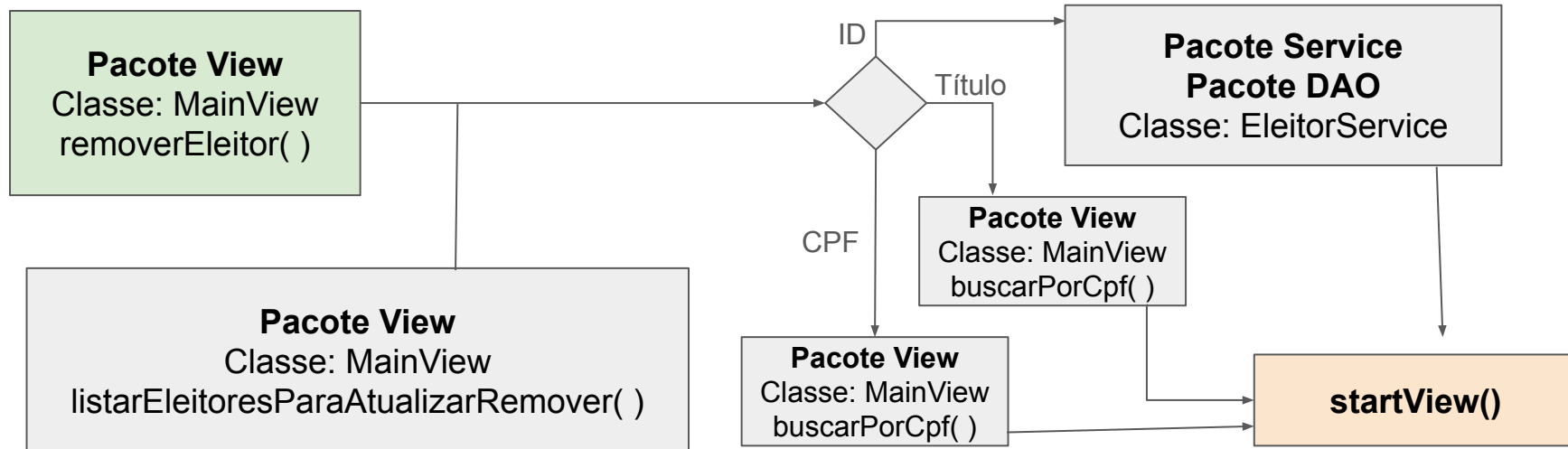
A remoção de um eleitor deve ser feita pelo ID, escolha uma opção:

Digite (1) se vc sabe o ID do eleitor que deseja remover

Digite (2) se vc sabe o CPF do eleitor que deseja remover. Anote o ID do eleitor que será exibido na tela.

Digite (3) se vc sabe o número do Título do eleitor que deseja remover. Anote o ID do eleitor que será exibido na tela.

Caso não saiba nenhuma dessas opções, digite (4) para retornar ao MENU inicial e depois a opção de 'Listar' os eleitores




Ordenar Eleitores

- Imprime ordenado pelo nome
- Imprime ordenado pela Zona Eleitoral

Pacote Service

Classe: EleitorService

```
public List<Eleitor> ordenarEleitores(java.util.Comparator<Eleitor> comparator) {  
    return listarTodosEleitores().stream().sorted(comparator).collect(Collectors.toList());  
}
```




Ordenar Eleitores

- Imprime ordenado pelo nome
- Imprime ordenado pela Zona Eleitoral

```
private void ordenarEleitores() {  
    System.out.println("==== Ordenar Eleitores =====");  
    System.out.println("1. Por Nome");  
    System.out.println("2. Por Zona Eleitoral");  
    System.out.print("Escolha uma opção: ");  
    int opcao = scanner.nextInt();  
    scanner.nextLine(); // Consumir nova linha  
  
    List<Eleitor> eleitoresOrdenados;  
    switch (opcao) {  
        case 1:  
            eleitoresOrdenados = eleitorService.ordenarEleitores(  
                (e1, e2) -> e1.getPessoa().getNome().compareToIgnoreCase(e2.getPessoa().getNome()));  
            mostrarEleitores(eleitoresOrdenados);  
            break;  
        case 2:  
            eleitoresOrdenados = eleitorService.ordenarEleitores(  
                (e1, e2) -> Integer.compare(e1.getZonaEleitoral(), e2.getZonaEleitoral()));  
            mostrarEleitores(eleitoresOrdenados);  
            break;  
    }  
}
```

Pacote View
Classe: MainView

Two arrows originate from the code. One arrow starts at the line 'System.out.println("1. Por Nome");' and points to the 'Pacote View' box. The other arrow starts at the line 'System.out.println("2. Por Zona Eleitoral");' and points to the 'Classe: MainView' box.

Iniciar Eleição

- Setta o ano na lista de anos que não votou
- Setta multa somando o valor anual da multa
- Setta situação para false

Iniciar Eleição

Pacote View
Classe: MainView
iniciarEleicao()

```
for (Eleitor eleitor : eleitores) {  
    eleitor.setSituacao(false);  
    eleitor.adicionarAnosNaoVotados(anoAtual);  
    eleitor.setMultas(eleitor.getMultas() + multa);
```

```
System.out.println("Digite o número do Título do próximo eleitor a votar:");  
String titulo_votar = scanner.nextLine();  
StringBuilder sb = new StringBuilder(titulo_votar);  
sb.insert(4, " ");  
sb.insert(9, " ");
```

```
if (eleitor.getAnosSemVotar().size() == 1){  
    eleitor.setSituacao(true);  
    eleitor.limparAnosSemVotar();  
    System.out.println(eleitor.getPessoa().getNome() + " votou com sucesso.");  
    System.out.println("Digite (1) para o próximo eleitor ou (2) para encerrar a eleição.");  
} else {  
    System.out.println("O eleitor " + eleitor.getPessoa().getNome() + " não pode votar pois está INAT");  
    System.out.println("Digite (1) para o próximo eleitor ou (2) para encerra
```