UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

DEPARTAMENTO DE INFORMÁTICA

PROGRAMAÇÃO ORIENTADA A OBJETO

Afonso Salvador de Magalhães

Thamya Vieira Hashimoto Donadia

Análise dos resultados das eleições brasileiras

Trabalho I - Java

Vitória, ES

2022

Introdução

O projeto consiste no desenvolvimento de relatórios da base de dados das eleições brasileiras de 2022. A partir das informações disponibilizadas pelo Tribunal Superior Eleitoral, foi possível obter análises acerca dos votos para deputados federais e estaduais, bem como dos candidatos a tais cargos. O programa manipula 4 pacotes principais: o de eleição, que organiza e mantém os dados acerca dos candidatos e dos partidos presentes na seção a ser avaliada; o de registrados, que contém as classes do candidato e do partido; o de serviços de entrada e saída, isto é, leitura dos dados e impressão na saída padrão; e, por fim, o de relatórios.

A partir do arquivo de candidatos e de votos, cujos nomes são passados como parâmetro na linha de comando, o código estrutura a relação partido-candidatos e coleta os dados a serem utilizados no desenvolvimento dos relatórios. As funcionalidades principais do projeto foram definidas na classe “Relatorio.java”, que compreende a avaliação das informações fornecidas.

Implementação

A priori, para a estruturação do sistema foram definidas 7 classes, as quais são especificadas a seguir:

A. *App.java*

A classe “App” contém a função main, a qual, a priori, organiza as informações passadas na linha de comando da execução do programa, isto é, os arquivos a serem utilizados, a data da análise e o modelo de consulta (federal ou estadual). A partir disso, ocorre a chamada das funções principais do programa: leitura dos arquivos e desenvolvimento dos relatórios.

B. *Candidato.java e Partido.java*

As duas classes, “Candidato” e “Partido” detém informações individuais sobre as duas instâncias e apresentam funções de Get e Set. Vale ressaltar que são duas classes comparáveis, isto é, um candidato é comparável a outro e o mesmo vale para um partido.

C. *Eleicao.java*

A classe “Eleicao” contém os dados de todos os candidatos válidos e partidos da seção analisada. Vale ressaltar que os candidatos foram organizados em uma estrutura de HashMap, com o intuito de otimizar o acesso a cada instância. Dessa forma, esta classe apresenta funções do tipo Get e Set, bem como de inserção e retirada nas estruturas de lista e map.

D. *Relatorio.java*

Na classe “Relatorio” são realizadas todas as análises e desenvolvimentos dos relatórios a serem impressos, a partir dos dados presentes na classe “Eleicao”. Desse modo, a partir da organização das informações e disposição da forma solicitada, os métodos correspondentes a cada relatório realizam a chamada das respectivas funções de impressão.

E. *Leitor.java e Impressora.java*

As classes “Leitor” e “Impressora” realizam os serviços de entrada e saída do programa. Sendo assim, por meio dos arquivos fornecidos na linha de comando, ocorre a leitura dos dados e armazenamento na classe “Eleicao”. Vale ressaltar que para isso, foi utilizado o BufferedReader, o qual permite a leitura linha a linha. Isso apresenta-se como uma vantagem, uma vez que o arquivo CSV apresenta conjunto de informações organizadas por linha. Ademais, os métodos de impressão, correspondentes a cada relatório, foram declarados para a saída padrão, de modo que são utilizados pela classe "Relatório" ao final da organização das informações.

No que se trata das exceções, foram tratadas àquelas que são referentes à abertura e busca de arquivos, bem como de entrada e saída. Desse modo, o tratamento das mesmas foi realizado por meio do uso da ferramenta “try, catch”. Em conclusão, a arquitetura, em específico, a relação das classes ficou definida da seguinte forma:

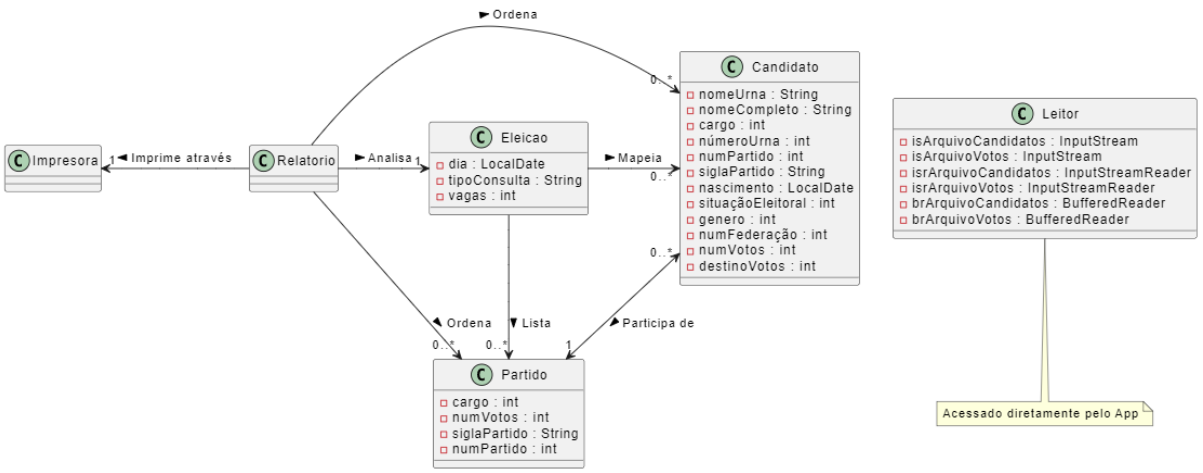

Figura 1 - Diagrama UML da relação de classes do programa.

O sistema foi projetado para que as funções principais, isto é, o desenvolvimento dos relatórios a partir dos dados coletados pelo leitor, fossem definidas dentro da classe “Relatorio”. Desse modo, foi possível manter as estruturas de dados protegidas, uma vez

que no “App.java”, o qual contém a *main*, apenas são chamadas as funções que gerenciam os relatórios, bem como a de leitura dos arquivos, como apresentado na Figura 2.

```java
public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {

        // verificação de argumentos
        if (args.length != 4) {
            System.out.println("uso: java -jar deputados.jar --<modalidade> <arquivo_candidatos> <arquivo_votação> <dia da votacao>");
            System.exit(1);
        }

        // obtenção dos dados passados como parâmetros na entrada padrão
        String modalidade = args[0];
        String caminhoArquivoCandidatos = args[1];
        String caminhoArquivoVotos = args[2];

        // formatação da data passada como parâmetro na entrada padrão
        DateTimeFormatter formatador = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate dia = LocalDate.parse(args[3], formatador);
        dia.format(formatador);

        // inicia a eleição e leitor
        Eleicao processo = new Eleicao(dia, modalidade);
        Leitor reader = new Leitor(caminhoArquivoCandidatos, caminhoArquivoVotos);
        reader.leituraCandidatos(processo);
        reader.leituraVotos(processo);

        // inicia o sintetizador de informação e gera os relatórios
        Relatorio sintetizador = new Relatorio(processo);
        sintetizador.numeroVagas();
        sintetizador.deputadosEleitos();
        sintetizador.deputadosMaisVotados();
        sintetizador.partidosEleitos();
        sintetizador.primeiroUltimoPartidos();
        sintetizador.eleitosPorIdade();
        sintetizador.eleitosPorSexo();
        sintetizador.distribuicaoVotos();

    }
```

Figura 2 - Arquivo App.java

Dessa forma, o sistema tem a seguinte ordem de funcionamento:

1. Leitura e tratamento dos parâmetros fornecidos na linha de comando: modalidade da análise, os caminhos para os arquivos de candidatos e votos e a data da eleição (formato dd/MM/yyyy);
2. Leitura dos arquivos de candidatos e de votos e criação do mapa de candidatos e da lista de partidos presentes na eleição analisada, bem como da rede de conexões entre as duas instâncias. Esses processos ocorrem na classe “Leitor”;
3. Desenvolvimentos dos relatórios, realizado pelo “Relatorio”, o qual também chama os métodos de impressão, a partir dos dados coletados pelo “Leitor”.

Vale ressaltar que, no tocante à leitura dos arquivos, o código lê uma linha inteira do CSV e a fragmenta na caractere que identifica uma coluna. As colunas são, então, armazenadas em um vetor. A partir disso, as informações dos votos e dos candidatos são

acessadas pelas posições do vetor para armazenar as informações, bem como construir os candidatos e partidos.

Descrição dos testes

A priori, ao longo da construção do programa e do desenvolvimento dos relatórios, o código foi testado apenas com os dados do Espírito Santo, gerando o Jar e utilizando o comando java -jar brazilian-elections-data-analysis.jar --estadual tests/consulta_cand_2022_ES.csv tests/votacao_secao_2022_ES.csv 15/11/2022 > output.txt, trocando a flag dependendo da modalidade de análise ("--estadual" ou "--federal"). O arquivo "output.txt" foi comparado com os relatórios apresentados no documento de apresentação do trabalho.

Durante a realização dos testes, foi observado que o tempo de processamento estava elevado e, após a análise do código, verificou-se que os candidatos dentro da classe "Eleição" estavam estruturados em forma de lista, logo após a leitura dos arquivos. No entanto, em função da alta frequência de acesso a tais informações, foi estabelecida a mudança para uma mais eficiente em termos de busca e de acesso, consolidando-se em um HashMap de candidatos.

Com o código finalizado, foi utilizado o script de correção disponibilizado pelo professor, no qual o programa é testado para os seguintes estados: AC, AL, ES, MG, PE, RS nos quais os arquivos testados estão especificados na Figura 3.

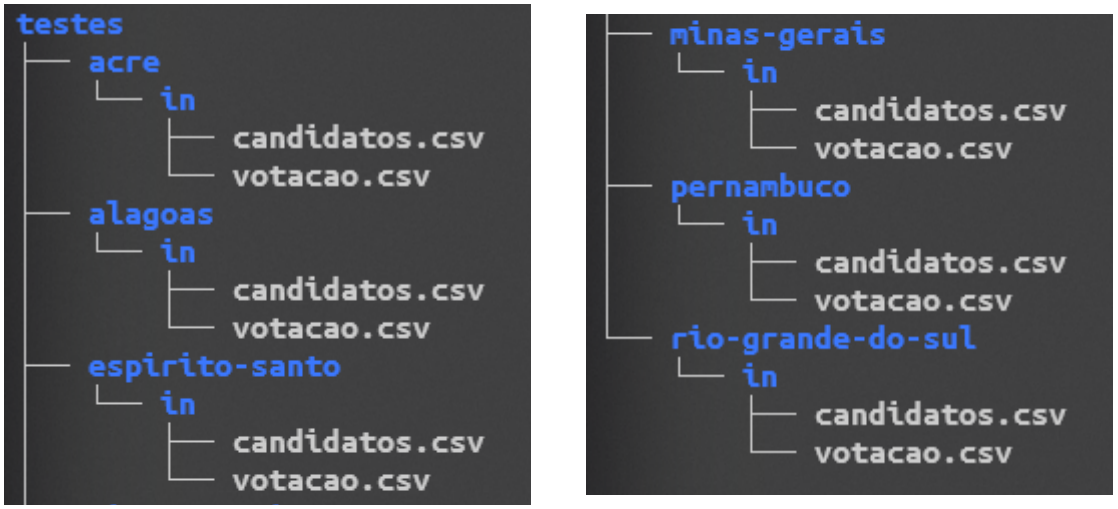

Figura 3 - Arquivos utilizados pelo script para os testes

Desse modo, o script informou que os relatórios desenvolvidos estão de acordo com o esperado, como apresentado na Figura 4.

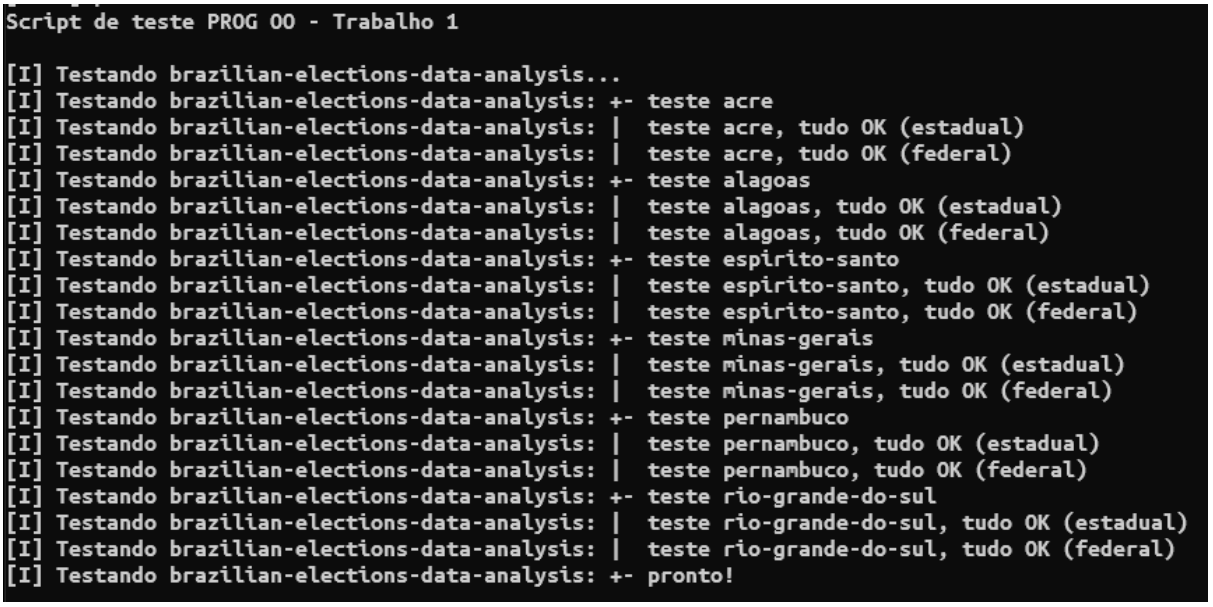

Figura 4 - Resultados dos testes realizados pelo script

Bugs

Ao realizar uma análise do código, observou-se que caso o usuário digite um valor inválido no campo de modalidade, o programa funcionará em fluxo regular, considerando a modalidade como estadual. Isso ocorre em virtude da escolha de diferenciação das flags no código, isto é, por comparação de Strings.

Conclusão

A elaboração do sistema foi pautada em uma organização prévia, em que foi definida a arquitetura geral dos pacotes e classes, isto é, quais seriam implementadas e como estariam correlacionadas. Desse modo, a partir dessa documentação inicial, o processo de escrever o código foi guiado, e, portanto, fluiu mais rapidamente.

Bibliografia

JAVA™ Platform, Standard Edition 8 API Specification: Packages. [*S. l.*]. Disponível em: https://docs.oracle.com/javase/8/docs/api/overview-summary.html. Acesso em: 15 nov. 2022.