Álvaro D. S. Alves
Elder Storck

Implementação do Sistema Eleitoral Brasileiro na Programação

Goiabeiras - Vitória - ES - Brasil
2022, Fevereiro

Álvaro D. S. Alves
Elder Storck

Implementação do Sistema Eleitoral Brasileiro na Programação

Implementação do Sistema Eleitoral Brasileiro na Linguaguem Java (Linguaguem de Programação Orientada a Objetos)

Universidade Federal do Espírito Santo
Curso de Engenharia de Computação
Disciplina de Programação III

Orientador: João Paulo A. Almeida

Goiabeiras - Vitória - ES - Brasil
2022, Fevereiro

SUMÁRIO

1 INTRODUÇÃO

Este trabalho tem como objetivo pôr em prática os conceitos de *Programação Orientada a Objetos* (POO) utilizando a linguagem de programação **Java**.

A tarefa em questão é utilizar as caracterísicas da POO (encapsulamento, herança, polimorfismo e instanciação) a fim de representar, de maneira abstrata porém clara, o sistema eleitoral brasileiro.

O trabalho também se utiliza de conceitos dos paradigmas de programação anteriores, como os paradigmas da programação procedural e estruturada, ao implementar funções em blocos, estruturas de repetição, estruturas condicionais, tratamento de erros e exceções além de estruturas de dados.

2 IMPLEMENTAÇÃO

Para a implementação da solução, o primeiro passo realizado foi o de dividir o problema em partes, por exemplo:

- Ler o arquivo csv e armazenar os dados em seus campos no seu formato correto;
- Implementar uma estrutura de dados que atendesse às necessidades para gerenciar os dados em questão;
- Criar um esquema de classes claro e objetivo;
- Implementar funcionalidades para gerenciar os dados;
- Implementar funcionalidades para exibir os relatórios;

Com isso surgiram classes como:

- A classe principal `Main`, que apenas recebe os parâmetros e instancia as outras classes;
- A classe `Election`, que realiza todas as ações concernentes à eleição de modo geral;
- A classe `Utils`, que fornece funções simples de leitura e comparação;
- A classe `Reports`, que exibe os relatórios;
- E, por fim, as classes `Candidate` e `Party`, que servem para a instanciação das entidades dos candidatos e partidos.

Além de outras classes utilizadas para comparação e ordenação.

Todas as classes que possuem *atributos* utilizam o recurso de encapsulamento, visando maior restrição e controle sobre tais atributos e possuindo, além de *métodos construtores*, métodos de manipulação dos atributos, conhecidos como *getters* e *setters*.

Uma vez observado que as classes possuírem diferenças em demasia e poucas semelhanças, não foi vista como necessária a utilização do recurso de herança de classes. Porém a implementação de *interfaces* e o polimorfismo foram utilizados nas classes comparadoras.

2.1 EXEMPLO DE CLASSE EM JAVA

```java
// exports and imports
package src.classes;
import java.util.LinkedList;


public class Party {
        // attributes
        private int number = 0;
        private String name = "";
        private String alias = "";
        private int partyVotes = 0;
        private int totalVotes = 0;
        private LinkedList<Candidate> candidates = null;


        // constructor method
        public Party(String alias) {
                this.alias = alias;
        }


        // methods
        public int getNumber() {
                return number;
        }
        public void setNumber(int number) {
                this.number = number;
        }

        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }

        public int getPartyVotes() {
                return partyVotes;
        }
        public void setPartyVotes(int votes) {
                this.partyVotes = votes;
        }
}
```

3 RELACIONAMENTO ENTRE CLASSES

O relacionamento entre as classes e interfaces mencionadas acima é representado no diagrama a seguir, descrito em UML, a *Unified Modeling Language*.

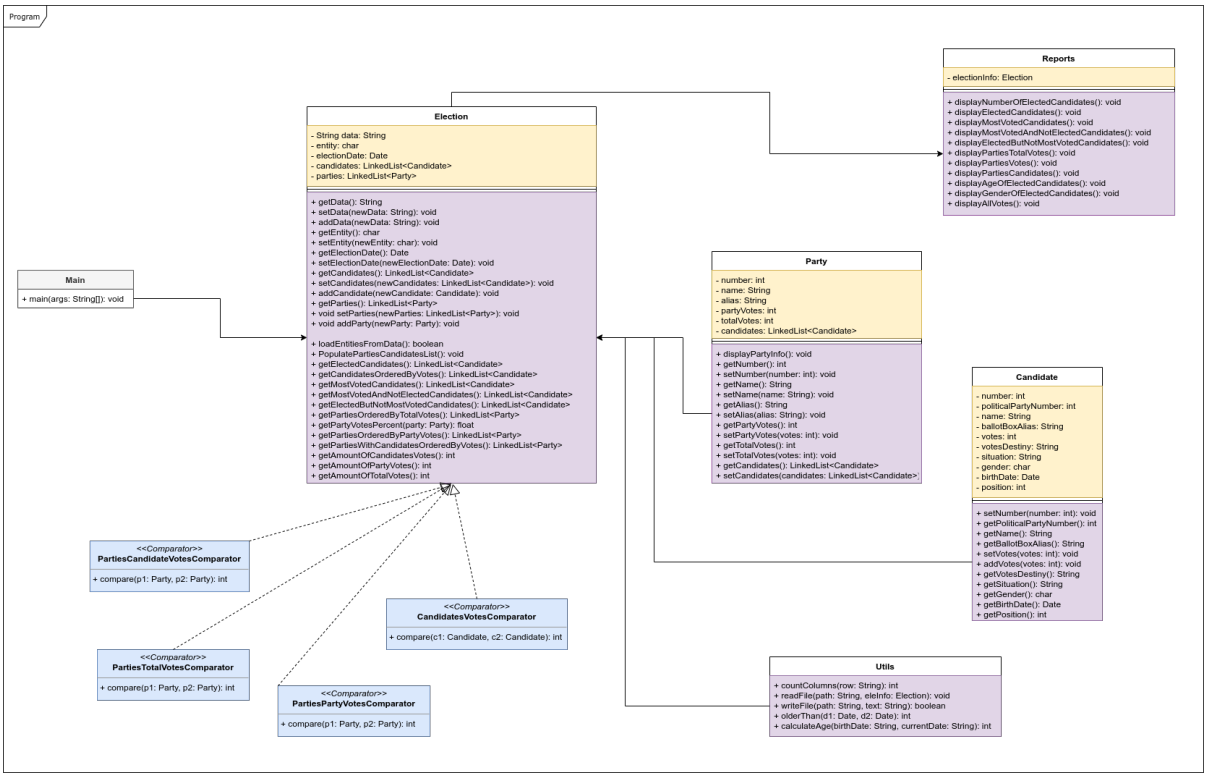

Figura 1.0 - Diagrama de Classes em UML

Onde as setas tracejadas represantam a implementação das interfaces (em azul) e as linhas contínuas representam tanto a instanciação de uma classe dentro de outra classe quanto o fluxo do programa.

Cada classe possui atributos (em amarelo) e/ou métodos (em roxo). É notório o que a classe **Election** possui uma maior quantidade de métodos do que as outras classes, isso se deve pelo fato desta classe ser referente a todo o contexto do programa para a eleição, que envolve desde a criação das entidades até a manipulação das estruturas que as contém.

Novamente é visível que a classe **Main** apenas instancia a classe de eleição e a utiliza para obter os dados que serão exibidos na classe **Reports**.

4 TRATAMENTO DE ERROS E EXCEÇÕES

A fim de evitar erros, foram feitos tratamentos de excessões. A exemplo, para a funções de leitura e escrita (se necessário) de arquivos foi feito o tratamento de input/output. Para formatação de datas foi utilizada uma biblioteca que retornava uma exceção de conversão, tal exceção também foi tratada.

Além disso, também foram feitos tratamentos de dados retornados por funções, a fim de evitar erros de tipagem, parâmetros, argumentos ou métodos inválidos.

5 KNOWN BUGS

Como já dito anteriormente, o tratamento de possíveis erros também foi realizado, porém isso não impede que novos erros ocorram em caso de divergências, mesmo que sutis.

Um exemplo que vale ser mencionado se encontra na linha 119 do arquivo `src/Election.java`, onde os dados das linhas dos arquivos CSV são utilizado apenas caso sigam o formato padrão de separação por vírgula e caso cada linha possua a mesma quantidade de colunas/dados entre vírgulas que a primeira linha. Tal situação também possui um tratamento de exceções, que leva em conta a quantidade de colunas ser zero, a linha ser uma string vazia ou a linha ser nula. Qualquer caso fora do formato padrão ou destas 3 exceções resultaria em um erro visível no standart output, prejudicando o fluxo do programa.

Outro exemplo relacionado ao anterior se encontra na função readFile do arquivo `src/Utils.java`, que apenas leva em conta o formato padrão dos arquivos .csv para as duas entidades, candidato e partido. Qualquer tipo de dado ou formato além desses na base de dados resultaria em um bug.

6 TESTES

Fora os bugs mencionados anteriormente, nenhum outro problema foi detectado durante os testes. Testes esses realizados utilizando bases de dados de exemplo, como a base de dados do município da Serra, que pode ser encontrada junto ao repositório do código no GitHub.

Além disso, o trabalho passou em todos os cases de teste fornecidos pelo professor, o que auxilia a mensurar o estágio do desenvolvimento.

7 CONCLUSÃO

Por fim, com a realização desse trabalho foi possível compreender mais acerca da aplicabilidade do paradigma de Orientação a Objetos, pôr em prática os conhecimentos na linguagem Java, entender melhor o funcionamento do sistema eleitoral brasileiro além de adquirir contato com o processo de desenvolvimento de software de uma forma organizada, eficiente, documentada e clara, apesar de abstrata.