UNIVERSIDADE FEDERAL DO ESPIRITO SANTO

Primeiro Trabalho de Programação III

Relatório

Lucas Moraes Soares de Souza
Lucas Mendonça Emery Cade

Vitória
Outubro de 2018

UNIVERSIDADE FEDERAL DO ESPIRITO SANTO

Lucas Moraes Soares de Souza
Lucas Mendonça Emery Cade

Primeiro Trabalho de Programação III

Relatório

Trabalho referente à disciplina de Programação III do curso de Ciência de Computação do Departamento de Informática da Universidade Federal do Espírito Santo.

Vitória
Outubro de 2018

Sumário

1 Introdução

O objetivo deste trabalho é a prática de conceitos básicos da linguagem java, tais como leitura de arquivo e organização de dados.

Dado o resultado de uma eleição municipal, o programa deve investigar e gerar listas ordenadas segundo os critérios abaixo:

- Número de vagas;
- Candidatos eleitos (sempre indicado partido, número de votos e coligação, se houver);
- Candidatos mais votados dentro do número de vagas;
- Candidatos não eleitos e que seriam eleitos se a votação fosse majoritária;
- Candidatos eleitos no sistema proporcional vigente, e que não seriam eleitos se a votação fosse majoritária;
- Votos totalizados por coligação;
- Votos totalizados por partido;
- Total de votos nominais.

As listas estarão ordenadas segundo o número de votos contabilizados pelo item em questão e cada um destes apresentará informações relevantes para a identificação de cada partido, coligação ou candidato.

A explicação do código será desenvolvido em capítulos. Cada classe será dividida em subseções específicas da seção de Desenvolvimento [2] e, em cada uma, será explicada os métodos e atributos referente a classe em questão.

2 Desenvolvimento

2.1 Candidato

2.1.1 Atributos

Atributos da classe:

- situação;
- colocação;
- número;
- nome;
- partido;
- coligação;
- votos.

Os atributos possuem suas funções getters e setters.

2.1.2 Função setColigacao

Recebe uma String que precisa ser tratada para se dividir os nomes dos partidos nela contida.

2.1.3 Função setColocacao

A String que contem a colocação do candidato contém outros dados relevantes que precisam ser separados. Caso o tamanho da String seja 5 contém um char de status na primeira posição, logo se extrai esse primeiro carácter ou apenas se registra a colocação do candidato e se põe 'n' na situação do candidato.

2.1.4 Função toString

O toString da função foi alterado para retornar o formato pedido de exibição de candidatos

O caso referente a um partido sem coligação será abordado no capítulo referente à classe Coligacao [2.4].

Figura 1: Caption

2.2 Diagrama

2.3 Partido

2.3.1 Atributos

Atributos da classe:

- nome;
- votos somatório do número de votos dos candidatos pertencentes ao partido;
- quantidade de candidatos eleitos do partido;

Cada atributo possui suas respectivas funções getter e setter.

2.3.2 Função toString

O toString da função foi alterado para retornar o formato pedido de exibição de partido.

2.3.3 Função imprimeNomePartido

Retorna apenas o nome, escolhemos criar tal função para facilitar a elaboração de futuras funções que consultam partido.

Para mais informações verifique a função compare da classe Candidato [2.1].

2.4 Coligação

2.4.1 Atributos

A classe Coligação possui os seguintes atributos:

- ArrayList dos nomes dos partidos que formam a coligação;
- votos somatório do número de votos dos candidatos pertencentes à coligação;
- quantidade de cadidatos eleitos que pertencem a coligação.

Cada atributo possui suas respectivas funções getter e setter.

2.4.2 Função coligacao

Recebe os partidos que formam a coligação juntos em uma String e os separa, assim se sabe individualmente, quais partidos fazem parte da coligação.

2.4.3 Função toString

O toString da função foi alterado para retornar o formato pedido de exibição de coligação.

2.4.4 Função imprimeColigacao

Retorna uma String com a formatação exigida para no enunciado para a saida do programa.

2.4.5 Função comparaColigacao

Checa se uma coligação dada como parâmetro é a mesma da atual coligação.
Para mais informações verifique a função compare no capitulo sobre a classe Candidato [2.1].

2.5 Lista candidatos

2.5.1 Atributos

A classe Lista candidatos possui os seguintes atributos:

- lista de todos os candidatos;
- lista de todos os eleitos;
- lista dos mais votados;
- lista dos candidatos que seriam eleitos caso a eleição fosse majoritária;
- lista dos candidatos que foram eleitos graças ao sistema proporcional.

Cada uma com seus respectivos getters, setters e toStrings especializados para o correto retorno de dados. Tal classe foi criada para tratar unicamente às listas que se basearam no desempenho de cada candidato, se diferenciando assim nas classes mostradas nos tópicos a frente que se concentram em gerar as listas referente ao desempenho dos partidos e coligações.

2.5.2 Função preencheListas

- Irá organizar em ordem decrescente os candidatos segundo seus votos;
- Descobrir os candidatos que não seriam eleitos caso não houvesse os votos do sistema proporcional;
- Verificar os candidatos com o maior número de votos que não foram eleitos e inserí-los na lista de eleitos majoritária, até que ela tenha o mesmo tamanho que a lsita beneficiados.

2.6 Lista partidos

2.6.1 Atributos

A classe Lista de partidos possui os seguintes atributos:

- lista de todos os partidos.

Tal classe foi criada para padronizar a organização das listas referentes aos partidos. Seu único atributo possui as funções getters e setters padrão. O diferencial se encontra na função toString da classe, que a organiza segundo os partidos mais votados e gera uma String de saída especializada para a classe.

2.7 Lista coligacao

2.7.1 Atributos

A classe Lista coligacao possui os seguintes atributos:

- lista de todos os partidos.

Tal classe foi criada para padronizar a organização das listas referentes às coligações. Seu único atributo possui as funções getters e setters padrão. O diferencial se encontra na função toString da classe, que a organiza segundo os partidos mais votados e gera uma String de saída especializada para a classe. Um partido que não possui uma coligação também será listado aqui.

2.8 Lista coligacao

2.8.1 Atributos

A classe Lista coligacao possui os seguintes atributos:

- lista de todos os partidos.

Tal classe foi criada para padronizar a organização das listas referentes às coligações. Seu único atributo possui seus métodos getters e setters padrão. O diferencial se encontra na função toString da classe, que a organiza segundo os partidos mais votados e gera uma String de saída especializada para a classe. Um partido que não possui uma coligação também será listado aqui.

2.9 Dados

2.9.1 Atributos

A classe Dados possui os seguintes atributos:

- uma instância da classe Lista candidato;
- uma instância da classe Lista partido;
- uma instância da classe Lista coligacao.

A abordagem de aglomerar as listas em classes específicas tem seu ápice nessa classe que acumula todas as demais classes que geram as listas exigidas para solucionar o problema. Cada um de seus atributos possui seus métodos getters e setters.

2.10 IO

2.10.1 Atributos

A classe IO não possui atributos, pois seu objetivo nessa implementação é aglomerar a leitura do arquivo de entrada e realizar a saída.

2.10.2 Função le arquivo

Ao realizar a leitura do arquivo de entrada em um método a parte é possível alertar exceções mais claramente.

2.10.3 Função imprimeSaida

Irá imprimir proceduralmente a saída referente a cada lista gerada durante o processo do programa, para evitar a formação de uma String gigantesca. Além de acrescentar as linhas de explicação sobre cada uma das listas como especificado no enunciado.

2.11 Testador

2.11.1 Atributos

A classe Testador não possui atributos. Seu objetivo nessa implementação é iniciar o programa e chamar os métodos na sequência necessária para a execução da aplicação.

3 Conclusão

Ao longo do desenvolvimento desse trabalho nos deparamos com algumas dificuldades. Primeiramente com relembrar conceitos básicos de Java. Pela falta de familiaridade com a linguagem foi necessário consultar as apostilas de aula, além de diversos sites e fóruns para relembrar como implementar determinadas estruturas e métodos.

Alguns problemas se deram também por conta de infraestrutura. Os computadores do LabGrad muitas vezes travavam durante a implementação gerando estresse e, com ele, perda de desempenho. Em alguns casos extremos, foi necessário reiniciar o computador perdendo assim códigos recém produzidos e não salvos.

A ideia a ser implementada possuía o nível de dificuldade ideal para programadores novos na linguagem, mas com experiencia em outras linguagens. Ela apresentava ideias de ordenação e leitura de arquivos, conceitos chave para o curso, e ao mesmo tempo, permitia a utilização de uma série de estruturas já existentes na linguagem que somente precisaram ser reutilizadas de forma adequada, exercitando assim a nova linguagem. Conclui-se que o objetivo de gerar uma familiarização com o Java e a IDE Eclipse foi cumprida, ao mesmo tempo em que noções básicas de programação foram revisadas.