PRIMEIRO TRABALHO DE IMPLEMENTAÇÃO (JAVA)

Alunos: Luca da Silva Ávila e Luiz Phillyp Sabadini Bazoni

Introdução

O presente relatório tem como objetivo servir de sumário e documentação para a implementação do programa que o acompanha. Escrito na linguagem Java, tal programa tem como finalidade o processamento de dados eleitorais fornecidos em arquivos csv pelo site oficial do TSE. O relatório é composto em sua totalidade por três partes, a primeira descreve o funcionamento geral do programa além de documentar detalhadamente as principais classes e estruturas que são usadas para o funcionamento do algoritmo principal, sem seguida é descrito o processo utilizado para o teste do programa usando diversos arquivos como entrada e também o sumário dos resultados obtidos após os testes. E por fim será documentado os principais casos que podem por ventura quebrar o código e a consequência desses erros no funcionamento imediato do programa assim como no longo prazo.

Compilação e execução do programa

Antes de relatar a estrutura geral do programa, cabe descrever o processo utilizado para a compilação e execução do programa e também o gerenciamento e padrões assumidos para entradas e saídas do programa. Também será descrito como é possível alterar o código para a leitura de diferentes formatos de entrada e o redirecionamento da saída do programa.
Para compilar o programa, usa-se o arquivo build.xml, que contém as informações relevantes sobre os nomes das pastas para que se possa fazer a compilação (é necessário rodar o comando “ant compile” para isso) e a formação do executável (usa-se o comando “ant jar”) … após esse processo o executável do programa fica compactado em um arquivo jar.

Para que o código rode, faz-se necessário que a linha de comando esteja escrita na forma “java -jar deputados.jar <tipoCargo> <arquivoCandidatos> <arquivoVotacoes> <dataEleicao>”, sendo que os campos entre as tags deve ser substituído pelo nome do que se deseja usar (especificamente tipoCargo precisa ser ou –estadual ou –federal e dataEleicao precisa ser no formato dd/MM/yyyy). Todas as entradas do código são extraídas dos arquivos de inscrições dos candidatos e de votação das eleições. As Exceptions possíveis no código

quanto às entradas e saídas são caso falte algum dos parâmetros para o executável, se não os coloque no padrão necessário ou se não encontrar os arquivos de entradas.

Foram implementadas no total três pacotes para gerenciar diversas partes do processamento dos dados: o pacote Candidatura, que tem como objetivo encapsular todas as classes que armazenam os dados referentes a candidatos, partidos e interações entre eles; o pacote Estaticos, mantém classes estáticas que implementam as configurações que serão usadas para leitura e escrita de arquivos; e por fim o pacote Relatórios gerencia a leitura e escrita dos arquivos que são usados ou gerados pelo programa.

–Descrição das classes usadas em cada Pacote

Candidaturas:

- Candidato: Classe responsável pelo armazenamento dos dados referentes aos candidatos que se inscreveram para a eleição. É o elemento mais básico do código implementado, já que todo o restante se baseia nos dados que estão sendo inseridos nele. O construtor inicialmente insere o candidato em uma estrutura de dados no tipo Partido, que será descrito posteriormente. Com exceção da data de nascimento do candidato, que teve o tipo LocalDate escolhido para sua representação, a maioria dos parâmetros do construtor precisam ser trabalhados na main quando o objeto é instanciado. A data de nascimento é fornecida como String e trabalhada dentro do construtor. Por conta disso, inserir uma String no parâmetro de data de nascimento que não esteja no formato “dd/MM/yyyy” gera uma DateTimeParseException. Dentro do bloco do construtor também estão presentes os enums de Situação, importantes para legibilidade do código e que serão explicados posteriormente. Além dos getters para os dados privados do objeto, a classe Candidato ainda apresenta uma função que adiciona um número dado de votos para o total de votos do candidato, uma função que calcula sua idade para determinada data e uma sobrescrita do método ToString.
- Partido: Uma das classes que reúne os candidatos instanciados ao longo do projeto, é composta por dados básicos que a descrevem (sigla e número), quantidades de votos nominais e de legenda e uma LinkedHashMap para ordenar os candidatos. A LinkedHashMap tem como característica misturar características específicas de lista e de HashMap. As funções presentes na classe são os getters, sendo um deles uma função que retorna uma cópia da LinkedHashMap ordenada (usando uma função

lambda para ordenar essa lista copiada), uma função para inserir um candidato na LinkedHashMap e outra para adicionar votos nominais e de legenda para os cálculos do partido.

- Enums de Situação: se tratam de três enums responsáveis por armazenar os valores padronizados pré-estabelecidos no projeto para gerar comparações com os valores armazenados nos dados do candidato, de modo que consiga haver uma legibilidade maior que decorar vários números diferentes e também consiga garantir uma mutabilidade facilitada no caso de o TSE mude os valores especificados para os códigos da eleição (desde que a forma que estão dispostos na tabela não mudem).

Estaticos:

- CodigoCargo: enum responsável para servir de parâmetro para decidir se os dados que aparecem no arquivo csv são correspondentes ao que se deseja. É onde se inseriu os números 6 e 7 como padrão para deputados federais e estaduais respectivamente.
- Genero: enum responsável para servir de parâmetro para comparar com o gênero dos candidatos do csv, colocando os números 2 e 4 como padrão para masculino e feminino respectivamente.

Relatorios:

- CsvReader: essa classe é responsável pela leitura das linhas do arquivo csv. Seus elementos internos são um tipo File para servir de fonte para as informações, uma HashMap para conseguir um cabeçalho de índices (nomeado “headerIndexes”) para leitura das linhas de arquivos, um tipo Scanner para manipular as Strings formadas de acordo com o cabeçalho formado pela HashMap, uma String para servir de delimitador para separar as Strings de acordo com o cabeçalho e ainda outra para servir para ler a linha atual do arquivo que se lê. A classe CsvReader usa de parâmetros para seu construtor o nome do caminho para um arquivo, um delimitador, e uma String que dê o nome do charSet de leitura a ser usado. Podem ser usadas duas versões para o CsvReader: uma que assume o uso de headers como automático e outra que não assume (isso depende do arquivo que está sendo lido). Os dois construtores funcionam de maneira semelhante: eles determinam qual é o delimitador para separar as Strings sendo lidas; instanciam dois objetos (um para o arquivo de fonte de dados e outro para a HashMap de índices de cabeçalho); separa os elementos da primeira linha; lê cada um deles e gera um índice para eles enquanto ainda houver elemento pra ser lido na

mesma linha; por fim, determina-se a linha atual que o CsvReader pode ler, sendo esse valor nulo se não houver mais linhas no arquivo, ou a segunda linha caso haja. As funções depois do construtor são referentes à leitura de próximas linhas (atualizando as próximas linhas a serem lidas até a última linha do arquivo, onde o CsvReader termina suas atividades), que retornam os valores desejados de cada linha do arquivo csv.

- Relatorio: a classe que nomeia o pacote. A função dessa classe é extrair os dados processados em SistemaEleitoral e imprimir as saídas desejadas (os 10 relatórios exigidos nas especificações do trabalho). A impressão das saídas ocorre no próprio terminal e o construtor dessa classe determina algumas coisas importantes, como: o objeto do tipo Locale é especificado para “pt-BR”, a formatação dos números inteiros e dos números racionais segue o mesmo padrão (apesar de que os números racionais têm exatamente duas casas decimais). Os parâmetros do construtor são referentes ao tipo SistemaEleitoral, que é de onde o relatório irá extrair os dados, o modo referente à configuração escolhida para avaliar o cargo (deputado federal ou estadual) e a data de eleição, já determinada como LocalDate. Nesse tipo ainda existem duas listas, uma para candidatos e outra para partidos. As duas recebem cópias ordenadas das LinkedHashMap de candidatos e partidos processados ao longo do código. Essa parte do código não afeta em nada o processamento de dados e não altera os valores gerais do código, tendo como único intuito a impressão dos relatórios exigidos.
- SistemaEleitoral: a esse enum cabe o processamento dos dados recebidos pela leitura do csv, sendo eles: quais candidatos e partidos estão presentes na eleição de um estado e como eles são votados. Por só ser necessário um tipo de SistemaEleitoral no código e ele não precisar de um construtor, foi usado o padrão de projeto Singleton, já que é um objeto global. Esse enum apresenta duas LinkedHashMap, sendo uma para candidatos e outra para partidos. Como é necessário que todo candidato pertença a um partido, a inscrição de um candidato (ou seja, sua inserção na LinkedHashMap de candidatos participantes) precisa já haver um partido inscrito para seu funcionamento (inscrever um candidato sem o partido inscrito anteriormente lança uma RunTimeException). Os funcionamentos para inscrição de candidatos e de partido colocam nas estruturas de dados respectivas caso eles já não estejam ali, evitando que um mesmo candidato e um mesmo partido repitam os processos de inserção. Além disso, ainda há as funções referentes ao incremento de votações, sendo responsáveis por aumentar o número de votos dos candidatos, dos partidos e do sistema como um todo, podendo ser de legenda ou nominais. Funções essenciais para a impressão dos

dados são as duas que geram cópias dos dados dispostos nas LinkedHashMap de candidatos e partidos, porque não só elas fornecem os dados todos dispostos na estrutura de dados, elas ordenam a cópia, usando para isso uma função lambda para gerar comparações. Também é importante citar que existem getters ao longo desse enum, com capacidade de buscar elementos que pertencem às estruturas de dados privadas de SistemaEleitoral.

–Descrição do algoritmo usado na main para o uso das classes implementadas

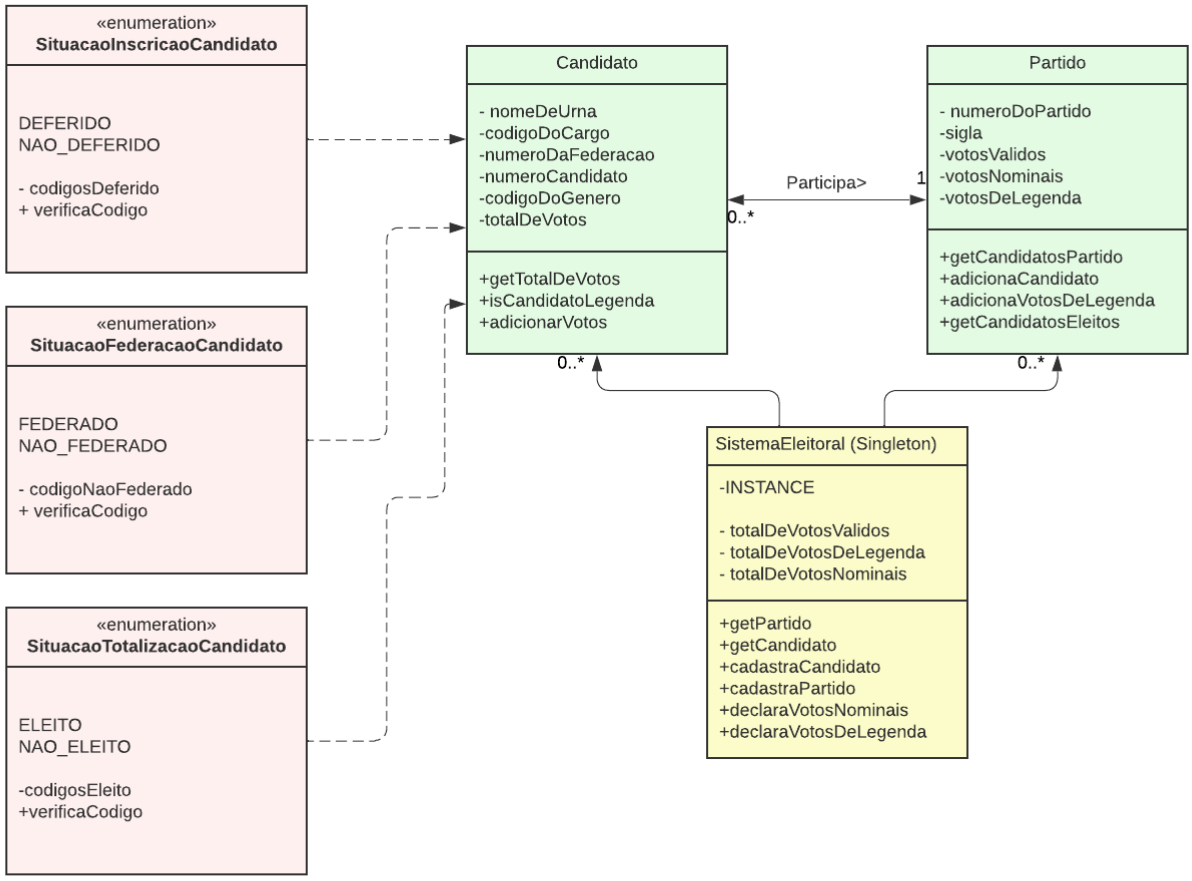
A classe onde se encontra main, chamada App, engloba a main e quatro outras funções estáticas. A importância das quatro funções estáticas além da main é para poder remover ou colocar aspas em volta de uma String, característica necessária para poder processar os dados e poder inseri-los no sistema eleitoral sendo analisado, afinal, não se pode usar certas funções de parse em Strings (exemplo: a função Integer.parseInt("\"12") retorna uma exceção porque esse parâmetro não é um número). Foi cogitado o uso da função String.replaceAll, mas essa opção foi descartada por ser uma má prática de programação considerando que existem diversas possibilidades que isso poderia gerar problema (o delimitador do CsvReader é o símbolo ';'. Supondo que existissem aspas duplas no meio de uma String, isso poderia gerar graves problemas de execução no caso de se usar a função String.replaceAll). Apesar dessa importância, a função unquote para vetores de Strings não foi usada.

A main começa recebendo os argumentos da linha de comando e trabalhando com eles para ver se o código pode continuar. Depois disso, vem a declaração dos CsvReaders e do sistema, que será o objeto mais usado ao longo da execução do programa. Então são declarados vetores para definirem os números dos headers no código, que irão definir as posições apenas dos valores extraídos do csv que serão usados.

O código tem dois loops, que usam da função booleana CsvReader.hasNextValue() nos dois CsvReader diferentes. O primeiro loop serve para registrar os candidatos e partidos do arquivo de candidaturas. O segundo para registrar as votações nos candidatos ou partidos registrados anteriormente. Em cada iteração dos loops lê-se uma linha do arquivo sendo trabalhado e partir do que é extraído decide se aquela iteração deve ser ignorada (instrução continue) ou se deve haver inscrição de candidato no sistema (para o primeiro loop) ou declaração de votos nominais ou de legenda (para o segundo loop).

Com o arquivo de votação inteiramente lido, essa parte sendo a parte mais demorada do código, fica declarada a parte do relatório. Instancia-se o objeto de Relatório com o sistema já trabalhado e atualizado, o cargo que se está votando e a data das votações, já trabalhada anteriormente. Assim são chamadas as dez funções do tipo Relatório e a saída do código inteira é gerada.

ESTRUTURA GERAL DO PROGRAMA (DIAGRAMA UML)



Descrição dos testes:

Por ser um código relativamente grande e de difícil alteração para testes específicos de funções, a maior parte dos testes das funções que não fossem de Relatorio ou de SistemaEleitoral foi feita em classes separadas com casos de teste específicos. A maior parte do trabalho foi feita usando a IDE Visual Studio Code, o que fez com que em algum ponto do

desenvolvimento do programa o arquivo csv baixado no site do TSE fosse convertido para UTF-8. Isso acabou bloqueando certas funcionalidades para testes porque o código para análise das eleições não é responsável por converter os caracteres escondidos do UTF-8 para o formato original deles. Ao perceber que existia esse problema, foi mudado o CsvReader para admitir algum formato de leitura específico, usando aqui o ISO-8859-1, formato em que os arquivos csv foram originalmente feitos.

Outros testes importantes foram os que testaram as classes mais básicas do programa (Candidato e Partido) e a ordenação das listas copiadas. Antes de haver um comparador usando função lambda, havia duas classes específicas para comparações, uma delas incluindo um bloco de declarações dentro de um try catch por conta de uma operação verbosa usando LocalDateTime para calcular idades e comparar as idades entre dois candidatos. Ao alterar o tipo para LocalDate, foi decidido também se desfazer dessas classes de comparação e implementar diretamente na função lambda, já que essas comparações só precisam ser feitas uma única vez por cópia.

Os teste mais importantes foram os que usaram os arquivos de candidatura e votação para cada estado para gerar as saídas. Houve problemas graves no que diz respeito a certas entradas e saídas que não conseguiram ser tratadas antes das testagens, sendo os principais causados por datas de nascimento não divulgadas no csv (gerando inclusive uma DateTimeParseException ao tentar trabalhar com os dados e nem terminando de rodar o código), por casos em que só há um candidato inscrito para um partido, por casos em que se vota em um partido que é de uma federação e tem votação válida sem ter candidatos inscritos (ou seja, essa votação não entra para a conta das especificações do trabalho) e por casos em que candidatos foram anulados tempos depois que a eleição já passou. Os testes (e a maioria dos erros menores de programação foram notados assim) foram majoritariamente feitos usando os arquivos de candidaturas do Espírito Santo, mas depois foram aplicados a outros estados usando o script que o professor forneceu para esses testes.

Bugs Conhecidos:

O bug mais marcante do código se trata da contabilização dos candidatos anulados pós-eleições. Esse caso faz com que o comportamento de alguns relatórios fique imprevisível quando é feita a contabilização de votos para casos que não correspondem aos códigos de deferido e que não sejam especificamente de destinação “Válido (legenda)”.

Outro bug considerável é o caso de vazar alguma String não válida no formato “dd/MM/yyyy”. Como os candidatos inscritos deferidos têm sua data de nascimento divulgada e a condição de deferimento existe no código, esse bug não é tão notável, mas ele é possível de ocorrer para certos formatos.

Ainda outro é o caso de o arquivo sendo lido ser um arquivo em formatos diferentes de ISO-8559-1. O código não converte um arquivo em outros formatos (como, por exemplo, UTF-8) para o formato padrão usado para os arquivos csv gerados pelo TSE.