Prof. Dr. Dalvan Griebler, dalvan.griebler@pucrs.br

Atualizado: 22/09/2024

TG1: Trabalho em Grupo sobre Python
Sistema de Votação

Objetivo

Durante a disciplina de programação orientada a dados, vocês estão aprendendo a programar na linguagem de programação em Python, que é multi-paradigma. Neste trabalho, o objetivo é avaliá-lo relativo a parte de Programação Orientada a Objetos (POO), Módulos e Manipulação de Arquivos. O trabalho também será uma forma de exercitar e aplicar os conceitos de forma prática.

Contexto

Você é um programador responsável por desenvolver um sistema de votação para um cenário de eleições gerais. Esse sistema precisa ser capaz de gerenciar diferentes cargos (como Governador, Presidente, Deputado Federal, Deputado Estadual e Senador), registrar os votos dos eleitores e, no final, imprimir o boletim de urna com os resultados.

Restrições

- Está permitido o uso de todos os métodos da classe str (string).
- Apenas os módulos **abc** e **sys** podem ser importados para implementar o sistema.
- Outros métodos como **len()** que são nativos e não dependem de importação de módulos estão autorizados (lista de métodos). **Métodos que não foram apresentados em aula, devem ser justificados e explicados através de comentário no código.**

Especificações

Você precisa desenvolver um sistema de votação que possa ser usado como um pacote do Python e também uma aplicação em Python que utilize este pacote para realizar uma sequência de operações que estão pré-definidas em um arquivo.

Especificações e requisitos do sistema de eleições:

1) Todo este sistema deve ser implementado como um pacote do Python chamado **Urna**, composto por um módulo **sistema**.
2) Precisa ter uma classe **Partido**, possuindo os seguintes dados:
    a) Nome (no máximo 50 caracteres)
    b) CNPJ (no máximo 14 caracteres)
    c) Número (no máximo 2 caracteres): Com o número do partido.
3) Precisa ter uma classe **Pessoa**, possuindo os seguintes dados:
    a) Nome (string com no máximo 50 caracteres)
    b) CPF (string com no máximo 11 caracteres)

c) Data de Nascimento.

4) Precisa ter uma subclasse **Candidato**, que estende a classe Pessoa possuindo os seguintes dados:
   a) **Número**, indica o número do candidato para votação.
   b) **Partido**, recebe o objeto do Partido correspondente do candidato.
   c) **Propostas**, descrição breve das propostas do candidato.

Esta classe precisa conter o método abstrato **verifica_numero_cargo**, que será futuramente implementado pelas subclasses. O setter do atributo número na classe Candidato deverá implementar esta função, garantindo que o número contenha a quantidade de dígitos conforme o cargo.

5) Precisa ter as seguintes subclasses estendidas da classe Candidato: **DepFederal, DepEstadual, Senador, Governador e Presidente**. Estas classes precisam implementar o método **verifica_numero_cargo**, este retornando verdadeiro caso o candidato tenha o número de acordo com o cargo e um erro caso esteja em desacordo. Segue a quantidade de digitos esperadas para cada cargo:
   a) Presidente: 2 digitos;
   b) Governador: 2 digitos;
   c) Senador: 3 digitos;
   d) DepFederal: 4 digitos;
   e) DepEstadual: 5 digitos.
6) Implemente a classe **Eleicao**, responsável por gerenciar todos os candidatos e partidos da eleição através de listas e métodos de busca.
7) Precisa ter uma Classe chamada **Votacao**, que armazena todas as escolhas de candidatos de um eleitor. Esta classe precisa implementar um **método de classe** responsável por contar quantos eleitores votaram.
8) Uma eleição costuma ter diversas urnas, por isso implemente a classe **Urna**, esta deve agregar os dados da eleição e da votação. Através do conceito de sobrecarga de operadores implemente uma contagem de votos entre as urnas, permitindo a soma de dois objetos da classe Urna, concatenando o ID das mesmas (ex. 1_2) e a lista das votações.
9) É necessário implementar controle de acesso aos dados dos candidatos e partidos. Você pode optar por usar decoradores ou descritores.
10) É necessário que sejam implementadas classes para tratamento de exceções quando ocorrerem operações incorretas. Por exemplo, número de digitos diferente do cargo esperando.
11) Também é necessário tratar erros com tipos de dados inválidos e incoerentes (por exemplo, nome, cpf, etc).
12) Caso queira, você pode implementar a classe **Utils**, com métodos estáticos que auxiliarão no processamento de toda a aplicação.

Especificações e requisitos da aplicação de eleição:

1) A aplicação deve receber como argumento na linha de comando os arquivos de texto (eleicaoRS.txt, urna1.txt, urna2.txt, …), onde o primeiro arquivo contém os dados da eleição e os subsequentes os votos por urna.
2) Após popular as classes com os dados do arquivo de texto recebido como argumento da linha de comando, a aplicação deverá gerar os seguintes arquivos de saída:
   a) **boletimUrna.txt**: este boletim deve ter as seguintes informações: id da urna, todos os candidatos da eleição com seus nomes, números e quantidade recebida de votos, agrupados por cargo.

b) **contabilizacao.txt**: Contendo quem ganhou a eleição para cada cargo, composto pela quantidade total de eleitores (use o método da classe implementado para isto), nome, número e quantidade de votos recebidos.

3) A aplicação não pode parar quando se deparar com ações/operações/transações não permitidas ou inválidas (ex.: votos em candidatos que não existem, candidatos com quantidade de digitos inválidos, CPF inválido etc). Os erros devem ser armazenados em um arquivo de log.

Relato

Faça um resumo de no máximo 400 palavras relatando as dificuldades, desafios ou outras observações que achar relevante comentar sobre a realização deste trabalho. Você também pode comentar sobre sua evolução no conhecimento. Esse resumo deve ser enviado junto dos demais arquivos.

Entrega

Você deve enviar como pull request contendo o seguinte:

- Urna (diretório do pacote)
- Files (diretório aonde devem estar os arquivos lidos e gerados)
    - Entrada (arquivos de entrada)
    - Saida (logs e relatórios)
- app.py (aplicativo que simula o sistema)
- README.md (arquivo descrevendo como usar o aplicativo e o pacote)
- RESUMO.md (arquivo contendo o resumo de 400 palavras)