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

Atualizado: 18/04/2024, Co-autor: Gabriell Alves de Araujo

TG1: Trabalho em Grupo sobre Python

Sistema de ensino de idiomas

Durante a disciplina de programação orientada a dados, vocês estão aprendendo a programar na linguagem de programação 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 da disciplina de forma prática.

Contexto

Você acaba de ser contratado por uma empresa, a qual visa implementar um sistema de ensino de idiomas que permita que estudantes possam ter aulas particulares através de vídeo chamadas com professores de diferentes países.

Restrições

- Está permitido o uso de todos os métodos da classe str (string).
- Apenas os módulos **abc**, **sys**, e **datetime** 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 comentários no código.**

Especificações

Você precisa desenvolver um sistema de aprendizagem de idiomas 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 estarão pré-definidas em um arquivo.

Especificações e requisitos do sistema de aprendizagem de idiomas:

1) Todo este sistema deve ser implementado como um pacote do Python chamado **LearnLanguages**, composto por um módulo **Users**. Subpacotes e submodulos são opcionais.
2) Todos os atributos das classes precisam ter seu acesso gerenciado por setters e getters.
3) Caso existam valores inválidos, o programa deve lançar exceções, as quais devem ser tratadas para que o programa não pare de funcionar na aplicação (**app.py**).
4) Implementar a classe **Idioma**, a qual armazena o nome do idioma (50 caracteres no máximo) e nível de proficiência (A1, A2, B1, B2, C1, C2, ou Nativo).
5) Implementar a classe **Horario** para representar os horários em que o professor está disponível para lecionar as aulas de idiomas. O pacote datetime pode ser utilizado para representar a data. Métodos são opcionais.
6) Implementar a classe **TipoDeAula** para identificar os diferentes tipos de aulas oferecidos pelos professores. Cada tipo de aula possui informações tais como idioma, preço, e identificador.
7) Implementar a classe **Aula**. Uma aula reúne informações como professor, estudante, horário, e tipo da aula.
8) Implementar a classe **Carteira**, a qual serve para permitir ao estudante pagar o preço das aulas aos professores. A classe Carteira deve permitir depositar, sacar, e fazer transferências de dinheiro entre carteiras.
9) Implementar a classe abstrata **Usuario**:

a) Atributos:
    i) Nome do usuário com no maximo 50 caracteres.
    ii) E-mail com no maximo 50 caracteres.
    iii) País de origem com no máximo 10 caracteres.
    iv) País atual com no máximo 10 caracteres.
    v) Lista de idiomas que o usuário sabe falar.
    vi) Lista de horários do usuário.
    vii) Objeto do tipo carteira, necessário para efetuar ou receber pagamentos das aulas.
b) Métodos:
    i) ***imprimirRelatorio***: Método abstrato que deve ser implementado pelas classes que herdam a classe **Usuario**. O método deve imprimir um relatório completo sobre as atividades do usuário.
    ii) Demais métodos que deseja implementar.

10) Implemente a classe **Estudante** como uma subclasse de **Usuario**:
a) Atributos:
    i) Lista de idiomas que o estudante deseja aprender.
    ii) Lista de professores do estudante.
b) Métodos:
    i) ***imprimirRelatorio***:
        (1) Listar Idiomas que o estudante sabe falar;
        (2) Listar Idiomas que o estudante deseja aprender;
        (3) Listar horários do estudante;
        (4) Listar professores do estudante;
        (5) Listar aulas concluídas pelo estudante;
        (6) Exibir dados do professor favorito do estudante (com qual professor o estudante teve mais aulas, e quantas foram);
        (7) Exibir idioma favorito do estudante (com qual idioma o estudante teve mais aulas, e quantas foram);
        (8) Exibir saldo da carteira do estudante.
    ii) ***agendarAula***: Efetua o pedido de agendamento de uma aula com um professor. O pedido contém o horário e o tipo de aula. O estudante só pode realizar o pedido de agendamento se tiver saldo o suficiente na carteira.
    iii) ***confirmarQueAulaFoiConcluida***: O estudante confima se a aula com o professor foi concluída. Se sim, 90% do valor do preço da aula deve ser transferido para a carteira do professor. Os 10% restantes do valor devem ser transferidos para a carteira do sistema. Caso a aula não tenha sido concluída, o dinheiro deve permanecer na carteira do estudante.

11) Implementar a classe **Professor** como uma subclasse de **Usuario**:
a) Atributos:
    i) Lista de idiomas que o professor ensina.
    ii) Lista de estudantes do professor.
b) Métodos:
    i) ***imprimirRelatorio***:
        (1) Listar Idiomas que o professor sabe falar;
        (2) Listar Idiomas que o professor ensina;
        (3) Listar horários do professor;
        (4) Listar de estudantes do professor;
        (5) Listar aulas concluídas pelo professor;
        (6) Exibir saldo da carteira do professor.
    ii) ***aceitarPedidoDeAgendamento***: Se o professor aceitar um pedido de agendamento, o horário do professor e do estudante devem ser marcados

como não disponíveis. Usuários não podem ter duas aulas marcadas no mesmo horário.

12) Fornecer um método para imprimir o relatório do sistema, o qual deve informar a quantidade de estudantes e professores cadastrados, bem como o saldo da carteira do sistema. O método também deve imprimir o relatório de cada usuário cadastrado no sistema.

Especificações e requisitos da aplicação de teste do pacote LearnLanguages:

1) A aplicação deve receber como argumento na linha de comando uma lista de arquivos texto (entrada_1.txt, entrada_2.txt, entrada_3.txt, ...), onde cada arquivo representa uma simulação do sistema sendo utilizado por estudantes e professores.
2) Após concluídas todas as ações contidas no arquivo de texto recebido como argumento da linha de comando, os dados do sistema precisam ser armazenados em um arquivo de saída (saida_1.txt, saida_2.txt, saida_3.txt, ...).
3) A aplicação não pode parar quando se deparar com ações não permitidas ou inválidas. Os erros devem ser armazenados em um arquivo de log (log_1.txt, log_2.txt, log_3.txt, ...). Por exemplo, se não foi possível agendar a aula de um estudante, o programa deve progredir para a próxima ação da simulação. Este arquivo de log serve para monitorar o que foi possível realizar ou não.

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 no repositório do GitHub

- LearnLanguages (diretório do pacote)
- Files (diretório onde devem estar os arquivos lidos e gerados)
- 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)