Código: 98H04-06 POD - Programação Orientada a Dados
Prof. Dr. Dalvan Griebler, dalvan.griebler@pucrs.br

Atualizado: 20/09/2023

TG1: Trabalho em Grupo sobre Python
Sistema de Jogo de Cartas

Contexto

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.

Depois de aprender Python na disciplina de POO, vocês estão sendo desafiados a testar seus conhecimentos implementando um sistema de jogo de cartas para se divertirem entre as sessões de estudo. O sistema todo orientado a objetos será para dar suporte aos jogos Blackjack (21) e Uno.

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 ou a aplicação bancária.
- 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 jogadores de cartas que possa ser usado como um pacote do Python e também uma aplicação em Python que utilize este pacote para simular uma sequência de jogos que estarão pré-definidos em arquivos.

Especificações e requisitos do sistema:

1) Todo o sistema deve ser implementado como um pacote do Python chamado jogoCartas, composto de um módulo chamado Jogador. Espera-se que outros colegas possam usar este pacote para implementar uma aplicação no futuro.
2) Implementar uma classe abstrata Jogador com as seguintes especificações:
    a) Jogadores possuem: nome, cpf, saldo e uma lista de cartas
    b) Independente do jogo, todos os jogadores devem ser capazes de:
        i) fazerJogada – Diferente para cada tipo de jogo
        ii) atribuirCarta – Diferente para cada tipo de jogo
    c) Lembre-se de utilizar setters e getters.
3) Um Baralho consiste em:
    a) naipes = ['Paus', 'Ouros', 'Copas', 'Espadas']
    b) valores = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'Valete', 'Dama', 'Rei', 'Ás']
4) Você deve implementar uma classe Deck que mantém os baralhos do jogo atual em um **arquivo. O Deck deve ser mantido em arquivo e não em memória**. Uma alternativa para facilitar a implementação é

criando uma classe chamada Deck. Deck pode conter mais de um baralho. O deck precisa implementar métodos como:

a) Embaralhar – As cartas removidas devem voltar ao baralho em uma nova configuração. O método embaralhar deve receber uma **seed (semente)** específica.
b) darCarta – Remove uma carta do topo do baralho
c) Número máximo de baralhos permitidos por Deck é 6.

5) Cada tipo de jogo precisa ter uma subclasse de Jogador
6) Regras do Blackjack:
   a) Os jogadores devem tentar somar o valor mais próximo de 21.
   b) Valete, Dama, Rei valem 10 pontos. Ás pode valer 1 ou 11 pontos, de acordo com o que for melhor para a situação atual do jogador. O restante das cartas tem o valor dos pontos equivalentes ao valor da carta.
   c) Jogadores competem apenas com o Dealer, que sempre tem condição de parada >=17. Considere que o dealer tem um saldo infinito.
   d) Jogadores de BlackJack possuem uma condição de parada fixa e imutável, que é específica para cada jogador (Dealer é 17, outros jogadores podem ter 20, 15 e assim por diante).
   e) Se o jogador estourar > 21, ele perde, independente do dealer.
   f) Se o jogador obtiver pontos < 21 e o dealer estourar, ele ganha.
   g) Se o jogador obtiver pontos <= 21 e pontos > dealer, ele ganha.
   h) Quando o jogador ganhar, ele dobra seu saldo.
   i) Às + 10 pontos o saldo do jogador é multiplicado por 3.
   j) Se os pontos do jogador = dealer, o saldo é retornado.
   k) Considere que o jogador sempre dá all in nos seus pontos.
   l) Se o jogador não possuir saldo, ele não pode jogar a rodada.
   m) As cartas são distribuídas uma para cada jogador por vez, terminando pelo Dealer. Ou seja, não atribua mais de uma carta por vez ao mesmo jogador.
7) Regras Uno:
   a) Jogadores começam com 5 cartas na mão. Inicialmente, as cartas são distribuídas uma para cada jogador por vez, ou seja, não atribua mais de uma carta por vez ao mesmo jogador.
   b) 1 carta começa na mesa, não oriunda de nenhum jogador.
   c) O jogador pode posicionar uma carta da sua mão sobre a carta da mesa, desde que seja ou do mesmo naipe, ou do mesmo valor.
   d) Caso não possuam jogadas, os jogadores devem adquirir novas cartas até que seja possível jogar.
   e) O jogador que ficar sem nenhuma carta na mão primeiro vence.
   f) Se acabarem as cartas do baralho sem nenhum vencedor, então o jogador que possuir a menor quantidade de cartas vence.
8) Você deve utilizar sobrecarga de operadores da classe JogadorBlackjack para adicionar, subtrair e multiplicar saldo.
9) Você deve utilizar sobrecarga de operadores na classe JogadorUno para comparar o número de cartas entre os jogadores. Exemplo: Jogador 1 > Jogador 2 deve retornar True quando o jogador 1 possuir mais cartas do que o jogador 2.
10) Jogadores com CPF inválido não podem jogar
11) Você deve implementar uma aplicação **app.py**, passando como argumento arquivos que possuem os jogos. Todos os jogos tem parâmetros descritos em arquivos: **jogo_1.txt, jogo_2.txt, ..., jogo_n.txt**
    a) Exemplo de uso: **python app.py jogo_1.txt jogo_2.txt**

Padrão dos arquivos:
NomeJogo(string)--NumeroJogadores(int)--NumeroBaralhos(int)--seed(int)--rodadas(int)
NomeJogador1(string)--cpf(string)--saldo(int)--condiçãoParada(int)

Exemplo jogo Blackjack:
Blackjack--3--3--101--5
Dom Pedro Segundo--12345678901--100--21
Mara Cutaia--87451236912--100--17
Galvan Driebler--50321478908--100--16

Exemplo jogo Uno:
Uno--3--3--101--5
Dom Pedro Segundo--123456789--100--0
Mara Cutaia--874512369--100--0
Galvan Driebler--503214789--1000--0

12) Cada jogo possui um número de rodadas.
    a) Blackjack não re-embaralha a cada rodada.
    b) Uno re-embaralha a cada rodada, somando + 1 a seed cada vez.
    c) No caso do blackjack, jogadores sem saldo não podem continuar a jogar. É possível que todos os jogadores fiquem sem saldo, nesse caso, você deve terminar o jogo atual.
    d) Os jogadores de Uno não precisam de saldo para jogar.
    e) Ao final de cada rodada, você deve reportar os resultados, escrevendo um arquivo de saída referente ao jogo.
        i) jogo_1.txt → jogo_1.saida
        ii) jogo_2.txt → jogo_2.saida
    f) Preste atenção para os formatos de saídas esperadas:
        i) Exemplo de saída Blackjack:
        Começando jogo 1
        Começando Rodada 1
        Dealer: Jogador: Dealer CPF: 00000000000 pontos: 19 saldo: 1
        Jogador: Dom Pedro Segundo CPF: 12345678901 pontos: 30 saldo: 0
        Jogador: Mara Cutaia CPF: 87451236912 pontos: 20 saldo: 400
        Jogador: Galvan Driebler CPF: 50321478908 pontos: 17 saldo: 0
        Começando Rodada 2
        Dealer: Jogador: Dealer CPF: 00000000000 pontos: 24 saldo: 1
        Jogador: Mara Cutaia CPF: 87451236912 pontos: 31 saldo: 0
        Começando Rodada 3
        Todos os jogadores estão sem saldo!

        ii) Exemplo de log Blackjack:
        ERRO: número de baralhos inválido

        iii) Exemplo de saída Uno:
        Começando jogo 2
        Começando Rodada 1
        Vencedor: Jogador: Galvan Driebler CPF: 50321478908 Número cartas: 3
        Começando Rodada 2
        Vencedor: Jogador: Mara Cutaia CPF: 87451236912 Número cartas: 0
        Começando Rodada 3
        Vencedor: Jogador: Galvan Driebler CPF: 50321478908 Número cartas: 0
        Começando Rodada 4
        Vencedor: Jogador: Mara Cutaia CPF: 87451236912 Número cartas: 4
        Começando Rodada 5
        Vencedor: Jogador: Galvan Driebler CPF: 50321478908 Número cartas: 2

iv) Exemplo de log Uno:
ERRO: número Inválido de jogadores

13) Assuma que os arquivos que descrevem os jogos podem conter **erros**. Você deve tratar as **exceções** sem que o programa falhe. Algumas exceções que necessitam de tratamento:
    a) Condição de Parada Inválida – Atribuir 16
    b) Saldo Inválido – Atribuir 1
    c) Baralho vazio – Encerrar jogo atual

Lembre-se que outras exceções podem ocorrer, você deve reportar todas elas, seguindo os jogos e rodadas sem que o programa termine. Todos erros e Exceções devem ser armazenados em um arquivo de saída. Por exemplo, jogo_1.log, jogo_2.log e assim por diante.

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

Vocês devem enviar um arquivo compactado .zip contendo:

- **jogoCartas** (diretório do pacote)
- **saidas** (diretório onde devem estar os arquivos de saída e log gerados)
- **app.py** (aplicativo que simula os jogos)
- **README.md** (arquivo descrevendo como usar o aplicativo e o pacote)
- **RESUMO.md** (arquivo contendo o resumo de 400 palavras)