Karetní hra „Oko bere“

Úkol: Implementujte zjednodušenou verzi této hry!

Pravidla:

Hraje se s balíčkem o 32 kartách (7, 8, 9, 10, kluk (J – Jack), dáma (Q – Queen), král (K – King) a eso (A – Ace) v srdcích, kárách, křížích a pikách). Hru řídí bankéř - dealer. Bankéř na začátku hry rozdá každému hráči jednu kartu.

Hodnoty karet se v oku počítají následovně:

- Karty s číslicí 7, 8, 9, 10 mají právě tuto hodnotu,
- Esa jsou za 11 bodů,
- Obrázky (kluk, dáma, král) se počítají za 1 bod.

Cílem hry je dosáhnout, nebo se co nejvíce přiblížit součtu jednadvacet neboli oka. Bankéř postupně nabízí hráčům další a další kartu (tj. prvnímu, druhému, třetímu… hráči po jedné kartě), dokud hráč není se svým součtem spokojen a neřekne dost (a tím nepřestane hrát). Pozor, když součet hodnot karet hráče přesáhne 21, je mu přiděleno 0 bodů a dále už nehraje. Hra končí, kdy žádný z hráčů nemá zájem o další kartu.

Implementace – třída `Dealer`:

Vytvořte třídu `Dealer`, která bude zodpovědná za průběh hry. Obsahuje reprezentaci balíčku karet `deck` jako seznam znakových řetězců s označením karty (například: '♥A', '♦10', '♣J' nebo '♠K')

Třída `Dealer` implementuje následující metody:

`shuffle()` - zamíchá karty, takže jejich pořadí v seznamu (tj. v balíku) bude náhodné.

`deal(n)` - vrátí (tj. rozdá) n karet ze začátku seznamu (tj. z vrchu balíčku). Metoda vrací seznam karet (pořadí nehraje roli) v požadovaném počtu (případně méně, pokud v balíčku už není dostatek karet) nebo prázdný seznam, pokud už v balíčku žádné karty nejsou.

Příklad volání těchto metod:

```
dealer = Dealer()
dealer.shuffle()
myHand = dealer.deal(5)
print(myHand)
dealer.shuffle()
myHand = dealer.deal(3)
print(myHand)
```

… a výsledku:

```
['♠Q', '♥9', '♣7', '♥K', '♥10']
['♣8', '♣7', '♠J']
```

Další metody třídy `Dealer`:

`addPlayer(player)` – přidá hráče do hry

`removePlayer(player)` – odebere hráče z hry

`startRound()` – zahájí kolo, tj. jednotlivým hráčům rozdá po jedné kartě a pak jim postupně nabízí další karty až do té doby než i poslední hráč řekne „dost“.

`announceWinner(self)` – když už žádný z hráčů nechce další kartu, vyhlásí se vítěz a vypíše se bodový zisk všech hráčů v klesajícím pořadí

Implementace – třída `Player`:

Implementujme i třídu `Player`, tj. hráče, jež bude dle určité strategie hrát hru Oko.

Třída typicky může (ale nemusí) implementovat tyto atributy:

- `name` – jméno hráče
- `strategy` – strategie, viz. níže
- `hand` – seznam karet v držení

Třída `Player` má následující metody:

`getHandValue()` - vrátí celkovou hodnotu karet podle výše uvedených pravidel

`needsCard()` – vrací `True`, pokud hráč podle své strategie potřebuje další kartu

`acceptCard(cards)` - přijme nabízené karty (v této implementaci to bude vždy jen jedna karta) a vypíše všechny kary v držení hráče na konzoli. Tuto metodu volá `Dealer`, když předchozí `needsCard()` vrátí `True` – hráč tedy potřebuje další kartu.

Implementujme následující 3 strategie:

(Pozn.: protože neznáme dědičnost tříd v Python, všechny strategie budeme implementovat v jediné třídě `Player`.)

- `'Cautious'`: Akceptuje další karty od Dealer jen pokud celková hodnota držených karet nepřesáhne 10.
- `'Bold'`: Podobně, akceptuje další karty od Dealer jen pokud celková hodnota držených karet nepřesáhne 15.
- `'Human'`: Spoléhá na vstup od uživatele, tj. vypíše současné držené karty na konzoli a zeptá se uživatele, zdali si přeje další kartu.

Po implementaci výše uvedených může následující kód:

```
newDealer = Dealer()
player1 = Player('Čeněk Člověčí', 'Human')
player2 = Player('Vilda Vopatrný', 'Cautious')
player3 = Player('Olda Odvážný','Bold')
newDealer.addPlayer(player1)
newDealer.addPlayer(player2)
newDealer.addPlayer(player3)
newDealer.shuffle()
```

`newDealer.startRound()`

...simulovat jedno kolo hry např. takto:

```
--- rozdávám  ---
Čeněk Člověčí má nyní karty: ♣K
Vilda Vopatrný má nyní karty: ♦7
Olda Odvážný má nyní karty: ♥9
--- rozdávám  ---
Čeněk Člověčí má nyní karty: ♣K v hodnotě 1, chce další(A/N)?:A
Čeněk Člověčí má nyní karty: ♣K ♥8
Vilda Vopatrný má nyní karty: ♦7 ♣8
Olda Odvážný má nyní karty: ♥9 ♥A
--- rozdávám  ---
Čeněk Člověčí má nyní karty: ♣K ♥8 v hodnotě 9, chce další(A/N)?:A
Čeněk Člověčí má nyní karty: ♣K ♥8 ♦A
--- rozdávám  ---
Čeněk Člověčí má nyní karty: ♣K ♥8 ♦A v hodnotě 20, chce další(A/N)?:N
--- kolo skončilo ---
Hráč Čeněk Člověčí vyhrává se ziskem 20 bodů
Vyhrává také hráč Olda Odvážný se ziskem 20 bodů
Hráč Vilda Vopatrný získal 15 bodů
```

Poznámka

Nepoužívejte metody `random.shuffle()` ani `list.sort()`!

Volitelný úkol

1. Modifikujte kód tak, aby `dealer` nikdy nedošli karty, resp. implementujte metodu `newDeck()`, jež vrátí nový (uspořádaný) balík karet, a kterou bude `dealer` volat vždy, když potřebuje doplnit karty do `deck` - tedy v případě, kdy `deck == []`.

2. Implementujme další pravidlo hry Oko:

„Srdcová sedma, nejvýhodnější karta ve hře, tzv. Šantal, počítá se dle potřeby hráče za 1, 7, 10 nebo 11 bodů.“

...jako funkci `getHandValues()` třídy `Player`, vracející seznam (obsahující 1 nebo 4 hodnot) možných hodnot držených karet.

Využitím vrácených hodnot vytvořme specifickou strategii.

Šablona k implementaci i tento dokument jsou dostupné na adrese:

https://gitlab.fel.cvut.cz/seredlad/challenges/tree/master/python/blackjack