Wojna

Treść zadania

Chcemy zasymulować różne rodzaje gier karcianych, podobnych do klasycznej wojny. Każda karta ma 3 własności:

- **Wartość** - liczbę całkowitą z zakresu 1-10.
- **Kolor** - jeden spośród 4. Na potrzeby zadania przyjmiemy wartości kolorów: trefl - 1, karo - 2, kier - 3, pik - 4.
- **Rzadkość** - karta może być rzadka albo nie.

Każdy rodzaj wojny powinniśmy móc przeprowadzić ze śledzeniem jej przebiegu na żywo. Trudność zadania polega na tym, że prawdopodobnie w przyszłości powstanie dużo więcej rodzajów wojen (patrz wskazówka). Wynik wojny reprezentujemy przez 3 wartości: 1 gdy wygrała talia pierwsza, -1 gdy wygrała talia druga oraz 0 w przypadku remisu. Póki co, mamy 3 rodzaje wojen między taliami kart:

- `RarityBattle` - wygrywa ta talia, która posiada więcej kart rzadkich. W przypadku tej samej liczby w obu taliach, dochodzi do remisu.
- `ColorBattle` - wygrywa ta talia, której suma wartości kolorów wszystkich kart jest większa . W przypadku tej samej wartości w obu taliach, dochodzi do remisu.
- `NormalBattle` działa jak w typowej grze karcianej wojna: bierzemy pierwszą kartę z obu talii, a następnie porównujemy ich wartości. Kartę, która przegrała, odrzucamy, a wygrana będzie porównywana z następną kartą przeciwnej armii. Jeśli mieliśmy remis, obie karty są odrzucane (i bierzemy kolejne z obu talii). Wygrywa ta talia, w której na końcu zostanie choć jedna karta.

W zadaniu będziemy randomizować talie kart. Każda talia może być jednoznacznie opisana przez liczbę kart oraz ziarno losowania obiektu klasy `Random`. Talie powinny być randomizowane w następujący sposób[1]:

1. Tworzenie obiektu `Random` o zadanym seedzie.
2. Dla każdej karty:
   (a) Wylosowanie wartości z przedziału 1-10 przy użyciu stworzonego obiektu.
   (b) Wylosowanie koloru (spośród wcześniej wymienionych) przy użyciu stworzonego obiektu.
   (c) Wylosowanie `boolean`'a (odpowiadającego rzadkości karty) przy użyciu stworzonego obiektu.

Oczekiwany output:

```
Deck with 5 cards, seed 10: (4, 2, false) (1, 1, true) (8, 2, true) (5, 2, true) (2, 4, true)
Deck with 5 cards, seed 15: (2, 4, false) (8, 4, false) (10, 1, true) (1, 3, false) (6, 3, false)
Sizes 5, seeds 10 15, color battle result: -1
Sizes 5, seeds 15 10, color battle result: 1
Sizes 5, seeds 30 30, color battle result: 0
Sizes 5, seeds 10 15, rarity battle result: 1
Sizes 5, seeds 15 10, rarity battle result: -1
Sizes 5, seeds 30 30, rarity battle result: 0
Sizes 5, seeds 10 15, normal battle result: 1
Sizes 5, seeds 15 10, normal battle result: -1
Sizes 5, seeds 30 30, normal battle result: 0
```

Uwagi

1. Należy zadbać o podstawowe zasady programowania obiektowego - enkapsulację, gettery, settery, odpowiednią dostępność zmiennych... Te rzeczy nie są uwzględnione w treści zadania, ale są to dobre praktyki.
2. **Wskazówka**: użyć interfejsu.

[1] Jest to istotne, by móc porównywać output z oczekiwanym. Inne sposoby losowania są poprawne, lecz mogą dać inne wyniki