Zadanie zaliczeniowe: Gra „ogórek”

14 lipca 2016

1 Wstęp

Zadanie zaliczeniowe dotyczy klasycznej gry karcianej „ogórek”. Program powinien pracować w dwóch trybach:

- interaktywnej gry użytkownika z komputerem,
- symulatora gry porównującego różne strategie.

2 Zasady gry

W grę może grać od 2 do 8 graczy. Gra się standardową talią 52 kart. Na początku rozgrywki każdy gracz dostaje 6 kart. Gracz nr 1 wykłada na stół jedną kartę. Następnie gracze kolejno (w kółko) dokładają po jednej karcie na stół, przy czym

a wyłożona karta nie może być niższa niż ostatnio wyłożona karta (w kolejności starszeństwa: dwójka, ..., król, as; kolor nie ma znaczenia),

b jeżeli takiej karty nie ma, wtedy wykładana jest najniższa karta.

Gra toczy się przez 5 kolejek, tj. do chwili gdy każdy gracz ma 1 kartę. Wygrywa gracz (gracze), któremu pozostała najniższa karta, przy czym tym razem starszeństwo wygląda tak: as, dwójka, trójka, ..., król.

3 Założenia projektowe

Twój program powinien implementować co najmniej klasę `Gracz` oraz funkcję lub klasę `Rozgrywka`, która służy do przeprowadzenia pojedynczej gry. Klasa `Gracz` powinna mieć co najmniej 3 metody:

- Metodę `dostalesKarty(...)` wywoływaną w celu poinformowania gracza, jakie karty zostały mu rozdane (podane jako pierwszy argument tej metody). Drugim argumentem metody jest całkowita liczba graczy w rozgrywce.

- Metodę `jakiRuch()` wywoływaną w celu uzyskania od gracza informacji, jaką kartą teraz zagra (metoda nie ma argumentów, a zwraca kartę, jaką gracz postanowił zagrać). Tę samą metodę należy wykorzystać również do sprawdzenia, jaka karta została w ręce na końcu gry.
- Metodę `ruchPrzeciwnika(...)` wywoływaną w celu poinformowania gracza, jaki ruch wykonał inny gracz. Metoda ta jest wywowyławana po każdym ruchu dowolnego przeciwnika, tj. jeśli w grze uczestniczy r graczy, to każdy z graczy w ciągu kolejki otrzyma $r-1$ takich powiadomień.

Podczas trwania rozgrywki „komunikacja" z obiektem typu `Gracz` powinna zachodzić wyłącznie z wykorzystaniem powyższych funkcji.

W programie powinieneś zaimplementować co najmniej 4 typy graczy o różnych strategiach:

- *Człowiek* — metoda `jakiRuch()` polega na tym, że wypisuje na ekranie pozostałe karty i czeka, aż użytkownik sam wybierze (wpisuje z klawiatury), którą zagrać.
- *Minimalizator* — zagrywa zawsze najniższą kartę z możliwych.
- *Maksymalizator* — zagrywa zawsze najwyższą kartę z możliwych.
- *Strateg* — zagrywa kartę według wymyślonej przez Ciebie strategii.

Różne typy możesz zaimplementować np. jako osobne klasy (być może dziedziczące po jednej klasie bazowej) lub jako jedną klasę `Gracz` z parametrem `strategia`.

4 Działanie programu

Zaraz po uruchomieniu program powinien zapytać o tryb pracy: interaktywnej gry lub automatycznego symulatora. Jeśli użytkownik wybierze tryb interaktywnej gry, to program powinien zapytać o liczbę graczy oraz kolejno o typ każdego z nich, a następnie przeprowadzić rozgrywkę. Sama rozgrywka mogłaby wyglądać tak (w przykładzie człowiek jest graczem 2):

```
Gracz 1 zagral karte: 6_trefl.
Na stosie lezy 6_trefl. Wybierz karte do zagrania (2_pik, 3_pik, as_kier)
as_kier  [tekst wpisany przez uzytkownika]
Gracz 2 zagral karte: as_kier.
Gracz 3 zagral karte: dama_karo.
Gracz 1 zagral karte: krol_trefl
[...]
```

Jeśli użytkownik wybrał tryb automatycznego symulatora, to program powinien zapytać o liczbę graczy, kolejno o typ każdego z nich (tym razem nie

można wybrać człowieka), a potem o liczbę rozgrywek N. Następnie program przeprowadza N rozgrywek (oczywiście każdorazowo losując karty od nowa) i wypisuje statystyki, który gracz ile razy wygrał.

5 Punktacja

50% — program się kompiluje na komputerach w OKWF i działa poprawnie w oczekiwany sposób
20% — komputerowi gracze sprawdzają, czy przeciwnicy nie oszukują, np. czy nie zagrali właśnie karty, która już leży na stole jest we własnej ręce*
20% — punkty za własną strategię (im częściej wygrywa z Maksymalizatorem i Minimalizatorem tym lepiej)
10% — czytelność kodu, dobry podział strukturalny na funkcje, klasy...

$*$ — ekstra punkty za sprawdzanie, czy przeciwnik nie kłamał co do swojej najniższej karty (np. teraz zagrał niższą, niż wcześniej kiedy powinien swoją najniższą) lub najwyższej karty.

W przypadku stwierdzenia fragmentów cudzego kodu - 0pkt.