Klasówka z Programowania Obiektowego 18 IV 2012

Arimaa jest strategiczną grą planszową przeznaczoną dla dwóch graczy, zwanych dalej złotym i srebrnym, rozgrywaną na 64-polowej planszy o wymiarach 8x8, przypominającej szachownicę. Na planszy wyróżniono 4 pola (tzw. pułapki) - są to pola c3, f3, c6 i f6 wg notacji szachowej. Każdy z graczy dysponuje kompletem 16 bierek w kolorze srebrnym lub złotym, w skład którego wchodzą (w kolejności od najsilniejszych do najsłabszych): 1 słoń (), 1 wielbłąd (), 2 konie (), 2 psy (), 2 koty () i 8 królików (). **Na załączonych diagramach bierki gracza złotego są zwrócone w prawo, a bierki gracza srebrnego są zwrócone w lewo; ponadto cztery pola pułapek zostały zaznaczone jako ciemniejsze.**

Celem gry jest umieszczenie królika w ostatnim rzędzie planszy, tj. gracz grający złotymi wygrywa, gdy umieści jednego ze swoich królików w ósmym rzędzie, zaś srebrny gracz zwycięża, gdy umieści srebrnego królika w rzędzie pierwszym.

Na początku gry plansza jest pusta. Gracz grający złotymi rozpoczyna grę od rozmieszczenia swoich szesnastu bierek w rzędach pierwszym i drugim w dowolny sposób. Następnie srebrny gracz rozmieszcza swoje bierki w rzędach siódmym i ósmym, również wedle własnego uznania. Przykładowe początkowe rozmieszczenie pokazano na diagramie 1.

Po zakończeniu fazy przygotowawczej rozpoczyna się właściwa faza gry. Tury graczy odbywają się na przemian, pierwsza tura należy do gracza złotego.

W każdej turze można wykonać posunięcia zużywające nie więcej niż 4 kroki. Możliwe posunięcia to:

- **Przesunięcie:** gracz może przesunąć wybraną swoją bierkę na sąsiednie wolne pole. Słoń, wielbłąd, koń, pies i kot mogą poruszać się do przodu, do tyłu lub na boki. Królik może poruszać się jedynie do przodu i na boki. Każde przesunięcie trwa jeden krok.
- Silniejsza bierka może przemieścić sąsiadującą słabszą bierkę przeciwnego koloru. Każde przemieszczenie zużywa dwa kroki. Istnieją dwa rodzaje przemieszczenia:
    - **Pchnięcia** dokonuje się następująco: bierkę przeciwnika (słabszą, „pchaną”) gracz przesuwa na dowolne sąsiednie wolne pole, a własna bierka (silniejsza, „pchająca”) zajmuje jej miejsce. Przykładowo na diagramie 2 złoty słoń na d3 może popchnąć srebrnego królika z d2 na e2, a następnie samemu przesunąć się na d2.
    - **Pociągnięcia** dokonuje się następująco: gracz przesuwa własną (silniejszą, „ciągnącą”) bierkę na dowolne sąsiednie wolne pole, zaś bierka przeciwnika (słabsza, „ciągnięta”) zajmuje jej miejsce. Na przykład srebrny słoń z d5 może przesunąć się na c5 i pociągnąć za sobą złotego konia z d6 na d5.

Nie można jedną bierką jednocześnie przemieszczać dwóch lub więcej bierek przeciwnika - na przykład jedną pchać, a drugą ciągnąć w tym samym czasie. Bierki o tej samej sile nie mogą się nawzajem przemieszczać. Słoń, jako najsilniejszy, nie może zostać przemieszczony przez żadną inną bierkę.

Bierka, która nie sąsiaduje z żadną bierką tego samego koloru, a jednocześnie przylega do silniejszej bierki przeciwnika, jest traktowana jako zamrożona. Zamrożone bierki nie mogą się

ruszać, ale mogą być przemieszczane przez przeciwnika. Zamrożone bierki również powodują zamrożenie słabszych bierek przeciwnika. Na przykład złoty królik na b7 jest zamrożony przez srebrnego psa, ale srebrny królik na d2 już nie, ponieważ sąsiaduje ze srebrnym koniem. Psy na a6 i b6 nie zamrażają się nawzajem, ponieważ są równej siły. Słoń, jako najsilniejszy, nie może zostać zamrożony przez żadną inną bierkę.
Bierka, która stoi na polu-pułapce i nie sąsiaduje z żadną bierką tego samego koloru, zostaje natychmiast usunięta z planszy (zbita). Srebrne mogą zbić złotego konia na d6 pchając go na pole-pułapkę c6 słoniem z d5. Jeśli srebrny królik z c4 i srebrny koń z c2 opuszczą swoje obecne pozycje, to srebrny królik na c3 zostanie zbity.
Manewr przemieszczenia może zostać wykonany nawet wtedy, gdy w jego wyniku silniejsza figura wpadnie do pułapki. Na przykład srebrny koń z f2 może przesunąć się na pole f3 (wpadając do pułapki) i pociągnąć za sobą złotego królika z f1 na f2.
Gra może się zakończyć na jeden z trzech sposobów:

- **Gol:** gracz umieszcza swojego królika w ostatnim rzędzie planszy i wygrywa.
- **Unieruchomienie:** gracz nie może wykonać poprawnego ruchu, bo wszystkie jego bierki są zamrożone bądź zablokowane[1]. Wówczas wygrywa jego przeciwnik.
- **Eliminacja:** gracz stracił wszystkie swoje króliki. W tej sytuacji również wygrywa przeciwnik.

W grze Arimaa nie ma remisów.

Zadanie

1. Zaprojektuj hierarchię klas reprezentującą pojedynczą rozgrywkę w grę Arimaa. Projektując model zadbaj o łatwość rozszerzania, np. dodanie nowej bierki czy nowego rodzaju posunięcia. W klasach wskaż odpowiednie atrybuty i metody, oznacz dziedziczenie i klasy abstrakcyjne.

2. Zaimplementuj w języku Java metodę:
   ```
   void Plansza.ruch(Posuniecie[] posuniecia);
   ```
   która odpowiada za wykonanie pojedynczej tury gry. Uzupełnij definicję klas `Plansza` i `Posuniecie` oraz zadbaj o sprawdzanie poprawności podanych parametrów. W przypadku błędu metoda ma zgłosić wyjątek `ZlyRuchException`. Zaimplementuj również wszystkie metody, konstruktory i wyjątki, z których powyższa metoda korzysta. Podczas implementacji masz do dyspozycji klasę `WspolrzednePomocnicy`, której opis znajduje się po drugiej stronie kartki z diagramami.
   Złożoność implementacji nie ma wpływu na ocenę!

Powodzenia!!!

[1] Zablokowane bierki to takie, które nie mogą wykonać żadnego z posunięć, tzn. nie ma wokół nich wolnych pól lub słabszych bierek, które dawałoby się pchnąć.

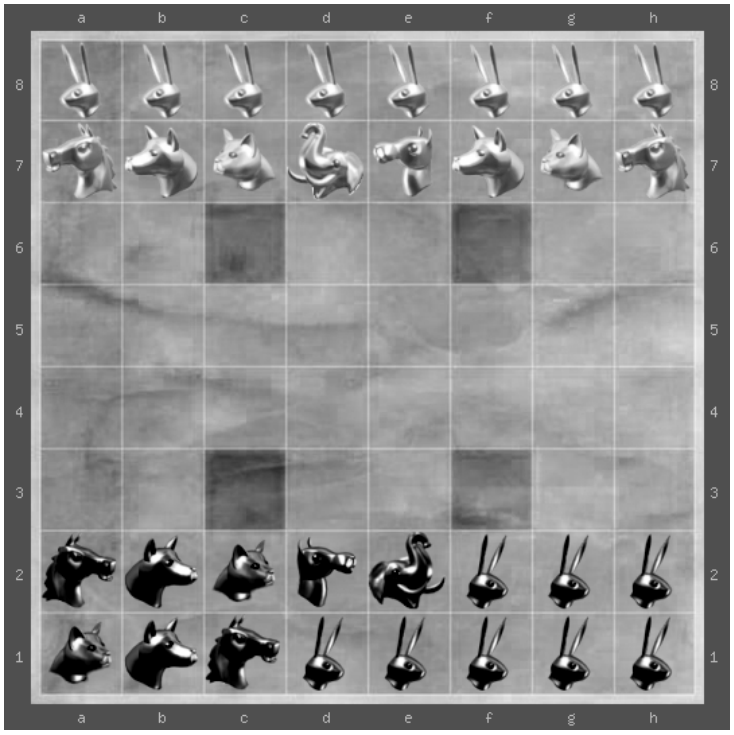

Diagram 1

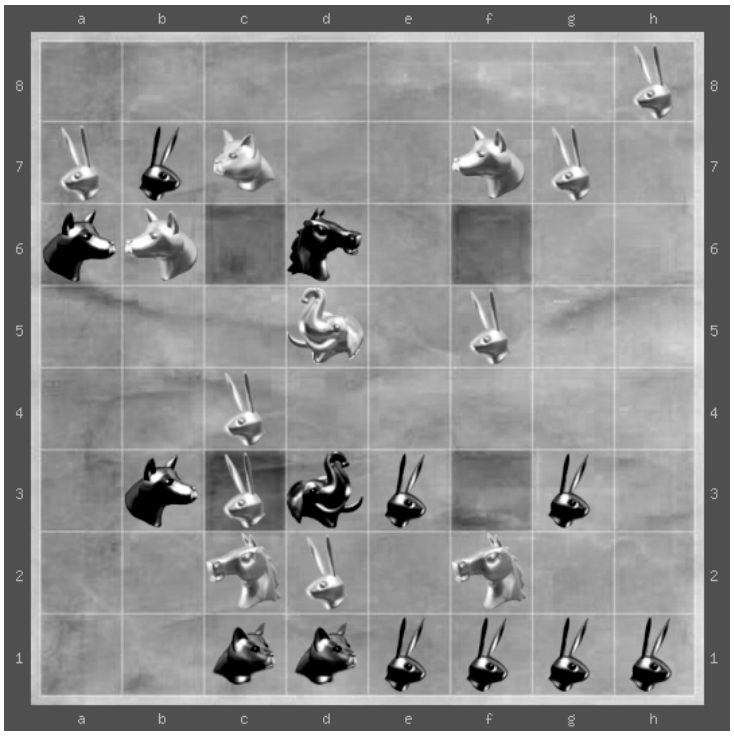

Diagram 2

```
/**
 * Klasa pomagajaca manipulowac wspolrzednymi pol na planszy.
 */
public class WspolrzednePomocnicy {
    /**
     * Stala zawierajaca wspolrzedne pol z pulapkami
     */
    public final static int[][] pulapki =
        { {2, 2}, {2, 5}, {5, 2}, {5, 5} };

    /**
     * Metoda, ktora zwraca tablice wspolrzednych sasiednich
     * pol, w kolejnosci zgodnej ze wskazowkami zegara, zaczynajac
     * od pola znajdujacego sie o wiersz wyzej.
     *
     * Przyklady:
     * sasiedniePola({4, 4}) == {{4, 5}, {5, 4}, {4, 3}, {3, 4}}
     * sasiedniePola({0, 0}) == {{0, 1}, {1, 0}}
     * sasiedniePola({7, 6}) == {{7, 7}, {7, 5}, {6, 7}}
     *
     * W przypadku podania nieprawidlowego parametru metoda zglasza
     * wyjatek IllegalArgumentException.
     */
    public static int[][] sasiedniePola(int[] pole)
        throws IllegalArgumentException {
        // ...
    }
}
```