Aufgabe 9.5 (H) Kampf der Geschlechter [30 Punkte]

Verwenden Sie für diese Aufgabe nur die erlaubten Java-Methoden (siehe Anhang und Aufgabentext).

Achtung: Zu dieser Aufgabe wird ein Programmgerüst mitgeliefert. Dieses soll verwendet und inhaltlich ergänzt werden. Insbesondere dürfen die gegebenen Methodensignaturen und Rückgabetypen nicht geändert werden. Andernfalls gilt die Aufgabe als nicht bearbeitet.

Es soll ein Brettspiel für zwei Personen (im Folgenden `M` und `W` genannt) implementiert werden, bei dem es darum geht, Tiere auf dem Brett zu bewegen. Jeder Spieler hat die selbe Anzahl an Tieren einer bestimmten Art. Das Spiel wird auf einem Schachbrett gespielt (Felder `"a1"` bis `"h8"` vom Datentypen `String`). Um zu unterscheiden, zu welchem Spieler ein Tier jeweils gehört, treten Männchen (Spieler `M`) und Weibchen (Spielerin `W`) gegeneinander an.

Es folgt zunächst eine Beschreibung der Spielregeln. Anschließend folgen weitere Angaben zur Umsetzung des Spiels als Java-Programm.

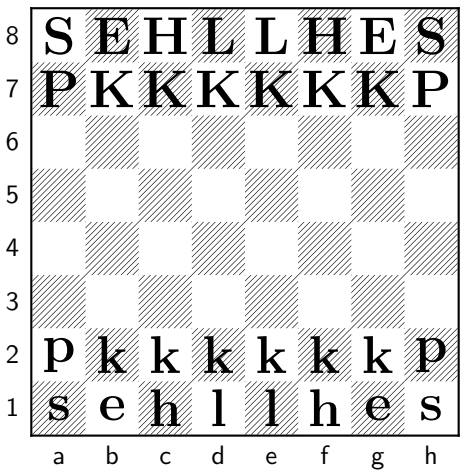
Jedes Tier ist entweder ein *Raubtier* oder ein *Vegetarier*. Die Raubtiere *fressen* die Vegetarier. Raubtiere *verhungern*, wenn sie eine bestimmte Zeit lang nicht gefressen haben. Es können nur Vegetarier gefressen werden. Die Art des Raubtiers oder Vegetariers ist dabei unerheblich. (D. h. ein Pinguin kann z. B. einen Elefanten verspeisen.) Die vorkommenden Tierarten sind:

- Vegetarier: Kaninchen, Elefant, Pferd
- Raubtiere: Leopard, Schlange, Pinguin

Die *Startaufstellung* ist ähnlich wie beim Schachspiel: Spielerin `W` hat

- sechs Kaninchen auf den Feldern `b2` bis `g2` (b2,c2,d2,e2,f2,g2)
- zwei Pinguine auf den Feldern `a2` und `h2`
- zwei Schlangen auf den Eckfeldern `a1` und `h1`
- zwei Elefanten auf den Feldern `b1` und `g1`
- zwei Pferde auf den Feldern `c1` und `f1`
- zwei Leoparden auf den Feldern `d1` und `e1`

Die Aufstellung für `M` entspricht der für `W`, aber gespiegelt an der Mittellinie des Bretts, d. h. die Tiere besetzen die Felder `a7` bis `h8`. Graphisch dargestellt (Kleinbuchstaben für Tiere von Spielerin W, Großbuchstaben für die von M) (L/l: Leopard(in), P/p: Pinguin(in), S/s: Schlange(rich), E/e: Elefant(in), K/k: Kaninchen(m/w), H/h: Pferd(engl. horse)):



Die Tiere bewegen sich nach einem weiter unten beschriebenen artspezifischen Muster fort, welches zunächst jeder Art für jedes Feld die Menge an prinzipiell möglichen Zielfeldern zuordnet, auf die gezogen werden kann. Um zu einem Zielfeld zu gelangen, müssen u. U. mehrere Zwischenfelder passiert werden. Beispiel: Eine Schlange kann vom Ausgangsfeld `a1` über die Zwischenfelder `b2` und `c1` zum Zielfeld `d2` gelangen.

Alle Zwischenfelder sind gleichzeitig auch mögliche Zielfelder. Die prinzipiell möglichen Züge der Tiere werden aufgrund von folgenden beiden Kriterien weiter eingeschränkt:

1. Auf dem Weg vom Ausgangsfeld zum Zielfeld (d. h. auf den Zwischenfeldern) dürfen keine anderen Tiere stehen.

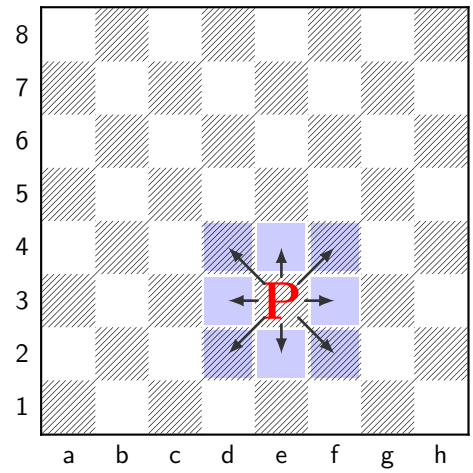
2. Auf ein besetztes Zielfeld kann gezogen werden, wenn es von einem gegnerischen Vegetarier besetzt ist und das ziehende Tier ein Raubtier ist. Das Raubtier frisst („schlägt“) dann das gegnerische Tier, welches vom Brett genommen wird und somit für den weiteren Spielverlauf nicht mehr zur Verfügung steht.

Im Folgenden werden die Zugmöglichkeiten der verschiedenen Arten beschrieben und für das Feld `e3` beispielhaft graphisch veranschaulicht.

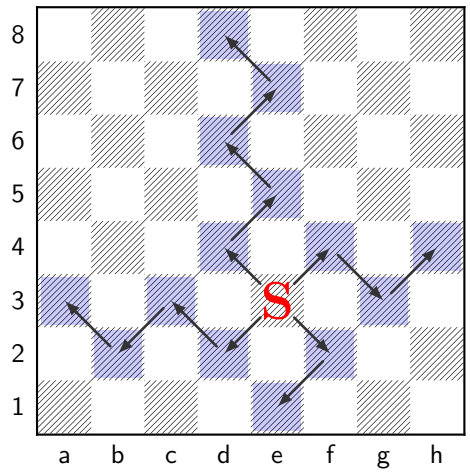
Pinguin: Bewegt sich genau ein Feld in gerader oder diagonaler Richtung.

Schlange: Bewegt sich im ersten Schritt (erstes Zwischenfeld) ein Feld diagonal. Danach bewegt sie sich diagonal weiter, wobei die Bewegungsrichtung zwischen links und rechts alterniert. Hierbei wird der Schritt auf das erste Zwischenfeld immer als Schritt nach links interpretiert. Der zweite Schritt geht also bzgl. der im ersten Schritt gemachten Bewegung nach rechts.

Mögliche Züge eines Pinguins (P) und einer Schlange (S) vom Feld e3 aus, sofern keine anderen Tiere im Weg stehen:



Zielfelder des Pinguins von e3 aus:
d4, e4, f4, d3, f3, d2, e2, f2

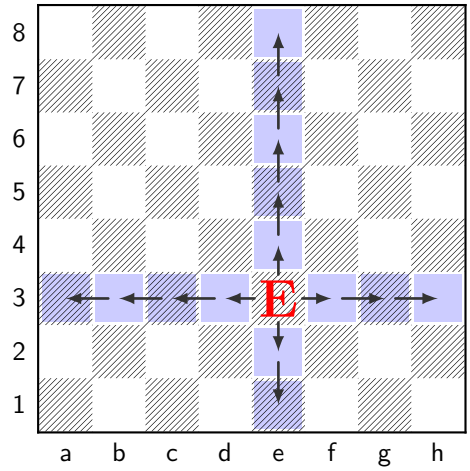


Wege der Schlange von e3 aus:
1. nach d8 via d4, e5, d6, e7
2. nach e1 via f2
3. nach h4 via f4, g3
4. nach a3 via d2, c3, b2

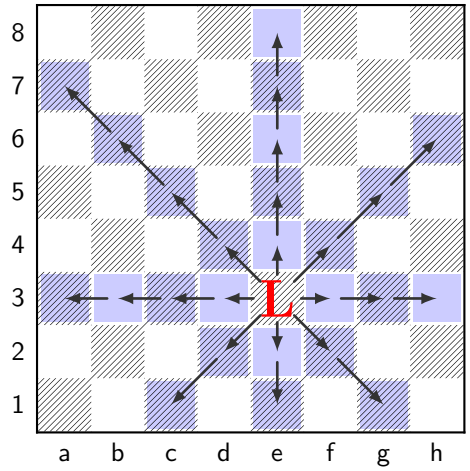
Elefant: Bewegt sich beliebig weit in gerader Richtung (entspricht einem Turm beim Schachspiel).

Leopard: Bewegt sich beliebig weit in gerader oder diagonaler Richtung (entspricht einer Dame beim Schachspiel).

Mögliche Züge eines Elefanten (E) und eines Leoparden (L) vom Feld e3 aus, sofern keine anderen Tiere im Weg stehen:



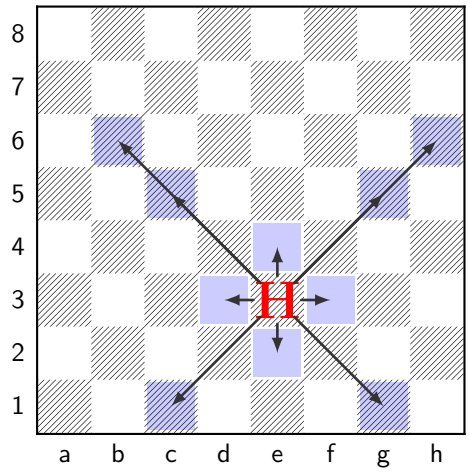
Wege des Elefanten von e3 aus:
1. nach e8 via e4, e5, e6, e7
2. nach e1 via e2
3. nach h3 via f3, g3
4. nach a3 via d3, c3, b3



Wege des Leoparden von e3 aus:
1. nach e8 via e4, e5, e6, e7
2. nach e1 via e2
3. nach h3 via f3, g3
4. nach a3 via d3, c3, b3
5. nach a7 via d4, c5, b6
6. nach h6 via f4, g5
7. nach c1 via d2
8. nach g1 via f2

Kaninchen: Das Kaninchen zieht genau gleich wie der Pinguin.

Pferd: Das Pferd darf geradeaus auf ein horizontal oder vertikal angrenzendes Feld ziehen. Außerdem darf es diagonal einmal **wahlweise zwei oder drei** Felder weit *springen*. Bei Zügen des Pferdes gibt es somit keine Zwischenfelder. Mögliche Züge eines Pferdes (H) vom Feld e3 aus:



Erreichbare Felder des Pferdes von e3 aus:
d3, f3, e2, e4, c1, g1, c5, b6, g5, h6
Keine Zwischenfelder!

Spielverlauf: Zu Beginn wird festgelegt, ob `W` oder `M` mit dem Ziehen beginnt. Der Spielverlauf gliedert sich in einzelne *Spielrunden*, in welchen zuerst die Person zieht, welche das Spiel begonnen hat, und danach die andere. Die Spielrunden werden so lange ausgeführt, bis das Spiel beendet ist. `W` und `M` ziehen in jeder Spielrunde jeweils bis zu vier eigene Tiere. Diese einzelnen Züge werden parallel ausgeführt, **nicht** nacheinander.

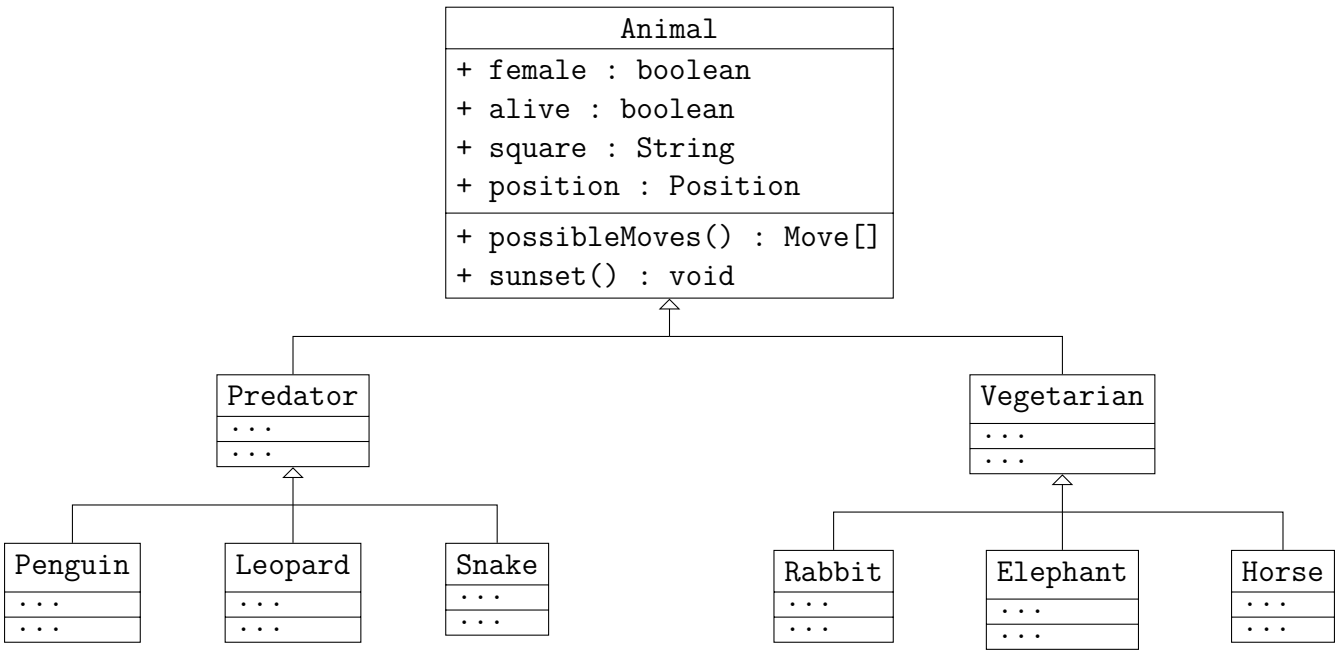
Es darf prinzipiell jeder Zug ausgewählt werden, der nach den oben beschriebenen Regeln in der Spielposition für das entsprechende Tier möglich ist. Jedoch müssen die Zielfelder der parallel ausgeführten Züge verschieden sein, d. h. zwei Tiere dürfen nicht auf das selbe Feld ziehen. Verschiedene Tiere dürfen aber beim parallelen Ausführen der Züge die selben Zwischenfelder benutzen, um zum Zielfeld zu gelangen. Jedes Zielfeld darf zugleich Zwischenfeld von parallel ausgeführten Zügen sein.

Es dürfen null bis drei Vegetarier und null bis ein Raubtier umgesetzt werden, d. h. man kann auch *passen*. Wird ein Tier vom Raubtier gefressen, wird es vom Spielbrett entfernt. Nachdem beide Spieler ihre Züge ausgeführt haben, werden die „Restlaufzeiten“ der Raubtiere aktualisiert und die verhungerten Tiere vom Spielbrett entfernt. Die Restlaufzeit bezeichnet die Anzahl Tage, welche das Raubtier noch ohne zu fressen auskommen kann. Sie wird als natürliche Zahl einschließlich der Null angegeben. Jede Spielrunde zählt als ein Tag und verringert die Restlaufzeit um 1. Ist der Wert 0 erreicht, muss das Raubtier in der folgenden Runde fressen, um im Spiel zu bleiben. Jede Tierart hat eine bestimmte Restlaufzeit. Frisst ein Raubtier in einer Spielrunde, so wird seine Restlaufzeit **sofort** auf den entsprechenden Wert zurückgesetzt. Alle Raubtiere sind zu Spielbeginn satt!

Spielende: Nach jeder Spielrunde wird ermittelt, ob das Spiel zu Ende ist. Sind alle Tiere vom Brett verschwunden, endet das Spiel unentschieden. Hat nur eine Seite keine Tiere mehr, verliert sie. Sind keine Raubtiere mehr übrig, so gewinnt, wer noch die meisten Vegetarier hat. Steht der Spielausgang während der letzten Spielrunde bereits fest, nachdem eine Seite gezogen hat, muss diese Spielrunde nicht mehr zu Ende gespielt werden.

Umsetzung:

Es ist folgende Klassenhierarchie mit englischen Bezeichnungen vorgegeben:



Programmgerüst: Verwenden und *ergänzen* Sie das beigefügte Programmgerüst. Beachten Sie dabei unbedingt die enthaltenen Kommentare! **Diese sind Teil der Aufgabenstellung** und damit bindend. Deklarationen und Signaturen dürfen *nicht geändert* werden. (D. h. z. B. darf eine Deklaration `private int x = 5;` nicht in `public int x;` geändert werden, sondern muss genau so stehenbleiben.) Es dürfen weitere Membervariablen, Methoden und Konstruktoren erstellt werden. Alle nötigen Klassen sind bereits vorhanden. Weitere Klassen dürfen Sie hinzufügen. Es sind u. a. folgende Elemente vorgegeben:

- Das Hauptprogramm befindet sich in der Klasse `Main`.
- Die Klasse `Game` ist für die Benutzerinteraktion gedacht. Sie enthält ein Attribut vom Typ `Position`, auf dem das Spiel gespielt werden kann.
- Objekte der Klasse `Position` repräsentieren eine Spielsituation.
- Die Klasse `IO` stellt Hilfsmethoden zum Einlesen von der Konsole bereit. Es ist somit nicht nötig, `MiniJava` zu verwenden. Stattdessen kann das Spielgeschehen auf der Konsole angezeigt und Eingaben von der Konsole eingelesen werden.
- Die Klasse `Globals` stellt String-Werte für die Ausgabe des Spielbretts auf der Konsole bereit. Ändern Sie den Wert von `Globals.ANSI` auf `false`, falls es zu Problemen mit der Darstellung der Unicode-Zeichen kommt. Abbildung 1 zeigt eine funktionierende Ausgabe der Unicode-Zeichen auf der Konsole.
- Die Klasse `Move` für Züge. Zur Eingabe soll die Form <Ausgangsfeld><Zielfeld> als String verwendet werden, also z. B. `"e2e4"` für einen Zug vom Feld `"e2"` zum Feld `"e4"`. Listen (z. B. von Zügen) werden generell als Arrays **ohne** leere Einträge (`null`) dargestellt. Für diese Aufgabe brauchen Sie also keine `Generics`.

  Hinweis: Sie dürfen die Methode `java.util.Arrays.copyOfRange` aus der Java-API verwenden, um ein Teilarray zu kopieren und so leere Einträge zu vermeiden.

Weitere Hinweise: Vermeiden Sie unnützen Code und Redundanzen nach Möglichkeit. Fügen Sie also beispielsweise nicht allen Vegetarier-Klassen das selbe Attribut hinzu, sondern deklarieren Sie es weiter oben in der Klassenhierarchie (z. B. in der Klasse `Vegetarian`). Die konkrete Art der Umsetzung fließt auch in die Bewertung ein. (Es wird also nicht nur Funktionalität bewertet.)

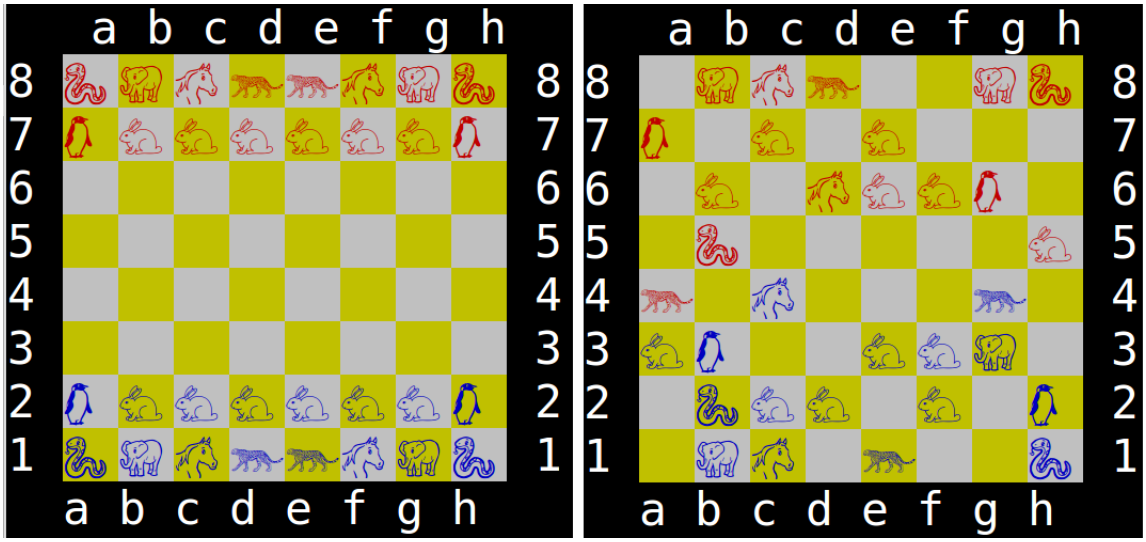


Abbildung 1: Startaufstellung (links) und Spielsituation in einer Linux-Konsole