Karlsruher Institut für Technologie

Prof. Dr.-Ing. Anne Koziolek
Institute of Information Security and Dependability (KASTEL)
https://sdq.kastel.kit.edu/wiki/Programmieren/
programmieren-vorlesung@cs.kit.edu

Programmieren – Wintersemester 2025/26

Abschlussaufgabe 1
Version 1.1

20 Punkte

Ausgabe:	16.02.2026, ca. 13:00 Uhr
Abgabestart:	02.03.2026, 12:00 Uhr
Abgabefrist:	17.03.2026, 06:00 Uhr

Änderungen v1.0 → v1.1

- Abbildung A.3 korrigiert (`k` statt `w`).
- Ausgabeformat von Der Befehl `move` angepasst
- Ausgabeformat von Der Befehl `place` für die sechste Einheit hinzugefügt
- Beispielinteraktion zu Der Befehl `yield` korrigiert (`"ERROR:"` statt `"Error, "`).
- Beispielinteraktion ergänzt: u.a. Präfixe bei der Spielbrettausgabe, Ausgabe des Spielbretts nach erfolgreichem Zug.
- Datei `board_boxes.txt` korrigiert und Dateiname im Text angepasst.
- Dateien `deck.txt`, `units.txt` und `random_numbers[...].txt` angepasst.

Plagiarismus

Es werden nur selbstständig angefertigte Lösungen akzeptiert. Das Einreichen fremder Lösungen, seien es auch nur teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen, ist ein Täuschungsversuch und führt jederzeit (auch nachträglich) zur Bewertung „nicht bestanden“. Ausdrücklich ausgenommen hiervon sind Quelltextschnipsel von den Vorlesungsfolien und aus den Lösungsvorschlägen des Übungsbetriebes in diesem Semester. Alle benutzten Hilfsmittel müssen vollständig und genau angegeben werden. Alles, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, muss deutlich kenntlich gemacht werden. Beachten Sie darüber hinaus die Richtlinien der Fakultät zum Verwenden von Generativer KI [1].

Studierende, die den ordnungsgemäßen Ablauf einer Erfolgskontrolle stören, können von der Erbringung der Erfolgskontrolle ausgeschlossen werden. Ebenso stellt unter anderem die Weitergabe von Teilen von Testfällen oder Lösungen bereits eine Störung des ordnungsgemäßen Ablaufs dar. Auch diese Art von Störungen können ausdrücklich jederzeit zum Ausschluss der Erfolgskontrolle führen. Dies bedeutet ausdrücklich, dass auch nachträglich die Punktzahl reduziert werden kann.

[1] https://www.informatik.kit.edu/faq-wiki/doku.php?id=generative_ki

Kommunikation und aktuelle Informationen

In unseren *FAQs*[2] finden Sie einen Überblick über häufig gestellte Fragen und die entsprechenden Antworten zum Modul „Programmieren“. Bitte lesen Sie diese sorgfältig durch, noch bevor Sie Fragen stellen, und überprüfen Sie diese regelmäßig und eigenverantwortlich auf Änderungen. Beachten Sie zudem die Hinweise im Wiki[3].

In den *ILIAS-Foren* oder auf *Artemis* veröffentlichen wir gelegentlich wichtige Neuigkeiten. Eventuelle Korrekturen von Aufgabenstellungen werden ebenso auf diesem Weg bekannt gemacht. Das aktive Beobachten der Foren wird daher vorausgesetzt.

Überprüfen Sie das Postfach Ihrer *KIT-Mailadresse* regelmäßig auf neue E-Mails. Sie erhalten unter anderem eine Zusammenfassung der Korrektur per E-Mail an diese Adresse. Alle Anmerkungen können Sie anschließend im Online-Einreichungssystem[4] einsehen.

Bearbeitungshinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen verpflichtender Tests für eine erfolgreiche Abgabe von Abschlussaufgabe 1 notwendig ist. Ihre Abgabe wird automatisch mit null Punkten bewertet, falls eine der nachfolgenden Regeln verletzt ist. Sie müssen außerdem zuerst die verpflichtenden Tests in Artemis bestehen, bevor die anderen Tests ausgewertet werden können. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

- Achten Sie auf fehlerfrei kompilierenden Programmcode.
- Verwenden Sie ausschließlich *Java SE 21*.
- Sofern in einer Aufgabe nicht ausdrücklich anders angegeben, verwenden Sie keine Elemente der Java-Bibliotheken. Ausgenommen ist die Klasse `java.util.Scanner` und alle Elemente aus den folgenden Paketen: `java.lang`, `java.io`, `java.util`, `java.util.regex`, `java.nio.file`, `java.nio.charset`.
- `System.exit()`, `Runtime.exit()` oder ähnliches dürfen nicht verwendet werden.
- Achten Sie darauf, den bereitgestellten Satz von Checkstyle-Regeln einzuhalten.

Beachten Sie desweiteren die allgemeinen Bewertungsrichtlinien für das Aufgabenblatt. Diese finden Sie unter `https://sdq.kastel.kit.edu/programmieren/Hauptseite`

[2] `https://sdq.kastel.kit.edu/wiki/Programmieren/FAQ`
[3] `https://sdq.kastel.kit.edu/programmieren/`
[4] `https://artemis.cs.kit.edu/`

Checkstyle

Das Online-Einreichungssystem überprüft Ihre Quelltexte während der Abgabe automatisiert auf die Einhaltung der Checkstyle-Regeln. Es gibt speziell markierte Regeln, bei denen das Online-Einreichungssystem die Abgabe mit null Punkten bewertet, da diese Regeln verpflichtend einzuhalten sind. Andere Regelverletzungen können zu Punktabzug führen. Sie können und sollten Ihre Quelltexte bereits während der Entwicklung auf die Regeleinhaltung überprüfen. Das Programmieren-Wiki beschreibt, wie Checkstyle verwendet werden kann.

Abgabehinweise

Die Abgabe im Online-Einreichungssystem wird am 02.03.2026, 12:00 Uhr, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Einreichungssystem bei der richtigen Aufgabe vor Ablauf der Abgabefrist am 17.03.2026, 06:00 Uhr, hochzuladen. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären. Falls Sie mit Git abgeben, *muss immer* auf den `main`-Branch gepusht werden.

- Geben Sie online Ihre `*.java`-Dateien zur Aufgabe A in Einzelarbeit mit der entsprechenden Ordnerstruktur im zugehörigen Verzeichnis ab.

Wiederverwendung von Lösungen

Falls Sie für die Bearbeitung der Abschlussaufgaben oder Übungsblätter Beispiellösungen aus diesem Semester wiederverwenden, *müssen* Sie in die entsprechenden Klassen "Programmieren-Team" ins Autor-Tag eintragen. Dies ist nötig, um die Checkstyle-Kriterien zu erfüllen.

Prüfungsmodus in Artemis

Wenn Sie mit einer Abschlussaufgabe fertig sind, können Sie diese frühzeitig abgeben. Dazu dient Schaltfläche „Vorzeitig abgeben“. Nach der frühzeitigen Abgabe einer Abschlussaufgabe können Sie keine Änderungen an Ihrer Abgabe mehr vornehmen.

Aufgabe A: Krone von Ackerland (`Crown of Farmland`)

In dieser Aufgabe implementieren Sie ein textbasiertes Spiel, in dem sich zwei Teams von Landwirten auf einem Spielfeld gegenseitig Schaden zufügen, um sich zu besiegen und die Vorherrschaft über ganz Ackerland (`Farmland`) zu erhalten.

A.1 Spielelemente

Zentrale Elemente der Modellierung dieser Aufgabe sind das Spielbrett, Einheiten und Bauernkönig, die Platzierung und Bewegung von Einheiten auf dem Spielfeld, Duelle zwischen Einheiten sowie die Hand und der Stapel.

A.1.1 Spielbrett

Die Bauernschlacht findet auf einem quadratischen *Spielbrett* statt, das aus 7×7 *Feldern* besteht. Auf jedem Feld darf sich zu jedem Zeitpunkt nur eine Einheit befinden. Die sieben Spalten und Zeilen des Spielbretts sind mit **A** bis **G** bzw. **1** bis **7** bezeichnet, sodass jedes Feld eindeutig durch die Kombination aus Spalte und Zeile identifizierbar ist. Feld **A1** ist das Feld in der *unteren linken* Ecke des Spielbretts.

A.1.2 Einheiten

Die Einheiten beider Teams sind die Akteure der Bauernschlacht. Sie können auf einem Feld platziert werden, auf dem Spielbrett bewegt werden und dabei Schaden anrichten.

Im Laufe des Zuges eines Teams dürfen sich seine Einheiten jeweils einmal bewegen, nämlich (standardmäßig) um *höchstens ein* Feld in jede der vier Richtungen entlang der Zeilen und Spalten (nicht diagonal). Eine Bewegung *en place*, d. h., auf dasselbe Feld, auf dem die Einheit bereits steht, ist zulässig und zählt als Bewegung.

Bewegt sich eine Einheit auf ein Feld, das von einer Einheit aus dem gegnerischen Team besetzt ist, wird ein *Duell* eingeleitet. Bewegt sich eine Einheit auf ein Feld, das von einer Einheit aus dem eigenen Team besetzt ist, wird ein *Zusammenschluss* eingeleitet. Eine Bewegung *en place* löst keinen Zusammenschluss aus.

Eine Einheit kann eine *Blockade* einleiten oder auflösen, sofern sie in diesem Zug noch nicht bewegt wurde. Die Blockade hält bis zur nächsten Bewegung der Einheit an. Das Einleiten oder Auflösen einer Blockade zählt als Bewegung.

Jede Einheit hat einen nichtnegativen *Angriffs-* und einen nichtnegativen *Verteidigungswert* (*ATK* bzw. *DEF*, kollektiv als *Statuswerte* bezeichnet), die über den Ausgang eines Duells entscheiden. Außerdem verfügt jede Einheit über einen zweiteiligen Namen, bestehend aus *Qualifikator* und *Rolle*, die in Programmausgaben stets durch ein Leerzeichen getrennt sind.

Wird eine Einheit platziert, ist sie zunächst *verdeckt*, d.h., das gegnerische Team kann Namen sowie Statuswerte der Einheit nicht einsehen. Eine Einheit kann von ihrem Team *aufgedeckt* werden,

sofern sie im selben Zug noch nicht bewegt wurde. Das Aufdecken einer Einheit zählt *nicht* als Bewegung. Aufgedeckte Einheiten können nicht wieder verdeckt werden. Wird ein Duell eingeleitet, werden beide kontrahierenden Einheiten aufgedeckt.

A.1.3 Bauernkönig

Der *Bauernkönig* ist eine besondere Einheit, die das jeweilige Team auf dem Spielbrett repräsentiert. Die Bauernkönige befinden sich zu Spielbeginn aufgedeckt auf Feld **D1** (Team 1) bzw. **D7** (Team 2). Bauernkönige können kein Duell initiieren. Bewegt sich ein Bauernkönig auf ein Feld, das durch eine eigene Einheit besetzt ist, wird diese aus dem Spiel genommen. Bauernkönige können sich nicht auf Felder bewegen, die von einer gegnerischen Einheit besetzt sind. Bauernkönige können keine Blockade einleiten und besitzen weder Angriffs- noch Verteidigungswert. Die beiden Bauernkönige verwenden in Programmausgaben als Namen `Farmer King`.

A.1.4 Duell

Ein Duell findet zwischen einer *angreifenden* Einheit und einer *verteidigenden* Einheit oder einem Bauernkönig statt. Dabei unterscheiden wir folgende Szenarien.

(ATK_A, DEF_A, ATK_B und DEF_B bezeichnen Angriffs- und Verteidigungswert der angreifenden Einheit A bzw. der verteidigenden Einheit B.)

Blockade Ist die verteidigende Einheit in Blockade, gewinnt die angreifende Einheit das Duell genau dann, wenn $ATK_A > DEF_B$. In diesem Fall wird die verteidigende Einheit aus dem Spiel genommen. Gilt umgekehrt $ATK_A < DEF_B$, nimmt das angreifende Team $DEF_B - ATK_A$ Schaden. Bei Gleichstand bleiben beide Einheiten und Teams unbeschadet. Gewinnt A das Duell nicht, verbleibt sie auf dem Feld, von dem aus sie angegriffen hat.

Angriff auf den Bauernkönig Wird ein Bauernkönig angegriffen, nimmt sein Team ATK_A Schaden. Die angreifende Einheit verbleibt auf dem Feld, von dem aus es angegriffen hat.

Standard Ist die verteidigende Einheit nicht in Blockade, gewinnt die Einheit mit dem größeren Angriffswert das Duell. Das Team der besiegten Einheit nimmt Schaden in Höhe der Differenz zwischen den Angriffswerten ($|ATK_A - ATK_B|$). Die besiegte Einheit wird aus dem Spiel genommen. Bei Gleichstand werden beide Einheiten aus dem Spiel genommen.

A.1.5 Hand

Die *Hand* jedes Teams enthält bis zu fünf eigene Einheiten. Zu Beginn der Bauernschlacht nehmen beide Teams jeweils vier Einheiten aus ihrem eigenen *Stapel* auf die *Hand*, und zu Beginn jedes eigenen Zuges eine weitere. Im Laufe des Zuges eines Teams kann höchstens einmal eine Einheit (oder mehrere Einheiten, siehe Zusammenschluss) aus seiner Hand auf einem der bis zu acht Felder, die an die Position des Bauernkönigs angrenzen (auch diagonal), platziert werden. Platziert ein Team eine Einheit, während bereits fünf eigene Einheiten auf dem Spielbrett platziert sind, und sind somit nun sechs eigene Einheiten auf dem Spielbrett, wird die soeben platzierte Einheit sofort aus dem Spiel genommen. Nur Einheiten aus der Hand können auf dem Spielbrett platziert werden.

Diese werden dann aus der Hand entfernt. Das Platzieren von Einheiten auf einem Feld, das durch eine gegnerische Einheit oder den gegnerischen Bauernkönig besetzt ist, ist unzulässig.

A.1.6 Stapel

Jedes Team besitzt einen eigenen Stapel von Einheiten. Dieser enthält zu Beginn der Bauernschlacht 40 Einheiten. Im Laufe des Spiels werden Einheiten vom Stapel *gezogen* und in die Hand aufgenommen. Es wird stets die oberste Einheit vom Stapel gezogen.

Hält das Team am Ende des Zuges fünf Einheiten auf der Hand, wählt es eine Einheit aus der Hand aus, wirft sie ab und nimmt sie aus dem Spiel. Zu Beginn jedes Zuges zieht das Team eine weitere Einheit vom Stapel. Ist der Stapel eines Teams leer und kann es demzufolge nicht nachziehen, verliert es die Bauernschlacht.

A.1.7 Lebenspunkte

Beide Teams verfügen zu Beginn des Spiels über 8000 *Lebenspunkte* (*LP*). Sinken die Lebenspunkte eines Teams auf 0, verliert es die Bauernschlacht.

A.1.8 Ende des Spiels

Hat ein Team das Spiel gewonnen, ist es zu Ende und das Programm wird beendet.

A.1.9 Zusammenschluss

Einheiten können sich *zusammenschließen*, sofern sie *kompatibel* sind, indem sich eine Einheit auf ein Feld bewegt, das von einer Einheit des gleichen Teams besetzt ist, oder eine Einheit aus der Hand auf einem solchen Feld platziert wird. Durch einen Zusammenschluss entstehen neue Einheiten, die sich nicht wieder trennen lassen. Zusammengeschlossene Einheiten zählen als *eine* Einheit. Wird ein Zusammenschluss zweier inkompatibler Einheiten eingeleitet, schlägt der Zusammenschluss fehl und die Einheit, die sich auf dem Zielfeld befunden hat, wird aus dem Spiel genommen.

Zusammengeschlossene Einheiten erhalten einen Namen, der die zusammengeschlossenen Einheiten wiedergibt. Dafür wird die Einheit A, die platziert oder bewegt wird, von der Einheit B, die sich auf dem Zielfeld befindet, unterschieden. Der Namen beider Einheiten, jeweils bestehend aus Qualifikator (*Daisy*, *Pitchfork*, *Milk Cow* etc.) und Rolle (*Farmer*, *Guard*, etc.), werden kombiniert: der Name der zusammengeschlossenen Einheit AB besteht aus dem Qualifikator von B, dem Qualifikator von A und der Rolle von B, die in Programmausgaben jeweils durch Leerzeichen getrennt werden.

Beispiele (unter Vernachlässigung von Kompatibilität)

- *Milk Cow Farmer* und *Barn Guard* ergeben *Barn Milk Cow Guard*.
- *Manure Spreader* und *Agro Architect* ergeben *Agro Manure Architect*.

Sofern A oder B verdeckt war, ist die zusammengeschlossene Einheit AB verdeckt. Die zusammengeschlossene Einheit AB kann sich im aktuellen Zug noch bewegen.

Zusammengeschlossene Einheiten können sich weiter zusammenschließen, wenn sie kompatibel zu anderen Einheiten sind.

A.1.10 Kompatibilität

Die Kompatibilität zweier Einheiten A und B richtet sich nach ihren Angriffs- und Verteidigungswerten. Zwei Einheiten mit demselben Namen können sich nicht zusammenschließen. Wir unterscheiden drei Szenarien, die auch Angriffs- und Verteidigungswert der zusammengeschlossenen Einheit beeinflussen. Die Kriterien für die drei Fälle sollen in der Reihenfolge geprüft werden, in der die Fälle hier angegeben sind.

Symbiose Seien A und B so gewählt, dass $ATK_A > ATK_B$ (*echt* größer). Gilt nun $ATK_A = DEF_B \wedge ATK_B = DEF_A$, so sind A und B *symbiotisch* kompatibel. Die zusammengeschlossene Einheit AB hat die Statuswerte $\begin{cases} ATK_{AB} = ATK_A \\ DEF_{AB} = DEF_B \end{cases}$.

Gleichgesinntheit Sei $g3t := max\{ggT(ATK_A, ATK_B), ggT(DEF_A, DEF_B)\}$ der größere größte gemeinsame Teiler der Statuswerte. Ist $g3t > 100$, sind A und B *konspirativ* kompatibel. Die zusammengeschlossene Einheit AB hat die Statuswerte $\begin{cases} ATK_{AB} = ATK_A + ATK_B - g3t \\ DEF_{AB} = DEF_A + DEF_B - g3t \end{cases}$.

Primkompatibilität Ist $g3t = 100$ und gilt außerdem, dass $ATK_A/100$ und $ATK_B/100$ beide prim sind oder $DEF_A/100$ und $DEF_B/100$ beide prim sind, sind A und B *prima* kompatibel. Die zusammengeschlossene Einheit AB hat die Statuswerte $\begin{cases} ATK_{AB} = ATK_A + ATK_B \\ DEF_{AB} = DEF_A + DEF_B \end{cases}$.

Trifft keines der drei Kriterien zu, sind die beiden Einheiten *inkompatibel*.

A.1.11 Verfügbare Einheiten und Stapelzusammensetzung

Die verfügbaren Einheiten werden beim Programmstart aus Eingabedateien eingelesen, ebenso die Zusammensetzung der Stapel für Team 1 und Team 2. Alles Weitere ist in Einlesen verfügbarer Einheiten und Einlesen der Stapelzusammensetzung nachzulesen.

A.1.12 Verfahren der gewichteten Zufallsauswahl

Im weiteren Verlauf der Aufgabe wird das Verfahren der gewichteten Zufallsauswahl zum Einsatz kommen. Basis hierfür ist eine endliche geordnete (nicht zwangsläufig sortierte) Liste von Ganzzahlen $(k_n) = (k_0, k_1, ..., k_{max})$, die das Gewicht der verschiedenen Optionen angeben. Negative Gewichte werden auf 0 gesetzt. Der n-ten Option wird anschließend das Intervall $I_n = [1 + \sum_{i=0}^{n-1} k_i, \sum_{i=0}^{n} k_i]$ zugeordnet. Anschließend wird eine Ganzzahl r zwischen 1 und $\sum_{i=0}^{max} k_i$ (inklusive) zufällig gewählt. Wir bestimmen das Intervall I_{n^*}, in dem r enthalten ist; die n^*-te Option wird ausgewählt.

Beispiel: Die geordnete Menge von Gewichten sei $(300, 400, 500)$. Das ergibt die Intervalle $[1, 300]$, $[301, 700]$ und $[701, 1200]$. Wir bestimmen zufällig ein r aus dem Intervall $[1, 1200]$. Für $r = 500$ wird demnach etwa die Option mit Gewicht 400 gewählt, für $r = 800$ die mit Gewicht 500.

A.1.13 Verfahren der umgekehrt gewichteten Zufallsauswahl

Hier entspricht ein größeres Gewicht einer kleineren Wahrscheinlichkeit dafür, dass die Option gewählt wird. Hierfür wird aus allen Gewichten $(k_0, k_1, ...)$ das Maximum k^* bestimmt und zur Gewichtung die neuen Gewichte $(k^*-k_0, k^*-k_1, ...)$ verwendet. Dann wird das gewöhnliche Verfahren der gewichteten Zufallsauswahl angewandt; die Reihenfolge der Optionen bleibt erhalten.
Hinweis: Die ursprünglich mit k^* gewichteten Optionen scheiden aus der Auswahl aus.

Beispiel: Für die ursprünglichen Gewichte $(300, 400, 500)$ werden die neuen Gewichte $(200, 100, 0)$ bestimmt. Die resultierenden Intervalle sind $[1, 200]$, $[201, 300]$, $[301, 300]$, wobei das letzte Intervall als leer angesehen wird.

A.2 KI-Gegner

Das gegnerische Team wird von einer einfachen Logik gesteuert. In jedem Zug werden folgende Schritte durchlaufen:

Bewegung des Bauernkönigs Die Felder, auf sich die der Bauernkönig bewegen kann (Bewegung *en place* inbegriffen), werden nach folgenden Kriterien bewertet: Entfernung (*distance*, 0 oder 1), Anzahl der das Feld in acht Richtungen umgebenden gegnerischen (*enemies*) und eigenen Einheiten (*fellows*, den Bauernkönig nicht mit eingeschlossen), sowie ob sich eine eigene eigene Einheit auf dem Feld befindet (*fellowPresent*, 0 oder 1, den Bauernkönig nicht mit eingeschlossen). Der Bauernkönig bewegt sich auf das Feld mit der größten Punktzahl $score = fellows - 2 * enemies - distance - 3 * fellowPresent$. Bei Gleichstand zwischen mehreren Feldern wird aus diesen durch das Verfahren der gewichteten Zufallsauswahl eines ausgewählt. Dabei betragen die Gewichte jeweils 1 und die Reihenfolge der Felder ist *oben*, *rechts*, *unten*, *links*, *en place*.

Einheit platzieren – Feldwahl Das Feld, auf das eine Einheit aus der Hand platziert werden soll, wird ermittelt, indem alle potenziellen Zielfelder nach folgenden Kriterien bewertet werden: Anzahl der nötigen Schritte, um den gegnerischen Bauernkönig zu erreichen (*steps*, Manhattan-Distanz), Anzahl der das Feld in vier Richtungen umgebenden gegnerischen (*enemies*) und eigenen Einheiten (*fellows*, den Bauernkönig mit eingeschlossen). Das Feld mit der höchsten Punktzahl $score = -steps + 2 * enemies - fellows$ wird gewählt. Bei Gleichstand zwischen mehreren Feldern wird aus diesen durch das Verfahren der gewichteten Zufallsauswahl eines ausgewählt. Dabei sind betragen die Gewichte jeweils 1 und die Reihenfolge der Felder ist, beginnend mit dem Feld oberhalb des Bauernkönigs, im Uhrzeigersinn um den Bauernkönig herum.

Nur dann, wenn kein freies Feld zur Verfügung steht, um eine Einheit zu platzieren, platziert das Team in diesem Zug keine Einheit und holt dies auch nicht später im Zug nach.

Feld platzieren – Wahl der Einheit Die zu platzierende Einheit wird durch das Verfahren der gewichteten Zufallsauswahl ermittelt. Die Angriffswerte der Einheiten auf der Hand bilden die Gewichte, die Reihenfolge entspricht der auf der Hand. Die ermittelte Einheit wird auf das ermittelte Feld platziert.

Bewegung der Einheiten Für jede Einheit A erhalten das Feld oberhalb, rechts, unterhalb und links der Einheit eine Punktzahl mit folgenden Komponenten:

- Möglicher Zusammenschluss: Ist das Feld durch eine eigene Einheit B besetzt und ist ein Zusammenschluss zur Einheit AB möglich, ergibt das eine Punktzahl von $ATK_{AB} + DEF_{AB} - ATK_A - DEF_A$.
- Elimination: Ist das Feld durch eine eigene Einheit B besetzt und ist ein Zusammenschluss nicht möglich, ergibt das eine Punktzahl von $-ATK_B - DEF_B$.
- Gegner angreifen: Ist das Feld durch eine gegnerische Einheit B besetzt, ergibt das eine Punktzahl von ATK_A, falls B der gegnerische Bauernkönig ist, $ATK_A - 500$, falls B verdeckt ist, und andernfalls $ATK_A - DEF_B$, falls B blockiert, und andernfalls $2 * (ATK_A - ATK_B)$.
- Vorrücken: Ist das Feld unbesetzt, ergibt das eine Punktzahl von $10 * steps - enemies$, wobei $steps$ wieder die Anzahl der nötigen Schritte bezeichnet, um den gegnerischen Bauernkönig zu erreichen (Manhattan-Distanz), und $enemies$ die Anzahl der das Feld in vier Richtungen umgebenden gegnerischen Einheiten.

Zusätzlich erhält das Blockieren eine Punktzahl in Höhe von $max(1, \frac{DEF_A - ATK^*_B}{100})$, wobei ATK^*_B dem größten Angriffswert unter allen gegnerischen Einheiten B entspricht, die A in gerader Linie umgeben. Die Bewegung *en place* erhält die Punktzahl $max(0, \frac{ATK_A - ATK^*_B}{100})$.

Die Punktzahlen der möglichen Bewegungen werden anschließend aufsummiert. Die Einheit mit der höchsten Gesamtpunktzahl kommt zum Zug; der auszuführende Zug wird durch das Verfahren der gewichteten Zufallsauswahl bestimmt. Die Punktzahlen bilden die Gewichte; die Reihenfolge der Optionen entspricht der, in der die sie hier aufgeführt sind (oben, rechts, unten, links, blockieren, Bewegung *en place*). Steht keine Bewegung mit positiver Punktzahl zur Auswahl, wird eine Blockade eingeleitet. Bei Gleichstand der Gesamtpunktzahl mehrerer Einheiten wird wiederum die Einheit per Verfahren der gewichteten Zufallsauswahl gewählt, die ziehen darf. Die Gewichte betragen jeweils 1, und die Reihenfolge entspricht derjenigen, in der die Einheiten platziert wurden. Sich zusammenschließende Einheiten werden als neu platziert gezählt.

Anschließend beginnt das Verfahren für alle Einheiten, die weiterhin bewegt werden können, von vorn.

Ende des Zugs Nachdem alle Einheiten bewegt worden sind, beendet der KI-Gegner seinen Zug. Falls er eine Einheit abwerfen muss, wird diese im Verfahren der umgekehrt gewichteten Zufallsauswahl bestimmt, wobei die Gewichte der Summe aus ATK und DEF der Einheit entsprechen und die Reihenfolge der der Einheiten auf der Hand entspricht.

Schlüssel	Typ	Beispielwert	Beschreibung
`seed`	Ganzzahl	`123456`	Startwert für den Zufallsgenerator (siehe A.4)
`board`	Pfad	`board.txt`	Spielbrettsymbole-Datei (siehe A.5.2)
`units`	Pfad	`units.txt`	Verfügbare Einheiten (siehe A.1.11)
`deck`	Pfad	`deck.txt`	Stapelzusammensetzung, beide Teams (siehe A.1.11)
`deck1`	Pfad	`deck1.txt`	Stapelzusammensetzung, Team 1 (siehe A.1.11)
`deck2`	Pfad	`deck2.txt`	Stapelzusammensetzung, Team 2 (siehe A.1.11)
`team1`	Zeichenkette	`Player`	Teamname, Team 1 (bis 14 Zeichen)
`team2`	Zeichenkette	`Enemy`	Teamname, Team 2 (bis 14 Zeichen)
`verbosity`	Zeichenkette	`compact`	`all`: Gesamte Ausgabe; `compact`: kompakte Ausgabe.

Tabelle A.1: Gültige Schlüssel-Wort-Paare beim Programmaufruf.

A.3 Programmstart

Eingabeformat: `java -jar <Main> [<key>=<value>...]`

Initialisiert ein Bauernduell unter Verwendung der eingelesenen Schlüssel-Wert-Paare. Gültige Schlüssel-Wert-Paare sind in Tabelle A.1 angegeben. Die Schlüssel-Wert-Paare können in beliebiger Reihenfolge angegeben werden; sie werden jedoch stets in der Reihenfolge verarbeitet, wie sie in Tabelle A.1 angegeben sind. Die Inhalte der übergebenen Dateien werden zunächst unverändert ausgegeben, bevor sie verarbeitet werden und ggf. eine Fehlermeldung ausgegeben wird.

`units` sowie `seed` müssen angegeben werden. Es muss entweder `deck` oder `deck1` und `deck2` verwendet werden (aber keine Kombination von beiden). `verbosity` ist optional, es sind aber nur die beiden angegebenen Argumente zulässig. Kein Schlüssel darf mehr als einmal angegeben werden. Wird `team1` nicht angegeben, wird die Zeichenkette `"Player"` verwendet, analog `"Enemy"` für `team2`. Wird ein angegebener Dateipfad nicht gefunden, hat eine Datei einen unerwarteten Inhalt, wurde eine ungültige Ganzzahl angegeben oder ein sonstiges fehlerhaftes Argument gefunden, wird das Programm mit einer Fehlermeldung beendet, ohne dass weitere Argumente verarbeitet werden.

Nun werden die erwarteten Dateiformate beschrieben.

A.3.1 Einlesen des angepassten Symbolsatzes

Zur angepassten Darstellung des Spielbretts werden 29 Zeichen benötigt. Diese werden in einer einzigen Zeile und ohne jegliche Trennzeichen erwartet; Abbildung A.1 zeigt ein Beispiel. Sie finden in den Unterlagen zur Aufgabe eine weitere solche Datei `board_boxes.txt`, die sich optisch wesentlich besser zur Darstellung eignet als Buchstaben. Ihre Implementierung sollte mit UTF-8-kodierten Eingabedateien wie `board_boxes.txt` umgehen können. In Der Befehl `board` finden Sie Beispiele für Ausgaben des Spielbretts unter Verwendung verschiedener Symbolsätze.

A.3.2 Einlesen verfügbarer Einheiten

Die Datei zur Definition der verfügbaren Einheiten enthält bis zu 80 verschiedene Einheiten in je einer Zeile. Angegeben sind für jede Einheit ihr Name (bestehend aus den beiden Zeichenketten

Eingabe

`board_abc.txt`

```
1 abcdefghijklmnopqrstuvwxyzäöü
```

Abbildung A.1: Beispiel für eine Datei, die einen angepassten Symbolsatz definiert.

Qualifikator und *Rolle*), ihre Angriffspunkte und ihre Verteidigungspunkte, die jeweils durch Semikolons getrennt sind. Die Zeilen sollen ohne Semikolon und ohne weitere Leerzeichen enden. Abbildung A.2a zeigt ein Beispiel.

In Programmausgaben sollen Qualifikator und Rolle durch ein Leerzeichen getrennt sein. Sie dürfen davon ausgehen, dass die Namen der Einheiten keine Semikolons enthalten werden. Enthält eine Zeile falsche Datentypen oder nicht die richtige Anzahl von Angaben, schlägt der Programmstart fehl.

A.3.3 Einlesen der Stapelzusammensetzung

Die Zusammensetzung der Stapel wird entweder für beide Teams übereinstimmend aus derselben Datei (Argument `deck`) oder mit verschiedenen Zusammensetzungen beider Stapel aus zwei Dateien (Argumente `deck1` und `deck2`) ausgelesen.

In diesen Dateien wird für jede eingelesene Einheit eine nichtnegative Zahl in je einer eigenen Zeile erwartet, die die Anzahl der jeweiligen Einheiten im Stapel angibt; Abbildung A.2b zeigt ein Beispiel. Wird für eine Einheit die Anzahl 0 angegeben, wird sie im entsprechenden Stapel nicht verwendet. Unterscheidet sich die Anzahl der Zeilen von der der definierten Einheiten, oder ist die Gesamtanzahl der Einheiten in einem Stapel ungleich 40, schlägt der Programmstart fehl.

A.3.4 Initialisierung des Spielbretts

Nach dem Einlesen der Eingabedateien wird der Stapel für beide Teams befüllt, nämlich in der Reihenfolge der Deklaration verfügbarer Einheiten und in der Anzahl entsprechend der Deklaration der Stapelzusammensetzung. Die erste deklarierte Einheit, die wenigstens einmal im Stapel vorkommen soll, liegt demnach im Stapel zuoberst (d.h., bei Index 0). Anschließend wird zunächst der Stapel von Team 1, dann der Stapel von Team 2 zufällig gemischt, je vier Einheiten nacheinander oben vom Stapel in die Hand gezogen und die Bauernkönige platziert. Dann beginnt der erste Zug von Team 1.

A.4 Zufallsgenerator

Der Zufallsgenerator soll an verschiedenen Stellen in Ihrem Programm eingesetzt werden, um das Programmverhalten (pseudo-)indeterministisch zu machen. Um das Programm dennoch auf korrektes Verhalten testen zu können, muss der Zufallsgenerator korrekt initialisiert werden und es müssen zufällige Auswahlen immer in derselben Reihenfolge und durch Aufruf derselben Methode getroffen werden wie hier spezifiziert.

Eingabe

units.txt

```
1  Daisy;Farmer;300;500
2  Pitchfork;Farmer;400;400
3  Stable;Farmer;500;300
4  Manure;Spreader;600;600
5  Potato;Farmer;700;500
6  Fence;Farmer;300;1000
7  Barn;Guard;400;900
8  Chicken;Farmer;800;700
9  Tractor;Farmer;900;600
10 Silo;Operator;500;1100
11 Beetroot;Farmer;1000;900
12 Goat;Farmer;1000;1000
13 Pig;Farmer;1200;800
14 Grain;Farmer;1100;1100
15 Threshing;Maid;1300;1000
16 Shield;Farmer;800;1600
17 Barrier;Builder;900;1700
18 Milk Cow;Farmer;1400;1400
19 Livestock;Sorceress;1500;1200
20 Field Marshal;Farmer;1700;1500
21 Agro;Architect;1900;1800
22 Titan Tractor;Farmer;2200;2000
23 Seed;Farmer;2500;2300
24 Egg-laying Wool-Milk-Pig;Farmer;3000;3000
```

(a) Beispiel für eine Datei, die verfügbare Einheiten definiert.

Eingabe

deck.txt

```
1  3
2  2
3  3
4  2
5  1
6  2
7  2
8  1
9  3
10 1
11 2
12 1
13 1
14 2
15 1
16 2
17 1
18 1
19 1
20 2
21 2
22 2
23 1
24 1
```

(b) Beispiel für eine Datei, die eine Stapelzusammensetzung definiert.

Abbildung A.2: Beispiele für Konfigurationsdateien.

Sie finden in den Unterlagen eine Datei, die die Ausgaben des Zufallsgenerators für die Beispielinteraktion für einen bestimmten Startwert enthält. Ihr Programm sollte für diesen Startwert in derselben Reihenfolge und dem gleichen Ergebnis dieselben Anfragen an den Zufallsgenerator stellen.

Initialisierung Verwenden Sie ein globales Objekt der Klasse `java.util.Random` als Zufallsgenerator. Initialisieren Sie es mit dem Startwert `seed`, der beim Programmstart übergeben wird.

Stapel mischen Verwenden Sie die statische Methode `Collections.shuffle(List<?> list, Random rnd)`, um Stapel zu mischen.

Ganzzahl ziehen Verwenden Sie die Methode `nextInt(int origin, int bound)` Ihres `Random`-Objekts, um eine Ganzzahl zwischen `origin` (inbegriffen) und `bound` (**nicht** inbegriffen) zu ziehen.

A.5 Befehle

Nach dem Start wird zuerst ein Hilfetext (in einer Zeile) ausgegeben:

`"Use one of the following commands: select, board, move, flip, block, hand, place, show, yield, state, quit."`.

Danach nimmt Ihr Programm über die Konsole die Befehle unter Verwendung der Standardeingabe entgegen, die im Folgenden näher beschrieben werden. Alle Befehle werden auf dem aktuellen Programmzustand ausgeführt. Nach Abarbeitung einer Eingabe wartet Ihr Programm auf weitere Eingaben, bis es durch den Befehl `quit` beendet wird. Die Groß- und Kleinschreibung von Befehlen und deren Argumenten wird nicht beachtet.

Achten Sie darauf, dass durch die Ausführung der folgenden Befehle die gegebene Spezifikation nicht verletzt wird, und geben Sie in diesen Fällen immer eine aussagekräftige Fehlermeldung aus. Wenn die Benutzereingabe nicht dem vorgegebenen Format entspricht oder ein Befehl nicht erfolgreich ausgeführt werden kann, ist auch eine Fehlermeldung auszugeben. Nach der Ausgabe einer Fehlermeldung soll das Programm wie erwartet fortfahren und wieder auf die nächste Eingabe warten. Jede Fehlermeldung muss mit der Zeichenkette `"ERROR:"` gefolgt von einem Leerzeichen beginnen und darf keine Zeilenumbrüche enthalten. Den weiteren Text der Fehlermeldung dürfen Sie frei wählen; er sollte jedoch sinnvoll sein.

Beachten Sie, dass in der Beschreibung der Eingabe- und Ausgabeformate die Wörter zwischen spitzen Klammen (`<` und `>`) als Platzhalter dienen, die bei der konkreten Ein- und Ausgabe durch Werte ersetzt werden. Diese eigentlichen Werte enthalten bei der Ein- und Ausgabe keine spitzen Klammern. Auf dieselbe Weise werden die eckigen Klammern (`[` und `]`) für optionale Platzhalter verwendet, die bei Befehlen angegeben werden können, aber nicht erforderlich sind.

- **`<atk>`:** Angriffswert der betroffenen Einheit.
- **`<bc>`:** Die Anzahl der Einheiten des betroffenen Teams auf dem Spielbrett (*board count*), den Bauernkönig nicht inbegriffen.

- **`<board>`:** Die Ausgabe des Befehls `board`.
- **`<damage>`:** Der aus einem Duell resultierende Schaden an den Lebenspunkten des angreifenden oder verteidigenden Teams
- **`<dc>`:** Die Größe des Stapels (*deck count*) des betroffenen Teams.
- **`<def>`:** Verteidigungswert der betroffenen Einheit.
- **`<field>`:** Ein Feldbezeichner der Form `A1` bis `G7`.
- **`<idx>`:** 1-basierter Index der Einheiten auf der Hand.
- **`<lp>`:** Die verbleibenden Lebenspunkte des betroffenen Teams.
- **`<name>`:** Der Name der betroffenen Einheit oder `???`, falls die Einheit verdeckt ist und dem Team angehört, das nicht am Zug ist.
- **`<padding>`:** Eine Kette aus Leerzeichen, um einen Abstand herzustellen.
- **`<show>`:** Die Ausgabe des Befehls `show`.
- **`<team>`:** Der Name des betroffenen Teams.

A.5.1 Der Befehl `select`

Eingabeformat: `select <field>`

Ausgabeformat:

Wählt ein Feld des Spielbretts aus. Anschließend werden das Spielbrett und die aktuelle Auswahl ausgegeben, wie in Der Befehl `board` bzw. Der Befehl `show` beschrieben.

Die Auswahl eines Feldes ist eine Voraussetzung für verschiedene andere Befehle. Die vorige Auswahl wird gegebenenfalls aufgehoben. Im Folgenden wird mit „die *ausgewählte Einheit*“ die Einheit bezeichnet, die sich auf dem ausgewählten Feld befindet. Wird eine ausgewählte Einheit vorausgesetzt, ist aber kein Feld ausgewählt oder das ausgewählte Feld leer, schlägt der Befehl fehl.

A.5.2 Der Befehl `board`

Eingabeformat: `board`

Ausgabeformat: Gibt eine Darstellung des Spielfelds auf der Konsole aus. Dabei wird das Spielfeld durch die Spielbrettsymbole und die Einheiten mittels Buchstabenkombinationen repräsentiert.

- Eigenes Team: `'x'` bzw. `'X'`
- Gegnerisches Team: `'y'` bzw. `'Y'`
- Bauernkönig: `'X'` bzw. `'Y'`
- Andere Einheiten: `'x'` bzw. `'y'`
- Blockade: Suffix `'b'`

- Einheit kann in diesem Zug noch bewegt werden: Präfix `'*'`

Das Spiel unterstützt zwei Darstellungen von Spielbrettern, die jeweils verschiedene Spielbrettsymbole verwenden: den *Standardsymbolsatz* und den *angepassten Symbolsatz*. Wird beim Programmaufruf eine Symbolsatzdatei angegeben, wird der angepasste Symbolsatz verwendet, andernfalls der Standardsymbolsatz. Am Zeilenende dürfen keine überzähligen Leerzeichen vorhanden sein.

Standardsymbolsatz Der Standardsymbolsatz verwendet folgende Symbole:

- ein Eckverbindungssymbol `'+'`,
- ein horizontales Verbindungssymbol `'-'` und
- ein vertikales Verbindungssymbol `'|'`.

Für diese drei Symbole gibt es jeweils eine Variante für das *ausgewählte Feld*: `'='`, `'N'` und `'#'`.

```
  +---+---+---+---+---+---+---+
7 |   |   |   | Y |   |   |   |
  +---+---+---+---+---+---+---+
6 |   |   | y | yb| yb|   |   |
  +---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |
  +---+---#===#---+---+---+---+
4 |   |   N   N   |   |   |   |
  +---+---#===#---+---+---+---+
3 |   |   |   |   |*x |   |   |
  +---+---+---+---+---+---+---+
2 |   |   |*x |   |   |   |   |
  +---+---+---+---+---+---+---+
1 |   |   |   |*X |   |   |   |
  +---+---+---+---+---+---+---+
    A   B   C   D   E   F   G
```

Angepasste Symbolsätze Angepasste Symbolsätze sind nicht standardisiert, sondern werden beim Programmstart optional aus einer Datei eingelesen. Sie unterscheiden einen erweiterten Satz von genau 29 Symbolen. Insbesondere liefert ein angepasster Symbolsatz unterschiedliche Varianten für einige Eckverbindungssymbole, je nachdem, welches angrenzende Feld ausgewählt ist.

Im Folgenden wird ein angepasster Symbolsatz anhand der Kleinbuchstaben `a` bis `z` sowie `ä`, `ö` und `ü` dargestellt; dies soll zum einfachen Testen dienen. Zusätzlich werden hier zur besseren Leserlichkeit diejenigen Symbole fett dargestellt, die zur Darstellung ausgewählter Felder benutzt werden, was in Ihrer Programmausgabe nicht erfolgen soll. Beachten Sie, dass die Darstellung der Einheiten durch angepasste Symbolsätze nicht beeinflusst wird.

Wir unterscheiden

- je ein Eckverbindungssymbol

  - *oben links* `'a'` (`'l'`), *oben rechts* `'b'` (`'m'`), *unten links* `'c'` (`'n'`), *unten rechts* `'d'` (`'o'`),
  - *oben mittig* `'e'` (ausg. Feld links: `'p'`, ausg. Feld rechts: `'q'`),
  - *rechts mittig* `'f'` (ausg. Feld oben: `'r'`, ausg. Feld unten: `'s'`),
  - *unten mittig* `'g'` (ausg. Feld links: `'t'`, ausg. Feld rechts: `'u'`),
  - *links mittig* `'h'` (ausg. Feld oben: `'v'`, ausg. Feld unten: `'w'`).
- ein horizontales Verbindungssymbol `'i'` (`'x'`),
- ein vertikales Verbindungssymbol `'j'` (`'y'`),
- ein *zentrales* Eckverbindungssymbol `'k'` (ausg. Feld oben links: `'z'`, ausg. Feld oben rechts: `'ä'`, ausg. Feld unten links: `'ö'`, ausg. Feld unten rechts: `'ü'`).

Beim Einlesen eines angepassten Symbolsatzes werden die 29 Zeichen in exakt der alphabetischen Reihenfolge in einer einzigen Zeile erwartet.

Ausgabemodus Beim Programmstart können die Ausgabemodi `all` oder `compact` bestimmt werden; ohne Angabe wird der Ausgabemodus `all` verwendet. In diesem Modus werden die Spielbretter dargestellt wie oben; im Modus `compact` werden diejenigen Zeilen ausgelassen, die nicht die Felder selbst, sondern ausschließlich Verbindungssymbole enthalten. In Abbildung A.3f ist eine kompakte Beispielausgabe unter Verwendung des Standardsymbolsatzes abgebildet, die dem Spielbrettzustand der Ausgabe auf Seite 15 entspricht.

A.5.3 Der Befehl `move`

Eingabeformat: `move <field>`

Ausgabeformat (Bewegung):

```
[<name> no longer blocks.]
<name> moves to <field>.
```

Ausgabeformat (Duell):

```
[<name> no longer blocks.]
<name> (<atk>/<def>) attacks <name> [(<atk>/<def>)] on <field>!
[<name> (<atk>/<def>) was flipped on <field>!]
[<name> (<atk>/<def>) was flipped on <field>!]
[<name> was eliminated!]
[<name> was eliminated!]
[<team> takes <damage> damage!]
[<name> moves to <field>.]
[<team>'s life points dropped to 0!]
[<team> wins!]
```

Ausgabeformat (Erfolgreicher Zusammenschluss):

```
  aiiieiiieiiieiiieiiieiiieiiib
7 j   j   j   j Y j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
6 j   j   j ybj ybj ybj   j   j
  hiiikiiikiiikiiikiiikiiikiiif
5 j   j   j   j   j   j   j   j
  hiiikiii üxxxö iiikiiikiiikiiif
4 j   j   y   y   j   j   j   j
  hiiikiii äxxxz iiikiiikiiikiiif
3 j   j   j   j   j*x j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
2 j   j   j*x j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
1 j   j   j   j*X j   j   j   j
  ciiigiiigiiigiiigiiigiiigiiid
    A   B   C   D   E   F   G
```

(a) Spielbrettdarstellung mit ausgewähltem Feld **C4**.

```
  lxxxpiiieiiieiiieiiieiiieiiib
7 y   y   j   j Y j   j   j   j
  vxxxziiikiiikiiikiiikiiikiiif
6 j   j   j ybj ybj ybj   j   j
  hiiikiiikiiikiiikiiikiiikiiif
5 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
4 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
3 j   j   j   j   j*x j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
2 j   j   j*x j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
1 j   j   j   j*X j   j   j   j
  ciiigiiigiiigiiigiiigiiigiiid
    A   B   C   D   E   F   G
```

(b) Spielbrettdarstellung mit ausgewähltem Feld **A7**.

```
  aiiieiiieiiieiiieiiieiiiqxxxm
7 j   j   j   j Y j   j   y   y
  hiiikiiikiiikiiikiiikiii äxxxr
6 j   j   j ybj ybj ybj   j   j
  hiiikiiikiiikiiikiiikiiikiiif
5 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
4 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
3 j   j   j   j   j*x j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
2 j   j   j*x j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
1 j   j   j   j*X j   j   j   j
  ciiigiiigiiigiiigiiigiiigiiid
    A   B   C   D   E   F   G
```

(c) Spielbrettdarstellung mit ausgewähltem Feld **G7**.

```
  aiiieiiieiiieiiieiiieiiieiiib
7 j   j   j   j Y j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
6 j   j   j ybj ybj ybj   j   j
  hiiikiiikiiikiiikiiikiiikiiif
5 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
4 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
3 j   j   j   j   j*x j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
2 j   j   j*x j   j   j   j   j
  wxxxöiiikiiikiiikiiikiiikiiif
1 y   y   j   j*X j   j   j   j
  nxxxtiiigiiigiiigiiigiiigiiid
    A   B   C   D   E   F   G
```

(d) Spielbrettdarstellung mit ausgewähltem Feld **A1**.

```
  aiiieiiieiiieiiieiiieiiieiiib
7 j   j   j   j Y j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
6 j   j   j ybj ybj ybj   j   j
  hiiikiiikiiikiiikiiikiiikiiif
5 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
4 j   j   j   j   j   j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
3 j   j   j   j   j*x j   j   j
  hiiikiiikiiikiiikiiikiiikiiif
2 j   j   j*x j   j   j   j   j
  hiiikiiikiiikiiikiiikiiiüxxxs
1 j   j   j   j*X j   j   y   y
  ciiigiiigiiigiiigiiigiiiuxxxo
    A   B   C   D   E   F   G
```

(e) Spielbrettdarstellung mit ausgewähltem Feld **G1**.

```
7 |   |   |   | Y |   |   |   |
6 |   |   | y | yb| yb|   |   |
5 |   |   |   |   |   |   |   |
4 |   |   N   N   |   |   |   |
3 |   |   |   |   |*x |   |   |
2 |   |   |*x |   |   |   |   |
1 |   |   |   |*X |   |   |   |
    A   B   C   D   E   F   G
```

(f) Kompakte Spielbrettdarstellung mit ausgewähltem Feld **C4**.

Abbildung A.3: Beispiele für Spielbrettabbildungen.

```
[<name> no longer blocks.]
<name> moves to <field>.
<name> and <name> on <field> join forces!
Success!
```

Ausgabeformat (Fehlgeschlagener Zusammenschluss):

```
[<name> no longer blocks.]
<name> moves to <field>.
<name> and <name> on <field> join forces!
Union failed. <name> was eliminated.
```

Die *ausgewählte Einheit* bewegt sich auf das angegebene Feld zu, wodurch unter Umständen ein Zusammenschluss oder ein Duell ausgelöst wird. Gegebenenfalls wird die Blockade der Einheit beendet. Falls ein Duell initiiert wird, wird gegebenenfalls zuerst die sich bewegende Einheit und dann die angegriffene Einheit aufgedeckt. Die Bewegung ist unzulässig, wenn

- das Zielfeld entlang der Zeilen und Spalten (nicht diagonal) mehr als ein Feld entfernt ist,
- eine Einheit sich auf das Feld des eigenen Bauernkönigs bewegt, oder
- ein Bauernkönig sich auf das Feld einer gegnerischen Einheit bewegt.

Zulässig ist hingegen eine Bewegung *en place* auf dasselbe Feld, auf dem die ausgewählte Einheit bereits steht.

Beachten Sie die in Duell definierten Fälle, in denen die sich bewegende Einheit auf dem Feld verbleibt, von dem aus es sich bewegt hat, oder eliminiert wird.

In allen anderen zulässigen Fällen wird die Bewegung vollzogen, (die Einheit ggf. zusammengeschlossen) und das angegebene Feld ausgewählt.

Eine zulässige Bewegung abschließend, werden das Spielbrett und die aktuelle Auswahl ausgegeben, wie in Der Befehl `board` bzw. Der Befehl `show` beschrieben.

Ist die Bewegung unzulässig, wird eine Fehlermeldung ausgegeben und die Einheit und das *ausgewählte Feld* bleiben unverändert.

Hinweise zur Ausgabe:

- Bei Unentschieden wird kein Schaden in Höhe von 0 ausgegeben.
- Werden bei Unentschieden beide betroffenen Einheiten eliminiert, wird zuerst die Elimination der angegriffenen, dann der sich bewegenden Einheit ausgegeben.
- Auch wenn entstehender Schaden die verbliebenen Lebenspunkte übersteigt, wird der berechnete Schaden in voller Höhe ausgegeben und nicht nur in Höhe der verbleibenden Lebenspunkte.
- Bewegungen *en place* werden ausgegeben wie unter *Ausgabeformat (Bewegung)* beschrieben.

A.5.4 Der Befehl `flip`

Eingabeformat: `flip`

Ausgabeformat: `<name> (<atk>/<def>) was flipped on <field>!`

Deckt die *ausgewählte Einheit* auf. Abschließend werden das Spielbrett und die aktuelle Auswahl ausgegeben, wie in Der Befehl `board` bzw. Der Befehl `show` beschrieben.

Das Aufdecken zählt nicht als Bewegung. Das Aufdecken ist unzulässig, wenn

- die ausgewählte Einheit in diesem Zug bereits bewegt worden ist oder
- sie bereits aufgedeckt ist.

Ist das Aufdecken unzulässig, wird eine Fehlermeldung ausgegeben und die Einheit bleibt unverändert.

A.5.5 Der Befehl `block`

Eingabeformat: `block`

Ausgabeformat: `<name> (<field>) blocks!`

Die *ausgewählte Einheit* leitet eine Blockade ein. Anschließend werden das Spielbrett und die aktuelle Auswahl ausgegeben, wie in Der Befehl `board` bzw. Der Befehl `show` beschrieben.

Eine Blockade einzuleiten, gilt als Bewegung. Die Blockade ist unzulässig, wenn die ausgewählte Einheit in diesem Zug bereits bewegt worden ist. Zulässig ist der Befehl hingegen, wenn die ausgewählte Einheit bereits blockiert und die vorhandene Blockade in einem früheren Zug eingeleitet worden ist. Ist die Blockade unzulässig, wird eine Fehlermeldung ausgegeben und die Einheit bleibt unverändert.

A.5.6 Der Befehl `show`

Eingabeformat: `show`

Zeigt Informationen zur *ausgewählten Einheit* an. Ist das ausgewählte Feld leer, wird dies ebenfalls ausgegeben.

Ausgabeformat (Leeres Feld):

```
<no unit>
```

Ausgabeformat (Einheit):

```
<name> (Team <team>)
ATK: <atk>
DEF: <def>
```

Gehört die Einheit dem Team an, das nicht am Zug ist, und ist die Einheit verdeckt, so wird anstelle des Namens, ATK und DEF jeweils `???` angegeben. Das betrifft beide Teams.

```
??? (Team <team>)
ATK: ???
DEF: ???
```

Ausgabeformat (Bauernkönig):

```
<team>'s Farmer King
```

A.5.7 Der Befehl `hand`

Eingabeformat: `hand`

Ausgabeformat:

```
[1] <name> (<atk>/<def>)
[2] <name> (<atk>/<def>)
...
```

Gibt eine nummerierte 1-basierte Liste der Einheiten auf der Hand des Teams aus, das am Zug ist.

A.5.8 Der Befehl `place`

Eingabeformat: `place <idx> [<idx> [<idx ...]]`

Ausgabeformat (kein Zusammenschluss) `<team> places <name> on <field>.`

Ausgabeformat (Erfolgreicher Zusammenschluss)

```
<team> places <name> on <field>.
<name> and <name> on <field> join forces!
Success!
```

Ausgabeformat (Erfolgloser Zusammenschluss)

```
<team> places <name> on <field>.
<name> and <name> on <field> join forces!
Union failed. <name> was eliminated.
```

Ausgabeformat (Sechste Einheit)

```
<team> places <name> on <field>.
<name> was eliminated!
```

Platziert eine oder mehrere Einheiten auf dem *ausgewählten Feld.* Anschließend werden das Spielbrett und die aktuelle Auswahl ausgegeben, wie in Der Befehl `board` bzw. Der Befehl `show` beschrieben.

Wird mehr als ein Index `idx` angegeben, bezieht sich jeder Index auf den Zustand der Hand vor dem Ablegen. Die genannten Einheiten aus der Hand werden in der Reihenfolge, in der ihre Indizes angegeben wurden, auf das Feld platziert. Dabei werden ggf. Zusammenschlüsse ausgelöst (auch mit der sich ggf. bereits auf dem genannten Feld befindenden Einheit). Das Platzieren ist unzulässig, wenn

- das Team in diesem Zug bereits Einheiten platziert hat,
- das Zielfeld entlang der Reihen, Spalten und Diagonalen mehr als ein Feld vom eigenen Bauernkönig entfernt ist,
- ein Index keiner Einheit auf der Hand zugeordnet werden kann,

- ein Index in einem Befehlsaufruf mehrfach aufgeführt ist oder
- das Zielfeld durch eine gegnerische Einheit oder den gegnerischen Bauernkönig besetzt ist.

Ist das Platzieren unzulässig, bleiben das Feld und die Hand unverändert und das Platzieren wird nicht gewertet; es kann also ggf. erneut eine Einheit oder mehrere Einheiten platziert werden.

A.5.9 Der Befehl `state`

Eingabeformat: `state`

Ausgabeformat:

```
   <team><padding><team>
   <lp>/8000 LP<padding><lp>/8000 LP
   DC: <dc>/40<padding>DC: <dc>/40
   BC: <bc>/5<padding>BC: <bc>/5
<board>
[<show>]
```

Gibt den Spielzustand aus, bestehend aus den Teamnamen, den Lebenspunkten beider Teams, der Anzahl der jeweiligen Stapeleinheiten (*deck count*) und der Anzahl der jeweiligen Einheiten auf dem Spielbrett (*board count*, Bauernkönig nicht inbegriffen). Die Ausgabe ist um zwei Leerzeichen eingerückt, um bündig mit der Ausgabe des Spielbretts zu sein. In jeder Zeile werden Informationen zu beiden Teams ausgegeben, die durch genau so viele Leerzeichen getrennt sind, dass jede Zeile eine Gesamtlänge von 31 hat, um bündig mit dem rechten Ende der Ausgabe des Spielfelds zu sein.

Anschließend wird das Spielbrett ausgegeben, wie in Der Befehl `board` beschrieben, und falls ein Feld ausgewählt ist, der Inhalt des *ausgewählten Feldes*, wie in Der Befehl `show` beschrieben.

Beispielinteraktion

```
> state
  Player                  Enemy
  7500/8000 LP     4000/8000 LP
  DC: 32/40           DC: 33/40
  BC: 2/5               BC: 3/5
  +---+---+---+---+---+---+---+
7 |   |   |   | Y |   |   |   |
  +---+---+---+---+---+---+---+
6 |   |   | y | yb| yb|   |   |
  +---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |
  +---+---#===#---+---+---+---+
4 |   |   N   N   |   |   |   |
  +---+---#===#---+---+---+---+
3 |   |   |   |   |*x |   |   |
  +---+---+---+---+---+---+---+
2 |   |   |*x |   |   |   |   |
  +---+---+---+---+---+---+---+
1 |   |   |   |*X |   |   |   |
  +---+---+---+---+---+---+---+
    A   B   C   D   E   F   G
<no unit>
```

A.5.10 Der Befehl `yield`

Eingabeformat: `yield [<idx>]`

Ausgabeformat:

```
[<team> discarded <name> (<atk>/<def>).]
It is <team>'s turn!
[<team> has no cards left in the deck!]
[<team> wins!]
```

Beendet den aktuellen Zug und setzt das *ausgewählte Feld* zurück. Falls ein Index `<idx>` angegeben wurde, wird die `<idx>`-te Einheit auf der Hand aus der Hand und aus dem Spiel genommen. Ist der Stapel des Teams, das nun am Zug ist, leer und kann es demzufolge nicht ziehen, verliert es das Spiel. Der Befehl ist unzulässig, wenn

- das Team, das aktuell am Zug ist, fünf Einheiten auf der Hand hält und *kein* Index angegeben wurde oder
- ein Index angegeben wurde und das Team, das aktuell am Zug ist, weniger als fünf Einheiten auf der Hand hält.

Ist der Befehl unzulässig, bleibt das Team am Zug; es dürfen bis zum Ende des Zuges jedoch nur noch Der Befehl `hand` und der Befehl `yield` verwendet werden, insbesondere dürfen keine Einheiten mehr platziert oder bewegt werden.

Beispielinteraktion

```
> yield
ERROR: Player's hand is full!
> place 3 C2
ERROR: cannot place a card, you must discard!
> hand
[1] Goat Farmer (1000/1000)
[2] Grain Farmer (1100/1100)
[3] Barrier Builder (900/1700)
[4] Grain Farmer (1100/1100)
[5] Goat Farmer (1000/1000)
> yield 1
Player discarded Goat Farmer (1000/1000).
It is Enemy's turn!
```

A.5.11 Der Befehl `quit`

Eingabeformat: `quit`

Das Programm wird beendet. Dafür darf nicht `System.exit()` oder dergleichen verwendet werden, sondern das Programm soll über gewöhnliche Kontrollflussmechanismen beendet werden.

A.6 Beispielinteraktion

Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle; sie dienen lediglich zur Orientierung in der gegebenen Beispielinteraktion. Nutzereingaben sind mit dem Zeichen `>` gekennzeichnet. Die Zeichen `%>` (Prozentzeichen und Größerzeichen gefolgt von einem Leerzeichen) stellen die Kommandozeile dar. Der Name des Programms dient nur als Beispiel und ist nicht vorgeschrieben.

A.6.1 Beispielinteraktion 1

Beispielinteraktion

```
1  %> java Main seed=-4022738 deck=input/decks/default.txt verbosity=compact
       units=input/units/default.txt
2  Daisy;Farmer;300;500
3  Pitchfork;Farmer;400;400
4  Stable;Farmer;500;300
5  Manure;Spreader;600;600
6  Potato;Farmer;700;500
7  Fence;Farmer;300;1000
8  Barn;Guard;400;900
9  Chicken;Farmer;800;700
10 Tractor;Farmer;900;600
11 Silo;Operator;500;1100
12 Beetroot;Farmer;1000;900
13 Goat;Farmer;1000;1000
14 Pig;Farmer;1200;800
15 Grain;Farmer;1100;1100
16 Threshing;Maid;1300;1000
17 Shield;Farmer;800;1600
18 Barrier;Builder;900;1700
19 Milk Cow;Farmer;1400;1400
20 Livestock;Sorceress;1500;1200
21 Field Marshal;Farmer;1700;1500
22 Agro;Architect;1900;1800
23 Titan Tractor;Farmer;2200;2000
24 Seed;Farmer;2500;2300
25 Egg-laying Wool-Milk-Pig;Farmer;3000;3000
26 3
27 2
28 3
29 2
30 1
31 2
32 2
33 1
34 3
```

Beispielinteraktion

```
1
2
1
1
2
1
2
1
1
1
2
2
2
1
1
Use one of the following commands: select, board, move, flip, block, hand, place, show, yield,
    state, quit.
> hand
[1] Daisy Farmer (300/500)
[2] Threshing Maid (1300/1000)
[3] Shield Farmer (800/1600)
[4] Milk Cow Farmer (1400/1400)
[5] Silo Operator (500/1100)
> select D1
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   N*X N   |   |   |
    A   B   C   D   E   F   G
Player's Farmer King
> move D2
Farmer King moves to D2.
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   N X N   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Player's Farmer King
> select D3
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   N   N   |   |   |
```

Beispielinteraktion

```
2 |   |   |   | X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
<no unit>
> place 2
Player places Threshing Maid on D3.
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   N*x N   |   |   |
2 |   |   |   | X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Threshing Maid (Team Player)
ATK: 1300
DEF: 1000
> hand
[1] Daisy Farmer (300/500)
[2] Shield Farmer (800/1600)
[3] Milk Cow Farmer (1400/1400)
[4] Silo Operator (500/1100)
> move D4
Threshing Maid moves to D4.
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   N x N   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   | X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Threshing Maid (Team Player)
ATK: 1300
DEF: 1000
> board
7 |   |   |   | Y |   |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   N x N   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   | X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
> yield
It is Enemy's turn!
Farmer King moves to D7.
7 |   |   |   N Y N   |   |   |
6 |   |   |   |   |   |   |   |
```

Beispielinteraktion

```
5 |   |   |   |   |   |   |
4 |   |   |   | x |   |   |
3 |   |   |   |   |   |   |
2 |   |   |   | X |   |   |
1 |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Enemy's Farmer King
Enemy places Livestock Sorceress on E6.
7 |   |   |   | Y |   |   |
6 |   |   |   |   N*y N   |
5 |   |   |   |   |   |   |
4 |   |   |   | x |   |   |
3 |   |   |   |   |   |   |
2 |   |   |   | X |   |   |
1 |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Livestock Sorceress (Team Enemy)
ATK: 1500
DEF: 1200
Livestock Sorceress moves to E7.
7 |   |   |   | Y N y N   |   |
6 |   |   |   |   |   |   |
5 |   |   |   |   |   |   |
4 |   |   |   | x |   |   |
3 |   |   |   |   |   |   |
2 |   |   |   | X |   |   |
1 |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Livestock Sorceress (Team Enemy)
ATK: 1500
DEF: 1200
It is Player's turn!
> hand
[1] Daisy Farmer (300/500)
[2] Shield Farmer (800/1600)
[3] Milk Cow Farmer (1400/1400)
[4] Silo Operator (500/1100)
[5] Egg-laying Wool-Milk-Pig Farmer (3000/3000)
> select E7
7 |   |   |   | Y N y N   |   |
6 |   |   |   |   |   |   |
5 |   |   |   |   |   |   |
4 |   |   |   |*x |   |   |
3 |   |   |   |   |   |   |
2 |   |   |   |*X |   |   |
1 |   |   |   |   |   |   |
    A   B   C   D   E   F   G
??? (Team Enemy)
ATK: ???
```

Beispielinteraktion

```
DEF: ???
> select D4
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   |   |   |   |   |
4 |   |   |   N*x N   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   |*X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Threshing Maid (Team Player)
ATK: 1300
DEF: 1000
> move D5
Threshing Maid moves to D5.
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   N x N   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   |*X |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Threshing Maid (Team Player)
ATK: 1300
DEF: 1000
> select D2
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | x |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   N*X N   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Player's Farmer King
> move D3
Farmer King moves to D3.
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | x |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   N X N   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Player's Farmer King
> yield 1
Player discarded Daisy Farmer (300/500).
```

Beispielinteraktion

```
It is Enemy's turn!
Farmer King moves to D7.
7 |   |   |   N Y N*y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | x |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   | X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Enemy's Farmer King
Enemy places Seed Farmer on D6.
7 |   |   |   | Y |*y |   |   |
6 |   |   |   N*y N   |   |   |
5 |   |   |   | x |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   | X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Seed Farmer (Team Enemy)
ATK: 2500
DEF: 2300
Seed Farmer (2500/2300) attacks ??? on D5!
Seed Farmer (2500/2300) was flipped on D6!
Threshing Maid (1300/1000) was flipped on D5!
Threshing Maid was eliminated!
Player takes 1200 damage!
Seed Farmer moves to D5.
7 |   |   |   | Y |*y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   N y N   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   | X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Seed Farmer (Team Enemy)
ATK: 2500
DEF: 2300
Livestock Sorceress moves to E7.
7 |   |   |   | Y N y N   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | y |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   | X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
```

Beispielinteraktion

```
Livestock Sorceress (Team Enemy)
ATK: 1500
DEF: 1200
It is Player's turn!
> hand
[1] Shield Farmer (800/1600)
[2] Milk Cow Farmer (1400/1400)
[3] Silo Operator (500/1100)
[4] Egg-laying Wool-Milk-Pig Farmer (3000/3000)
[5] Beetroot Farmer (1000/900)
> select E7
7 |   |   |   | Y N y N   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | y |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
??? (Team Enemy)
ATK: ???
DEF: ???
> state
  Player                  Enemy
  6800/8000 LP     8000/8000 LP
  DC: 33/40           DC: 34/40
  BC: 0/5               BC: 2/5
7 |   |   |   | Y N y N   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | y |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
??? (Team Enemy)
ATK: ???
DEF: ???
> select D5
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   N y N   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Seed Farmer (Team Enemy)
ATK: 2500
```

Beispielinteraktion

```
DEF: 2300
> select D4
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | y |   |   |   |
4 |   |   |   N   N   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
<no unit>
> place 4
Player places Egg-laying Wool-Milk-Pig Farmer on D4.
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   | y |   |   |   |
4 |   |   |   N*x N   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Egg-laying Wool-Milk-Pig Farmer (Team Player)
ATK: 3000
DEF: 3000
> move D5
Egg-laying Wool-Milk-Pig Farmer (3000/3000) attacks Seed Farmer (2500/2300) on D5!
Egg-laying Wool-Milk-Pig Farmer (3000/3000) was flipped on D4!
Seed Farmer was eliminated!
Enemy takes 500 damage!
Egg-laying Wool-Milk-Pig Farmer moves to D5.
7 |   |   |   | Y | y |   |   |
6 |   |   |   |   |   |   |   |
5 |   |   |   N x N   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |*X |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   |   |   |   |   |   |
    A   B   C   D   E   F   G
Egg-laying Wool-Milk-Pig Farmer (Team Player)
ATK: 3000
DEF: 3000
> quit
```