

---

# Programmieren – Wintersemester 2022/23

---

## Abschlussaufgabe 1

Version 1.0

20 Punkte

Ausgabe: 13.02.2023, ca. 12:00 Uhr

Abgabe: 27.02.2023, 12:00 Uhr

Abgabefrist: 14.03.2023, 06:00 Uhr

---

## Geschlechtergerechte Sprache

Wenn das generische Maskulinum gewählt wurde, geschieht dies zur besseren Lesbarkeit und zum einfachen Verständnis der Aufgabenstellung. Sofern nicht anders angegeben, beziehen sich Angaben im Sinne der Gleichbehandlung auf Vertretende aller Geschlechter.

## 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 zur Bewertung „nicht bestanden“. Ausdrücklich ausgenommen hiervon sind Quelltextsnipsel 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.

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 zum Ausschluss der Erfolgskontrolle führen.

## Kommunikation und aktuelle Informationen

In unseren *FAQs*<sup>1</sup> 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.

---

<sup>1</sup><https://sdq.kastel.kit.edu/wiki/Programmieren/FAQ>

In den *ILIAS-Foren* 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<sup>2</sup> einsehen.

## Bearbeitungshinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der verpflichtenden Tests für eine erfolgreiche Abgabe von Abschlusssaufgabe 1 notwendig ist. Ihre Abgabe wird automatisch mit null Punkten bewertet, falls eine der nachfolgenden Regeln verletzt ist. Sie müssen zuerst die verpflichtenden Tests 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 17*.
- 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.util.stream` und `java.util.function`.
- Achten Sie darauf, nicht zu lange Zeilen, Methoden und Dateien zu erstellen. Sie müssen bei Ihren Lösungen eine maximale Zeilenbreite von 120 Zeichen einhalten.
- Halten Sie alle Whitespace-Regeln ein.
- Halten Sie alle Regeln zu Variablen-, Methoden- und Paketbenennung ein.
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Nutzen Sie nicht das `default`-Package.
- `System.exit()`, `Runtime.exit()` oder ähnliches dürfen nicht verwendet werden.
- Halten Sie die Regeln zur Javadoc-Dokumentation ein.
- Halten Sie auch alle anderen Checkstyle-Regeln ein.

Diese folgenden Bearbeitungshinweise sind relevant für die Bewertung Ihrer Abgabe. Dennoch wird Ihre Abgabe durch das Abgabesystem *nicht* automatisch mit null Punkten bewertet, falls eine der nachfolgenden Regeln verletzt ist.

- Fügen Sie außer Ihrem u-Kürzel keine weiteren persönlichen Daten zu Ihren Abgaben hinzu.
- Beachten Sie, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden. Halten Sie die Hinweise zur Modellierung im ILIAS-Wiki ein.

---

<sup>2</sup><https://artemis.praktomat.cs.kit.edu/>

- Programmcode muss in englischer Sprache verfasst sein.
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich.
- Die Kommentare sollen einheitlich in englischer oder deutscher Sprache verfasst werden.
- Geben Sie im Javadoc-Autoren-Tag nur Ihr u-Kürzel an.
- Wählen Sie aussagekräftige Namen für alle Ihre Bezeichner.

## 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 im ILIAS beschreibt, wie Checkstyle verwendet werden kann.

## Abgabehinweise

Die Abgabe im Online-Einreichungssystem wird am 27.02.2023, 12:00 Uhr, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Einreichungssystem bei der richtigen Aufgabe vor Ablauf der Abgabefrist am 14.03.2023, 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.

## Prüfungsmodus in Artemis

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

## Aufgabe A: Queens Farming

(20 Punkte)

*Queens Farming* ist ein rundenbasiertes Gesellschaftsspiel, welches allein oder mit mehreren Personen gespielt werden kann. Im Spiel schlüpfen die Spielenden in die Rolle der Kinder von Königin Josephine, welche Namensgeber des Spiels ist. Die Kinder sollen lernen mit Geld umzugehen und bekommen daher von der Königin ein Startkapital und das Ziel einen gewissen Betrag in Gold zu erwirtschaften. Hierfür können sie Gemüse anpflanzen, ernten und verkaufen. Außerdem bekommt jedes Kind zu Beginn eine festgelegte Ackerfläche zum Bewirtschaften. Die Kinder können auch neues Land mit dem erwirtschafteten Gold kaufen. Gewonnen haben die Kinder, welche zuerst den festgelegten Betrag erwirtschaftet haben.

### A.1 Das Spiel

*Queens Farming* benötigt einige Spielelemente, um gespielt zu werden. Hierzu gehören

- *Gold*, um das die Kinder spielen,
- *Carrot*, *Salad*, *Tomato* und *Mushroom* – das Gemüse, das die Kinder anbauen und verkaufen können,
- *Barn*, *Garden*, *Field*, *Large Field*, *Forest*, *Large Forest* – die Arten von Spielkacheln, die die Kinder kaufen und bebauen können,
- und der *Market*, auf dem die Kinder Gemüse einkaufen oder verkaufen können.

#### A.1.1 Gemüse

Es gibt genau vier Arten von Gemüse, das die Kinder anbauen und verkaufen können. Jedes Gemüse hat eine bestimmte Zeit in Runden zugeordnet, die das Gemüse braucht, um auf einer Spielkachel zu wachsen. Dies ist in Tabelle A.1 dargestellt. Hierbei wächst ein Gemüse, indem die Anzahl auf der entsprechenden Spielkachel verdoppelt wird. Befinden sich z. B. zwei *Carrots* auf einem *Field*, so befinden sich nach einer Runde vier *Carrots* auf dem Feld.

Art (Abkürzung)	Runden zum Wachsen
Carrot (C)	1
Salad (S)	2
Tomato (T)	3
Mushroom (M)	4

Tabelle A.1: Gemüsearten im Spiel

#### A.1.2 Spielkacheln

Je nach Anzahl der Kinder der Königin gibt es unterschiedlich viele Spielkacheln. Insgesamt gibt es genau sechs verschiedene Arten. Eine Übersicht über die Kacheln findet sich in Tabelle A.2.

Art (Abkürzung)	Anzahl im Spiel	Kapazität	Anbaubares Gemüse
Barn (B)	$1 * n$	$\infty$	—
Garden (G)	$4 * n$	2	Alle
Field (Fi)	$4 * n$	4	Carrot, Salad, Tomato
Large Field (LFi)	$2 * n$	8	Carrot, Salad, Tomato
Forest (Fo)	$2 * n$	4	Carrot, Mushroom
Large Forest (LFo)	$1 * n$	8	Carrot, Mushroom

 Tabelle A.2: Spielkacheln im Spiel bei  $n$  Kindern

**A.1.2.1 Barn** Die Scheune (*Barn*) wird von den Kindern zum Lagern des Gemüses verwendet. Es kann nur Gemüse aus der Scheune auf dem Markt (*Market*) verkauft werden. Genauso wird eingekauftes und geerntetes Gemüse direkt in die Scheune eingelagert.

Sobald sich in der Scheune mindestens ein Gemüse ( $\#(\text{Gemüse}) > 0$ ) befindet, startet für die Scheune ein Countdown von 6 Runden. Nachdem der Countdown abgelaufen ist (also am Ende der 6. Runde), verfault das gesamte Gemüse in der Scheune und scheidet aus dem Spiel aus. Wenn die Scheune leer ist, also kein Gemüse enthält, hat sie keinen Countdown.

**A.1.2.2 Kacheln zum Anbauen** Neben der Scheune gibt es fünf weitere Arten von Spielkacheln (siehe Tabelle A.2). Diese unterscheiden sich in der Anzahl, in der sie im Spiel vorhanden sind. Die Anzahl ist abhängig von der Anzahl der Kinder  $n$  im Spiel. Außerdem unterscheiden sie sich durch ihre Kapazität, die angibt, wie viel Gemüse sich maximal auf ihnen befinden kann. Zuletzt unterscheiden sie sich noch durch die Arten von Gemüse, das auf ihnen angebaut werden kann. Sobald auf der Kachel ein Gemüse angebaut wird, startet für die Kachel ein Countdown von  $j$  Runden.  $j$  ist hier die Anzahl an Runden, die das entsprechende Gemüse zum Wachsen braucht. Nach dem Ablauf des Countdowns wird das entsprechende Gemüse auf der Kachel verdoppelt (hierbei kann die Kapazität der Kachel nicht überschritten werden). Hat die Kachel ihre Kapazität erreicht wird der Countdown entfernt.

### A.1.3 Spielstart

Zum Spielstart wird festgelegt, wie viel Gold  $G_{Start}$  jedes Kind erhält ( $G_{Start} \geq 0$ ) und wie viel Gold  $G_{Win}$  ein Kind haben muss, um zu gewinnen ( $G_{Win} \geq 1$ ). Jedes Kind bekommt zu Beginn eigenes Land. Dieses besteht aus 4 Kacheln, die immer gleich angeordnet sind: An der Position  $(0, 0)$  befindet sich die Scheune (*Barn*), in der Waren gelagert werden können. An den Positionen links  $(-1, 0)$  und rechts  $(1, 0)$  von der Scheune befindet sich jeweils ein Garten (*Garden*). An der Position über der Scheune  $(0, 1)$  befindet sich ein Feld (*Field*).

Insgesamt sieht also das Spielfeld jedes der Kinder zu Beginn wie folgt aus:

**A.1.3.1 Besitz der Kinder** Zu Beginn erhält jedes Kind ein Startkapital in Gold. Außerdem enthält die Scheune jedes Kindes zu Spielbeginn genau ein Gemüse jeder Art.

	Fi	
G	B	G

Tabelle A.3: Das Spielfeld jedes Kindes zu Beginn des Spiels. Barn ist an Position (0,0)

**A.1.3.2 Mischen der Kacheln** Nachdem jedes Kind die entsprechenden Kacheln erhalten hat, werden die übrigen Kacheln gemischt. Hierzu werden also  $2n$  Garden,  $3n$  Field,  $2n$  Large Field,  $2n$  Forest und  $n$  Large Forest in dieser Reihenfolge angeordnet und durchgemischt.

**A.1.3.3 Der Markt** Der Markt (*Market*) definiert die Zusammenhänge und Preisspannen der verschiedenen Gemüsearten. Hierbei sind die Preise von Pilzen (*Mushrooms*) und Karotten (*Carrots*) gekoppelt, der Preisindikator für diese Kopplung ist  $\diamond$ . Außerdem sind die Preise von Tomaten (*Tomatoes*) und Salat (*Salads*) gekoppelt, der Preisindikator für diese Kopplung ist  $\star$ . Zu Spielbeginn sind die Preise wie in Tabelle A.4 dargestellt. Karotten kosten also z. B. zu Beginn 2 Gold.

Mushroom	Carrot		Tomato	Salad
12	3		3	6
15	2		5	5
16	$\diamond$ 2		6	$\star$ 4
17	2		7	3
20	1		9	2

Tabelle A.4: Der Markt zu Spielbeginn

Nach dem Spielzug eines Kindes werden die Preise des Markts angepasst. Dabei können die Indikatoren nicht über die oberste Zelle und nicht unter die unterste Zelle hinausgeschoben werden. Die Preisindikatoren werden basierend auf der Anzahl der verkauften Gemüse wie folgt verändert.

- Paare die Pilze mit Karotten und die Tomaten mit Salaten und entferne alle Paare.
- Für je 2 verbleibende Pilze wird  $\diamond$  eine Zelle nach oben verschoben.
- Für je 2 verbleibende Karotten wird  $\diamond$  eine Zelle nach unten verschoben.
- Für je 2 verbleibende Tomaten wird  $\star$  eine Zelle nach oben verschoben.
- Für je 2 verbleibende Karotten wird  $\star$  eine Zelle nach unten verschoben.

Ein Beispiel zur Veranschaulichung: *Mira, ein Kind der Königin, verkaufte in ihrem Zug 5 Pilze, 2 Karotten, 1 Tomate und 3 Salate. Nach dem Beenden ihres Zugs wird der Markt wie folgt verändert: Nach dem Entfernen aller Paare gibt es noch 3 Pilze und 2 Salate. Daher wird  $\diamond$  um eins nach oben und  $\star$  um eins nach unten geschoben.*

### A.1.4 Das Spiel spielen

Nachdem die Spielvorbereitungen abgeschlossen sind, kann das Spiel beginnen. Hierbei werden Runden (*Rounds*), Spielzüge der Kinder (*Turns*) und Aktionen innerhalb der Spielzüge (*Actions*) unterschieden. In einer Runde führen die Kinder in fester Reihenfolge jeweils einen Spielzug durch. Innerhalb eines Spielzugs kann ein Kind maximal zwei Aktionen ( $\#(\text{Action}) \leq 2$ ) durchführen. Nach dem Ende einer Runde (also nachdem alle Kinder einen Spielzug vollendet haben), schreitet die Zeit um einen Zeitschritt voran. Dies hat Einfluss auf das Wachsen des Gemüses auf den Kacheln und auf den Verfall des Gemüses in den Scheunen.

Der Ablauf des Spiels ist auch in Abbildung A.1 dargestellt.

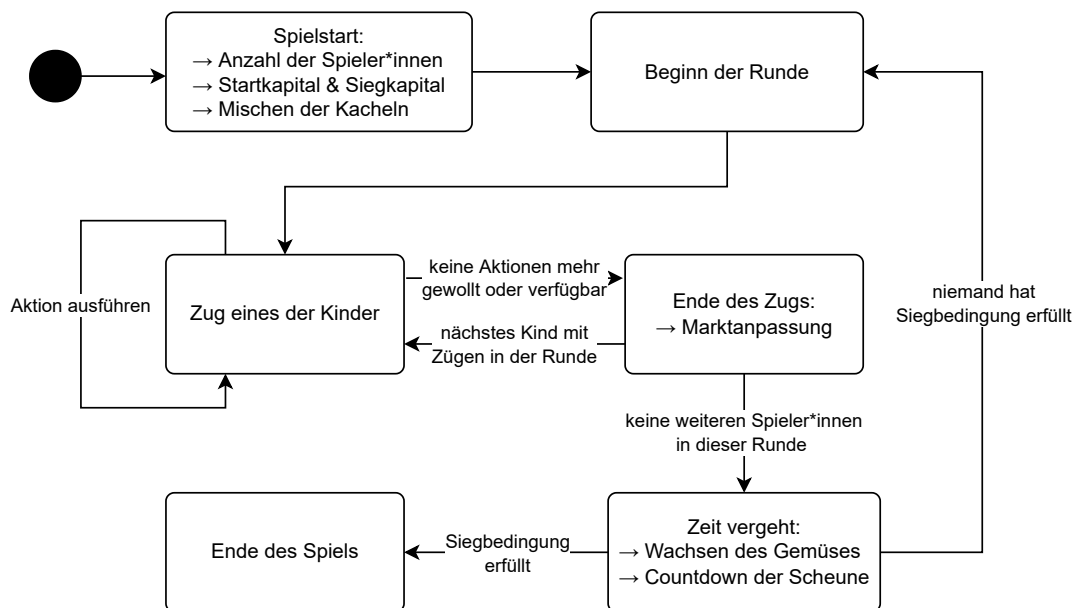


Abbildung A.1: Übersicht des Spielablaufs als Aktivitätsdiagramm

**A.1.4.1 Aktionen der Kinder** Im Folgenden werden die verschiedenen Aktionen der Kinder erläutert.

**Plant** Falls das Kind ein entsprechendes Gemüse in der Scheune hat, kann dieses auf einer Kachel ohne Gemüse angebaut werden. Zu beachten ist hierbei die Art der Kachel. So können z. B. in einem Wald (*Forest*) nur Karotten und Pilze angebaut werden (siehe Tabelle A.2). Außerdem wird für die Kachel ein Countdown gestartet. Dieser startet mit der Zeit in Runden, die das entsprechende Gemüse zum Wachsen braucht.

**Harvest** Ein Kind kann Gemüse von den Kacheln ernten, sobald auf der entsprechenden Kachel mindestens ein Gemüse ist. Beim Ernten nimmt das Kind die gewünschte Anzahl an Gemüse von genau einer Kachel und legt sie in die Scheune. Befindet sich nach der Ernte kein Gemüse mehr auf der Kachel, wird der Countdown für die Kachel entfernt. Andernfalls bleibt, wenn möglich,

der vorhandene Countdown erhalten. Ist dies nicht möglich (für die Kachel läuft derzeit kein Countdown) wird ein neuer Countdown für die Kachel gestartet.

**Sell** Ein Kind der Königin kann beliebig viel Gemüse aus der Scheune verkaufen. Das Gold für den Verkauf wird dem Kind sofort gutgeschrieben. Zur Erinnerung: Der Markt wird erst nach dem Ende des Spielzugs des Kindes verändert, nicht direkt nach dem Verkauf.

**Buy Vegetable** Mit Gold können Kinder der Königin genau ein Gemüse einer Gemüseart einkaufen. Das gekaufte Gemüse kommt direkt in die Scheune.

**Buy Land** Neben Gemüse kann ein Kind auch weitere Kacheln für Gold kaufen und so das eigene Spielfeld erweitern. Beim Kauf von neuem Land (einer Kachel) wird die erste Kachel vom gemischten Stapel der verbleibenden Kacheln gezogen. Die Art der Kachel ist also zufällig.

Die Kosten für die Kachel ergeben sich aus der Position, an der die Kachel platziert werden soll. Eine neue Kachel kann links, rechts oder oberhalb einer bestehenden Kachel platziert werden. Die Kosten werden mithilfe der Manhattan-Distanz  $d_1$  zur Scheune  $B$  (0,0) bestimmt. Für das Platzieren einer neuen Kachel  $T$  an der Position  $(T_x, T_y)$  fallen folgende Kosten an:

$$Gold(T) = 10 * (d_1(T, B) - 1) = 10 * (|T_x - B_x| + |T_y - B_y| - 1)$$

Eine Darstellung der Kosten für neue Kacheln ist in Tabelle A.5 dargestellt. Hierbei sind die Kosten in Boxen dargestellt (z. B.  $\boxed{1}$ ). Die Positionen, auf die keine neuen Kacheln platziert werden können, sind mit  $\infty$  markiert.

$\infty$	$\infty$	$\boxed{20}$	$\boxed{10}$	$\infty$	$\infty$	$\infty$
$\infty$	$\boxed{20}$	<b>Fi</b>	<b>Fi</b>	$\boxed{10}$	$\boxed{20}$	$\infty$
$\infty$	$\boxed{10}$	<b>G</b>	<b>B</b>	<b>G</b>	<b>Fo</b>	$\boxed{20}$

Tabelle A.5: Kosten für das Kaufen von Spielkacheln.  $\infty$  stellt dar, dass das Feld nicht gekauft werden kann.

**A.1.4.2 Ende des Spiels** Das Spiel endet entweder, wenn die Kinder entscheiden, dass sie nicht mehr spielen wollen oder falls am Ende einer Runde mindestens eines der Kinder die nötige Menge Gold hat ( $G_{Kind} \geq G_{Win}$ ). Falls mehrere Kinder die nötige Menge Gold erreicht haben, haben all diese Kinder das Spiel gewonnen. Hat keines der Kinder die nötige Menge Gold erreicht, haben die Kinder mit dem meisten Gold gewonnen.



## A.2 Implementierung von Queens Farming

Ihre Aufgabe ist es das Spiel mittels einer Text-basierten Schnittstelle spielbar zu machen. Die dafür nötigen Interaktionen werden im Folgenden dargestellt.

Achten Sie darauf, dass durch die Ausführung der folgenden Befehle die gegebenen semantischen und syntaktischen Spezifikationen nicht verletzt werden und geben Sie bei Verletzung der Spezifikationen immer eine aussagekräftige Fehlermeldung aus. Wenn die Benutzereingabe nicht dem vorgegebenen Format entspricht, ist auch eine Fehlermeldung auszugeben. Nach der Ausgabe einer Fehlermeldung soll das Programm fortfahren und auf die nächste Eingabe warten. Die Fehlermeldung sollte so geformt sein, dass für den Benutzer erkenntlich ist, warum eine Eingabe abgelehnt wurde.

Da wir automatische Tests Ihrer interaktiven Benutzerschnittstelle durchführen, müssen die Ausgaben exakt den Vorgaben entsprechen. Insbesondere sollen sowohl Klein- und Großbuchstaben als auch die Leerzeichen und Zeilenumbrüche genau übereinstimmen. Setzen Sie nur die in der Aufgabenstellung angegebenen Informationen um. Geben Sie auch keine zusätzlichen Informationen aus. Bei Fehlermeldungen dürfen Sie den englischsprachigen Text frei wählen, er sollte jedoch sinnvoll sein. Jede Fehlermeldung muss aber mit **Error:** beginnen und darf keine Sonderzeichen, wie beispielsweise Zeilenumbrüche oder Umlaute, enthalten.

Wenn nicht anders angegeben, ist für Eingabe immer die Standardeingabe `System.in` zu verwenden. Wenn nicht anders angegeben, ist für Ausgaben immer die Standardausgabe `System.out` zu verwenden. Für Fehlermeldungen kann anstelle der Standardausgabe optional die Standardfehlerausgabe `System.err` verwendet werden. Weisen Sie diese Standardeingabe und -ausgabe niemals neu zu.

### A.2.1 Darstellung der Beispielinteraktionen

In Beispielinteraktionen stellt das Symbol `%>` (Prozent-Zeichen und Größer-Zeichen gefolgt von einem Leerzeichen) die Kommandozeile dar. Der Programmname ist frei gewählt und muss bei Ihnen nicht *QueensFarming* lauten. Das Symbol `>` (Größer-Zeichen gefolgt von einem Leerzeichen) stellt eine Benutzereingabe dar und ist selbst nicht Teil der Eingabe. Sollte in einer Interaktion `[...]` geschrieben werden, so bedeutet dies, dass weitere Interaktionen ausgelassen worden sind, um auf eine spezielle Stelle der Interaktion einzugehen. `[...]` ist keine Ausgabe Ihres Spiels. Darstellungen in eckigen Klammern `[]`, wie `[player name]` stellen ebenso Platzhalter dar. Hierbei wird innerhalb der Klammern eine kurze Erklärung zum Verständnis eingefügt. Beachten Sie, dass solche Platzhalter niemals Teil der Ausgabe Ihres Programms sind. Zuletzt gibt es noch den Platzhalter `_`, welcher benutzt wird, um in ausgewählten Beispielinteraktionen ein Leerzeichen darzustellen. Dieses ist auch nie Teil der Ausgabe des Programms. Beachten Sie bitte, dass Sie die Interaktionen nicht aus dieser PDF kopieren sollten, da es beim Kopieren aus PDFs zu Veränderungen kommen kann. Verwenden Sie hierfür stattdessen die Textdateien im Ilias und fragen Sie bei Unklarheiten zum Format in den entsprechenden Foren nach.

## A.2.2 Quit-Befehl

Generell muss Ihr Programm jederzeit mit der Eingabe des Befehls `quit` beendet werden können. Beachten Sie, dass für das Testen Ihrer Abgabe dieser Befehl essenziell ist, da viele der Tests `quit` am Ende einer Testsequenz verwenden, um Ihr Programm zu beenden. Stellen Sie daher sicher, dass der Befehl in jeder Situation funktioniert. Beachten Sie bitte nochmals, dass Sie hierfür nicht `System.exit()` oder andere ausgeschlossene Funktionen verwenden dürfen.

## A.2.3 Initialisierung des Spiels

Das Spiel wird ohne Kommandozeilenargumente gestartet. Sollten ein oder mehrere Argumente übergeben werden, wird eine aussagekräftige Fehlermeldung ausgegeben und das Programm wird ohne weitere Ausgabe beendet.

Ansonsten beginnt das Programm mit der Ausgabe einer Pixel-Art, die Ihnen ebenfalls im ILIAS zur Verfügung steht. Abschließend wird abgefragt, wie viele Kinder mitspielen, wie diese heißen, wie viel Gold das Startkapital ist und wie viel Gold man zum Gewinnen braucht. Außerdem wird ein Seed für das Mischen der übrigen Kacheln erfragt.

Die Namen der Kinder müssen dem regulären Ausdruck `[A-Za-z]+` entsprechen. Ist eine Eingabe ungültig, wird eine aussagekräftige Fehlermeldung ausgegeben. Jede Abfrage wird so lange wiederholt, bis eine gültige Eingabe erfolgt ist. Nach der Initialisierung des Spiels startet der Zug des ersten Kinds der Königin.

Die Initialisierung des Spiels kann jederzeit durch den `quit`-Befehl beendet werden. In diesem Fall wird das Programm ohne eine weitere Ausgabe sofort beendet.

**Mischen** Bevor die erste Runde des Spiels gestartet werden kann, müssen die nicht vergebenen Kacheln gemischt werden. Verwenden Sie den Konstruktor `Random(long seed)`<sup>3</sup>, um einen neuen Zufallszahlengenerator mithilfe des *Seeds* zu instanziiieren.

Für das Mischen müssen die verbleibenden Kacheln in einer `List`<sup>4</sup> (z.B. einer `ArrayList`<sup>5</sup>) abgespeichert werden. Beachten Sie hierbei, dass die vorgegebene Reihenfolge der Spielanleitung exakt eingehalten werden muss. Um nun diese Liste nach dem Zufallsprinzip unter Verwendung der angegebenen Zufallsquelle zu mischen, muss diese der Methode `Collections::shuffle(List<?> list, Random random)`<sup>6</sup> übergeben werden.

Für die Vergabe der Kacheln werden aus der gemischten Liste immer Elemente beginnend bei Index 0 entnommen. Der mögliche Wertebereich eines *Seeds* umfasst in dieser Aufgabe das abgeschlossene Intervall `[-2147483648, 2147483647]`.

<sup>3</sup>[https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Random.html#%3Cinit%3E\(long\)](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Random.html#%3Cinit%3E(long))

<sup>4</sup><https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html>

<sup>5</sup><https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html>

<sup>6</sup>[https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html#shuffle\(java.util.List,java.util.Random\)](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html#shuffle(java.util.List,java.util.Random))

## ➤ Beispielinteraktion

```

1  %> java QueensFarming
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

```

### A.2.4 Beginn eines Zugs

Der Beginn eines Zugs läuft immer gleich ab. Die Ausgabe zu Beginn besteht aus maximal vier Zeilen, wobei zwei der Zeilen optional sind: Als Erstes wird zur besseren Lesbarkeit eine leere Zeile ausgegeben. Anschließend wird die Zeile `It is [player name]'s turn!` ausgegeben. Sollte seit dem letzten Zug des Kindes Gemüse nachgewachsen sein, so wird dies nach dem Schema `[n] vegetable has grown since your last turn.` bzw. `[n] vegetables have grown since your last turn.` dargestellt. Welches der beiden Ausgaben verwendet wird, hängt von der Anzahl  $n$  des gewachsenen Gemüses ab und muss entsprechend gewählt werden. Sollten das Gemüse in der Scheune durch den Ablauf des Countdown verfallen sein, so wird dies mittels `The vegetables in your barn are spoiled.` dargestellt.

### ➤ Beispielinteraktion

```

1 | [...]
2 |
3 | It is Mira's turn!
4 | 1 vegetable has grown since your last turn.
5 | The vegetables in your barn are spoiled.
6 | > show board
7 | [...]
```

#### A.2.5 Ende eines Zugs

Das Ende eines Zugs kann auf zwei Arten eintreten: Falls das Kind des aktuellen Zugs zwei Aktionen durchgeführt hat, wird automatisch der Zug beendet. Alternativ kann dies durch die Eingabe des Befehls `end turn` erzwungen werden.

Danach beginnt sofort der Zug des nächsten Kindes. Beachten Sie hierbei, dass die automatische Auswertung der Siegesbedingung (Vermögen der Kinder) am Ende einer Runde und nicht am Ende eines Zugs stattfindet.

#### A.2.6 Ende des Spiels

Das Ende des Spiels kann auf zwei Arten eintreten: Falls die Siegesbedingung am Ende einer Runde erfüllt ist, beendet sich das Spiel automatisch. Alternativ kann dies durch die Eingabe des Befehls `quit` erzwungen werden.

Zum Ende des Spiels erfolgt eine finale Ausgabe des Systems. Zunächst wird das aktuelle Vermögen der Kinder ausgegeben. Hierbei wird das Schema `Player [player number] ([player name]): [amount of gold]` verwendet. Abschließend wird noch ausgegeben, wer gewonnen hat. Hierfür gibt es je nach Anzahl der Sieger das Schema `[player name], ..., [player name] and [player name] have won!` oder `[player name] and [player name] have won!` oder `[player name] has won!` verwendet.

### ➤ Beispielinteraktion

```

1 | [...]
2 | It is Vincent's turn!
3 | > end turn
4 | Player 1 (Mira): 76
5 | Player 2 (Milan): 42
6 | Player 3 (Vincent): 42
7 | Mira has won!
```

#### A.2.7 Befehl: show

Der Befehl `show` wird dafür verwendet, den aktuellen Zustand des Spiels anzuzeigen.

**A.2.7.1 show barn** Mit der Eingabe von `show barn` wird der Zustand der Scheune des Kindes, das am Zug ist, angezeigt. In der ersten Zeile der Ausgabe wird `Barn` ausgegeben, gefolgt von der Information, wie viele Runden noch vergehen, bis das gelagerte Gemüse verfällt. Anschließend folgt eine Auflistung des in der Scheune vorhandenen Gemüses. Hierbei werden die Gemüse aufsteigend nach der Quantität sortiert. Sollten zwei Gemüsearten in der gleichen Anzahl vorhanden sein, werden diese aufsteigend alphabetisch sortiert. Die Ausgabe des Gemüses endet mit der Summe der verschiedenen Gemüse, welche durch eine Reihe von Trennstrichen (Minus-Zeichen) von der Auflistung getrennt werden. Getrennt mit einer Leerzeile folgt anschließend die Anzeige des vorhandenen Goldes. Sollte kein Gemüse in der Scheune vorhanden sein, so entfällt die Auflistung der Gemüsearten und die Summe mit folgender Leerzeile. Auch die Anzeige, in wie vielen Zügen das Gemüse verfällt, wird nicht angezeigt. Siehe dazu die folgenden Beispielinteraktionen.

Bei der Aufzählung des Gemüses wird deren Pluralform gefolgt von einem Doppelpunkt-Zeichen verwendet. Die verwendeten Pluralformen sind hierbei: `carrots`, `mushrooms`, `salads` und `tomatoes`. Beachten Sie, dass die Breite der Ausgabe durch die Anzahl der Ziffern der Zahlen bestimmt wird. Die Zahlen sollen alle rechtsbündig ausgerichtet sein, sodass die größte Zahl mit einem Leerzeichen von allen Doppelpunkt-Zeichen getrennt ist (siehe dazu die Beispielinteraktion unten).

▶ Beispielinteraktion (sichtbare Leerzeichen)

```

1  [...]
2  >_show_barn
3  Barn_(spoils_in_3_turns)
4  mushrooms:_1
5  tomatoes:___1
6  salads:_____2
7  -----
8  Sum:_____4
9
10 Gold:_____44
11 [...]
12 >_show_barn
13 Barn_(spoils_in_1_turn)
14 salads:_1
15 -----
16 Sum:_____1
17
18 Gold:___42
19 [...]
20 >_show_barn
21 Barn
22 Gold:_2028
23 [...]
```

**A.2.7.2 show board** Mit der Eingabe von `show board` wird der Zustand des Spielfelds des aktuellen Kindes dargestellt. Die Darstellung erfolgt hierbei immer nach dem folgenden Schema: Jede Kachel ist drei Zeilen hoch und 5 Zeichen breit. Getrennt werden Kacheln durch ein einzelnes Symbol | pro Zeile.

Die Kacheln werden immer mit den zugehörigen Abkürzungen beschrieben (vgl. Tabelle A.2). Diese befinden sich für alle Kacheln auf denen Gemüse angebaut werden kann, in der ersten Zeile. Für die Scheune wird die zweite Zeile verwendet. In der Zeile der Abkürzung wird auch immer der Countdown der Kachel dargestellt. Dies erfolgt entweder durch die entsprechende einstellige Zahl oder durch das Symbol \*, welches andeutet, dass kein Countdown vorhanden ist. Dies ist der Fall, wenn kein Gemüse in der Scheune ist, eine Kachel nicht bepflanzt ist oder sie ihre Kapazität erreicht hat. Für alle Kacheln, auf denen Gemüse angebaut werden kann, wird in der dritten Zeile die belegte und verfügbare Kapazität angegeben. Dies erfolgt nach dem Schema [belegter Platz]/[Kapazität]. Außerdem wird, sofern Gemüse auf der entsprechenden Kachel angebaut ist, die Art des Gemüses in der mittleren Zeile durch die entsprechende Abkürzung dargestellt. Entnehmen Sie das genaue Ausgabeformat der Beispielinteraktion.

#### ➤ Beispielinteraktion (sichtbare Leerzeichen)

```

1  [...]
2  >_show_board
3  |_G_*_|
4  |_0/2_|
5  |_Fi_*_|_Fo_*_|_LFo_4_|_Fi_*_|
6  |_0/4_|_0/4_|_2/8_|_0/4_|
7  |_G_*_|_G_*_|_G_*_|_Fo_1_|_LFi_3_|
8  |_S_|_B_2_|_C_|_T_|
9  |_2/2_|_0/2_|_0/2_|_1/4_|_7/8_|
10 [...]

```

#### ➤ Beispielinteraktion

```

1  [...]
2  > show board
3  | G * |
4  |   |
5  | 0/2 |
6  | Fi *| Fo *|   |LFo 4| Fi *|
7  |   |   |   |   | M |   |
8  | 0/4| 0/4|   | 2/8| 0/4|
9  | G *|   | G *| G *| Fo 1|LFi 3|
10 | S | B 2|   |   | C | T |
11 | 2/2|   | 0/2| 0/2| 1/4| 7/8|
12 [...]

```

**A.2.7.3 show market** Mit der Eingabe von **show market** wird der Zustand des Markts dargestellt. Die Darstellung der Kosten der Gemüsearten erfolgt hierbei immer in fester Reihenfolge. Beachten Sie auch hier, dass genau wie bei **show barn** die Zahlen rechtsbündig ausgerichtet werden.

#### ➤ Beispielinteraktion (sichtbare Leerzeichen)

```
1  [...]
2  >_show_market
3  mushrooms:_16
4  carrots:____2
5  tomatoes:___6
6  salads:_____4
7  [...]
```

#### ➤ Beispielinteraktion

```
1  [...]
2  > show market
3  mushrooms: 16
4  carrots:   2
5  tomatoes:  6
6  salads:    4
7  [...]
```

### A.2.8 Befehl: sell

Um Gold zu erhalten, können die Kinder Gemüse aus der Scheune verkaufen. Hierfür wird der Befehl **sell** verwendet. Ein möglicher Parameter für den Befehl ist **all**. Falls dieser übergeben wird, wird alles Gemüse in der Scheune verkauft. Alternativ kann auch eine Liste des konkret zu verkaufenden Gemüses angegeben werden. Hierfür wird das zu verkaufende Gemüse in einer mit Leerzeichen separierten Liste angegeben. Als Bezeichner für das Gemüse wird **mushroom**, **carrot**, **tomato** und **salad** verwendet. Der erfolgreiche Verkauf des Gemüses wird je nach Anzahl des verkauften Gemüses mit einer Meldung im Schema **You have sold [Anzahl Gemüse] vegetable for [gewonnenes Gold] gold.** bzw. **You have sold [Anzahl Gemüse] vegetables for [gewonnenes Gold] gold.** quittiert. Auch der Verkauf von 0 Gemüse ist eine zulässige Aktion.

#### ➤ Beispielinteraktion

```
1  [...]
2  > sell mushroom carrot mushroom
3  You have sold 3 vegetables for 34 gold.
4  > sell all
5  You have sold 1 vegetable for 15 gold.
6
7  It is Mira's turn!
8  [...]
```

### A.2.9 Befehl: buy

Die Kinder können mit dem Geld, das sie besitzen entweder Gemüse oder Land kaufen.

**A.2.9.1 Gemüse** Mit der Eingabe des Befehls `buy vegetable [Gemüsename]` kann ein Kind eine Einheit des entsprechenden Gemüses kaufen. Als Namen für die Gemüsearten, werden die gleichen verwendet, wie für den Befehl `sell`. Das System quittiert einen erfolgreichen Kauf mit einer Ausgabe nach dem Schema: `You have bought a [Gemüsename] for [Preis in Gold] gold.`

➤ Beispielinteraktion

```
1 | [...]
2 | > buy vegetable tomato
3 | You have bought a tomato for 6 gold.
4 | [...]
```

**A.2.9.2 Kacheln** Mit der Eingabe des Befehls `buy land [x-Koordinate] [y-Koordinate]` kann ein Kind neues Land an der angegebenen Position kaufen. Das System quittiert einen erfolgreichen Kauf mit einer Ausgabe nach dem Schema: `You have bought a [Kacheltyp] for [Preis in Gold] gold.` Die Bezeichner für den Namen der Kacheln sind in Tabelle A.2 definiert.

➤ Beispielinteraktion

```
1 | [...]
2 | > buy land 0 2
3 | You have bought a Forest for 10 gold.
4 | > buy vegetable carrot
5 | You have bought a carrot for 2 gold.
6 |
7 | It is Mira's turn!
8 | [...]
```

### A.2.10 Befehl: harvest

Der `harvest`-Befehl wird zum Ernten des Gemüses einer Kachel verwendet. Er folgt dem Schema `harvest [x-Koordinate] [y-Koordinate] [Anzahl]`. Man kann nur von Kacheln ernten, auf denen die geforderte Anzahl an Gemüse auch vorhanden ist. Die erfolgreiche Ernte wird vom System je nach Anzahl des geernteten Gemüses nach einem der folgenden Schemata quittiert: `You have harvested 1 [Gemüsename (Singular)]`. oder `You have harvested [Anzahl] [Gemüsename (Plural)]`. Als Namen für die Gemüsearten, werden die gleichen verwendet, wie für den Befehl `sell` (Singular) und `show barn` (Plural).



### ➤ Beispielinteraktion (sichtbare Leerzeichen)

```

1  [...]
2  It is Mira's turn!
3  > show board
4  |Fi 2|
5  | T |
6  | 1/4 |
7  | G * | G * |
8  | C | B 4 |
9  | 2/2 | 0/2 |
10 > harvest -1 0 2
11 You have harvested 2 carrots.
12 > harvest 0 1 1
13 You have harvested 1 tomato.
14 [...]
```

### ➤ Beispielinteraktion

```

1  [...]
2  It is Mira's turn!
3  > show board
4  | Fi 2|
5  |  T |
6  | 1/4 |
7  | G * | G * |
8  |  C | B 4 |
9  | 2/2 | 0/2 |
10 > harvest -1 0 2
11 You have harvested 2 carrots.
12 > harvest 0 1 1
13 You have harvested 1 tomato.
14 [...]
```

## A.2.11 Befehl: plant

Der letzte Befehl des Systems dient dem Anpflanzen von neuem Gemüse. Das Anpflanzen funktioniert mit dem Befehlsschema: `plant [x-Koordinate] [y-Koordinate] [Gemüsename]` Als Namen für die Gemüsearten, werden die gleichen verwendet, wie für den Befehl `sell`. Falls der Befehl erfolgreich ist, also genau 1 Gemüse auf einer davor leeren Kachel angebaut wurde, wird nichts ausgegeben.

### ➤ Beispielinteraktion (sichtbare Leerzeichen)

```

1  [...]
2  It is Mira's turn!
3  > show board
4  |Fi*|
5  |   |
6  |0/4|
7  |G*|   |G*|
8  |   |B6|   |
9  |0/2|   |0/2|
10 > plant -1 0 carrot
11 > show board
12 |Fi*|
13 |   |
14 |0/4|
15 |G1|   |G*|
16 |C|   |B6|   |
17 |1/2|   |0/2|
18 [...]
```

### ➤ Beispielinteraktion

```

1  [...]
2  It is Mira's turn!
3  > show board
4  | Fi *|
5  |     |
6  | 0/4 |
7  | G * |   | G * |
8  |     | B 6 |   |
9  | 0/2 |   | 0/2 |
10 > plant -1 0 carrot
11 > show board
12 | Fi *|
13 |     |
14 | 0/4 |
15 | G 1 |   | G * |
16 |  C |   | B 6 |   |
17 | 1/2 |   | 0/2 |
18 [...]
```

## A.3 Beispielinteraktionen

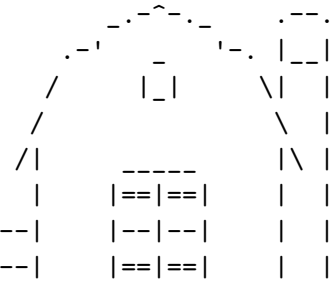
Sie finden die Beispielinteraktionen des Spiels auch als Textdatei im ILIAS.

### ➤ Beispielinteraktion

```

1  %> java QueensFarming
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

```



```

-----
~~~~~ QUEENS FARMING ~~~~~
-----

How many players?
> 3
Enter the name of player 1:
> Mira
Enter the name of player 2:
> Milan
Enter the name of player 3:
> Vincent
With how much gold should each player start?
> 42
With how much gold should a player win?
> 50
Please enter the seed used to shuffle the tiles:
> 6

It is Mira's turn!
> show barn
Barn (spoils in 6 turns)
carrots:      1
mushrooms:    1
salads:        1
tomatoes:     1
-----
Sum:           4

Gold:          42

```

## ➤ Beispielinteraktion

```

39  > show board
40      | Fi *|
41      |    |
42      | 0/4 |
43  | G * |    | G * |
44  |    | B 6 |    |
45  | 0/2 |    | 0/2 |
46  > show market
47  mushrooms: 16
48  carrots:    2
49  tomatoes:   6
50  salads:     4
51  > sell carrot
52  You have sold 1 vegetable for 2 gold.
53  > show barn
54  Barn (spoils in 6 turns)
55  mushrooms:  1
56  salads:     1
57  tomatoes:   1
58  -----
59  Sum:         3
60
61  Gold:        44
62  > plant 1 0 tomato
63
64  It is Milan's turn!
65  > end turn
66
67  It is Vincent's turn!
68  > end turn
69
70  It is Mira's turn!
71  > show board
72      | Fi *|
73      |    |
74      | 0/4 |
75  | G * |    | G 2 |
76  |    | B 5 | T  |
77  | 0/2 |    | 1/2 |
78  > end turn
79
80  It is Milan's turn!
81  > end turn
82
83  It is Vincent's turn!
84  > end turn
85
86  It is Mira's turn!

```

## ➤ Beispielinteraktion

```

87  > show board
88      | Fi *|
89      |    |
90      | 0/4 |
91  | G * |    | G 1 |
92  |    | B 4 | T  |
93  | 0/2 |    | 1/2 |
94  > end turn
95
96  It is Milan's turn!
97  > end turn
98
99  It is Vincent's turn!
100 > end turn
101
102 It is Mira's turn!
103 1 vegetable has grown since your last turn.
104 > show barn
105 Barn (spoils in 3 turns)
106 mushrooms: 1
107 salads:    1
108 -----
109 Sum:        2
110
111 Gold:       44
112 > harvest 1 0 2
113 You have harvested 2 tomatoes.
114 > show barn
115 Barn (spoils in 3 turns)
116 mushrooms: 1
117 salads:    1
118 tomatoes:  2
119 -----
120 Sum:        4
121
122 Gold:       44
123 > sell mushroom salad tomato tomato
124 You have sold 4 vegetables for 32 gold.
125
126 It is Milan's turn!
127 > end turn
128
129 It is Vincent's turn!
130 > end turn
131 Player 1 (Mira): 76
132 Player 2 (Milan): 42
133 Player 3 (Vincent): 42
134 Mira has won!

```