Karlsruher Institut für Technologie

Prof. Dr.-Ing. Anne Koziolek
Architecture-driven Requirements Engineering (ARE)
https://sdqweb.ipd.kit.edu/wiki/Programmieren
programmieren-vorlesung@ipd.kit.edu

Programmieren – Sommersemester 2020

Abschlussaufgabe 1 20 Punkte
Version 1

Ausgabe:	20.07.2020, ca. 13:00 Uhr
Praktomat:	03.08.2020, 13:00 Uhr
Abgabefrist:	18.08.2020, 06:00 Uhr

Abgabehinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe dieses Blattes notwendig ist. Der Praktomat wird Ihre Abgabe zurückweisen, falls eine der nachfolgenden Regeln verletzt ist. Eine zurückgewiesene Abgabe wird automatisch mit null Punkten bewertet. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

- Achten Sie auf fehlerfrei kompilierenden Programmcode.
- Verwenden Sie keine Elemente der Java-Bibliotheken, ausgenommen Elemente der Pakete `java.lang`, `java.util`, `java.util.regex`, `java.util.function` und `java.util.stream`.
- 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()` und `Runtime.exit()` dürfen nicht verwendet werden.
- Halten Sie die Regeln zur Javadoc-Dokumentation ein.
- Halten Sie auch alle anderen Checkstyle-Regeln an.

Bearbeitungshinweise

Diese Bearbeitungshinweise sind relevant für die Bewertung Ihrer Abgabe, jedoch wird der Praktomat Ihre Abgabe **nicht** zurückweisen, falls eine der nachfolgenden Regeln verletzt ist.

- 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.
- 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.
- Wählen Sie aussagekräftige Namen für alle Ihre Bezeichner.

Plagiat

Es werden nur selbstständig angefertigt 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 Quelltextschnipsel von den Vorlesungsfolien und aus den Lösungsvorschlägen des Übungsbetriebes in diesem Semester. Alle benutzten Hilfsmittel müssen vollständig und genau angegeben werden und alles, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, muss deutlich kenntlich gemacht werden. Ebenso stellt die Weitergabe einer Lösung oder von Teilen davon eine Störung des ordnungsgemäßen Ablaufs der Erfolgskontrolle dar. Dieser Ordnungsverstoß kann ebenfalls zum Ausschluss der Erfolgskontrolle führen. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

Checkstyle

Der Praktomat überprüft Ihre Quelltexte während der Abgabe automatisiert auf die Einhaltung der Checkstyle-Regeln. Es gibt speziell markierte Regeln, bei denen der Praktomat die Abgabe zurückweist, da diese Regel verpflichtend einzuhalten ist. 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 wird.

>_ Terminal-Klasse

Laden Sie für diese Aufgabe die Terminal-Klasse herunter und platzieren Sie diese unbedingt im Paket `edu.kit.informatik`. Die Methode `Terminal.readLine()` liest eine Benutzereingabe von der Konsole und ersetzt `System.in`. Die Methode `Terminal.printLine()` schreibt eine Ausgabe auf die Konsole und ersetzt `System.out`. Verwenden Sie für jegliche Konsoleneingabe oder Konsolenausgabe die Terminal-Klasse. Verwenden Sie in keinem Fall `System.in` oder `System.out`. Laden Sie die Terminal-Klasse niemals zusammen mit Ihrer Abgabe hoch.

Interaktive Benutzerschnittstelle

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 Text frei wählen, er sollte jedoch sinnvoll sein. Jede Fehlermeldung muss aber mit `Error ,` beginnen und darf keine Zeilenumbrüche enthalten.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Montag, den 03. August 2020 um 13:00 Uhr**, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Praktomat bei der richtigen Aufgabe vor Ablauf der Abgabefrist hochzuladen.

- Geben Sie Ihre Klassen zur Abschlussaufgabe 1 als `*.java`-Dateien ab.

Aufgabe A: Bäckers Walz (20 Punkte)

In dieser Aufgabe soll das Spiel *Bäckers Walz*[1] entworfen und implementiert werden. In diesem Spiel treten zwei bis vier Spieler gegeneinander an. Der Spieler, der als Erster einen Goldbetrag von 100 oder mehr erwirtschaftet hat, gewinnt das rundenbasierte Spiel.

A.1 Aufbau und Spielfeld

Das kreisförmige Spielfeld – beginnend mit einem Startfeld – wird immer vor Spielbeginn festgelegt. Die Felder des Spielfelds werden anhand von Regeln angeordnet und können in der Anzahl 4 bis 25 variieren. Diese Ordnung und Anzahl kann für jedes Spiel unterschiedlich sein (siehe A.5).

Die unterschiedlichen Spielfelder werden im Folgenden aufgelistet:

- ein Startfeld (`S`) sowie
- jeweils ein bis acht Felder – auf denen ggf. jeweils bestimmte Rohstoffe hergestellt und an den Markt verkauft werden können – für die Repräsentation:
  - einer Mühle (`M`) mit Rohstoff *Mehl* (`flour`),
  - eines Hühnerstalls (`H`) mit Rohstoff *Eier* (`egg`),
  - Kuhweide (`C`) mit Rohstoff *Milch* (`milk`).

[1] `https://de.wikipedia.org/wiki/Wanderjahre`

Es ist ebenso ein zentraler Markt vorhanden, auf dem der aktive Spieler die o.g. Rohstoffe kaufen und verkaufen kann. Zu Beginn des Spiels ist der Markt mit jeweils zwei Rohstoffen jeder Art bestückt (siehe A.4, Tabelle 0.2).

A.2 Spielzug und -ablauf

Die Abbildung 0.1 illustriert graphisch mögliche Aktionen innerhalb eines Spielzugs eines aktiven Spielers.

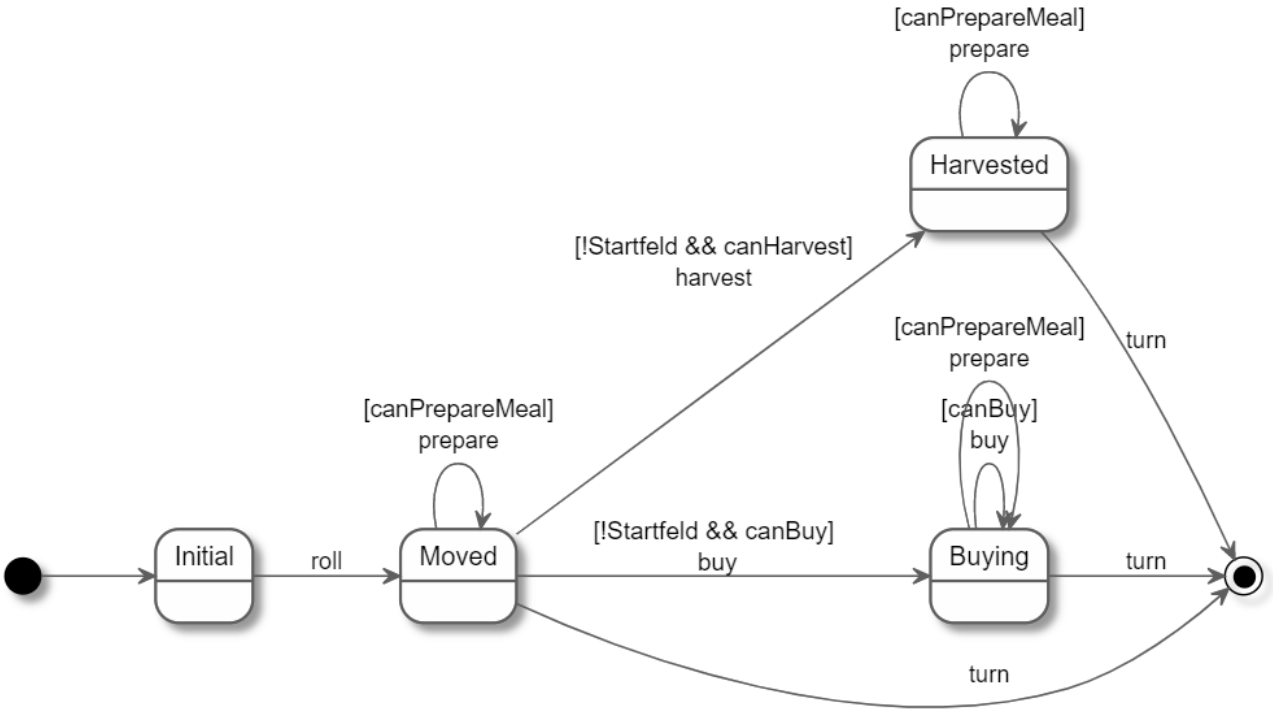

Abbildung 0.1: Mögliche Spielzüge eines aktiven Spielers

Jeder Spieler besitzt automatisch zu Spielbeginn 20 Goldstücke. Der Goldvorrat eines Spielers wird durch eine natürliche 32-Bit-Ganzzahl größer gleich Null repräsentiert.

Bei jedem Spielstart beginnt immer der erste Spieler, der im Folgenden als `P1` bezeichnet wird. Hat ein Spieler seine Aktionen in einem Spielzug vollendet, so ist der nächste Spieler an der Reihe. Nachdem der letzte Spieler seinen Spielzug vollendet hat, ist wieder der erste Spieler an der Reihe. Dieses rundenbasierte Spielprinzip wird solange vollzogen, bis einer der Spieler das Spiel gewonnen hat (siehe A.6/Spielende).

Während des Spiels würfelt im ersten Schritt der aktive Spieler und zieht auf ein neues Feld. Über die Anzahl der zu ziehenden Felder pro Spielzug entscheidet die Augenzahl des sechsseitigen Würfels. Es besteht Zugpflicht. Spieler dürfen nur in eine Richtung nach vorne ziehen. Auf einem Spielfeld (`M`, `H`,`C` oder `S`) können sich gleichzeitig mehr als zwei und maximal vier Spieler befinden (vgl. A.7). Spieler landen nach einem Würfelwurf entweder auf einem der Spielfelder `M`, `H` oder `C` oder auf dem Startfeld (`S`).
Landet der aktive Spieler auf einem der Spielfelder `M`, `H` oder `C`, kann der aktive Spieler im

darauffolgenden Schritt entweder (i) einen Rohstoff – abhängig vom Spielfeld auf dem er sich befindet – herstellen und ihn an den Markt verkaufen (siehe A.4/Verkaufen und A.6.1/Der `harvest`-Befehl) oder (ii) einen Rohstoff zum aktuellen Preis vom Markt kaufen (siehe A.4/Kaufen und A.6.1/Der `buy`-Befehl) oder (iii) Gerichte zubereiten (siehe A.6.1/Der `prepare`-Befehl). Der aktive Spieler kann im gesamten Spielzug nur einmalig einen Rohstoff herstellen und verkaufen. Das Herstellen eines Rohstoffs und dessen Verkauf an den Markt ist eine atomare Aktion (siehe A.6.1/Der `harvest`-Befehl). Das Kaufen von Rohstoffen ist in einem Spielzug mehrmals erlaubt (siehe Abbildung 0.1). Landet der aktive Spieler auf dem Startfeld (`S`), erhält er 5 Goldstücke.

Nach dem Würfeln und Vorrücken auf dem Spielfeld kann der aktive Spieler (auch vor dem Verkaufen oder Kaufen von Rohstoffen) beliebig oft Gerichte (siehe A.3) zubereiten, sofern er genügend Rohstoffe in seinem gekauften Vorrat hat. Ein Gericht kann nach einem der Rezepte hergestellt werden. Für die Zubereitung eines Gerichts werden aus dem Vorrat des aktiven Spielers die Rohstoffe entnommen, er erhält dafür die jeweilige Anzahl an Goldstücken (siehe Tabelle 0.1). In einem gesamten Spielzug kann der aktive Spieler beliebig oft Gerichte zubereiten, sofern passende Rohstoffe in seinem Vorrat vorhanden sind.

Ein Spielzug wird durch den aktiven Spieler mit dem Befehl `turn` (siehe A.6.1/Der `turn`-Befehl) beendet.

A.3 Rezepte

Die Rezepte geben an, welche Rohstoffe ein Spieler braucht, um ein Gericht zuzubereiten. Ebenso erhält jeder Spieler für die Herstellung einen Gewinn, der in Anzahl an Goldstücken angegeben wird (siehe Tabelle 0.1). Die verwendeten Rohstoffe (Mehl (`flour`), Eier (`egg`), Milch (`milk`)) stammen hierfür ausschließlich aus dem gekauften Vorrat (siehe A.4/Kaufen) des Spielers.

Gericht	Mehl	Eier	Milch	Gewinn
Joghurt (`yoghurt`)			3	8
Baiser (`meringue`)		3		9
Brot (`bread`)	3			10
Milchbrötchen (`bun`)	2		1	11
Crépe (`crepe`)	1	2		12
Pudding (`pudding`)		1	2	13
Kuchen (`cake`)	2	2	2	22

Tabelle 0.1: Überblick über die Rezepte

Jeder Spieler, der alle 7 Gerichte zubereitet hat, erhält einen Meisterbrief und einmalig 25 Gold.

A.4 Markt

Der Markt bestimmt die Preise jedes der drei Rohstoffe (Mehl (`flour`), Eier (`egg`), Milch (`milk`)). In Tabelle 0.2 wird die Startbelegung des Marktes zu Spielbeginn dargestellt. Der aktive Spieler

kann nur den Rohstoff ~~kaufen oder~~ verkaufen, auf dessen zugehörigen Spielfeld er nach dem Würfeln gelandet ist. D.h. Mehl auf Mühle, Eier auf Hühnerstall, Milch auf Kuhweide. Es können 0 bis 5 Einheiten eines jeden Rohstoffs auf dem Markt existieren.

Aktueller Preis	Mehl	Eier	Milch
1 Goldstück			
2 Goldstücke			
3 Goldstücke			
4 Goldstücke	Mehl	Ei	Milch
5 Goldstücke	Mehl	Ei	Milch

Tabelle 0.2: Startbelegung des Marktes

Verkaufen

Hat der aktive Spieler einen Rohstoff hergestellt und möchte ihn an dem Markt verkaufen, legt er diesen auf den Markt auf das nächste freie Feld in der zum Rohstoff gehörenden Spalte (siehe Tabelle 0.2). Dabei werden die Rohstoffe von unten nach oben angelegt. Wird in der obigen Tabelle nun eine Milch auf dem Spielfeld Kuhweide hergestellt und an der Markt verkauft, wird diese auf die 3. Zeile der Milch-Spalte gelegt. Der Verkauf eines Rohstoffs bringt dem aktiven Spieler ein Goldstück als Gewinn. Die Herstellung eines Rohstoffs und der Verkauf an den Markt stellt eine atomare Aktion dar (siehe A.6.1/Der `harvest`-Befehl). Spieler können nur einmal pro Spielzug einen Rohstoff herstellen und nur dann, wenn sie im gesamten Spielzug keinen Rohstoff kaufen (vgl. Abbildung 0.1).

Kaufen

Kauft der aktive Spieler einen Rohstoff vom Markt, wird der Preis zum Kauf des Rohstoffs durch den *aktuellen Preis* (vgl. Tabelle 0.2) der Zeile des obersten, zugehörigen Rohstoffs bestimmt. Sofern der aktive Spieler genügend Goldstücke besitzt und der Rohstoff auf dem Markt existiert, kann er den Rohstoff zum aktuellen Preis kaufen. Dabei werden die gekauften Rohstoffe von oben nach unten vom Markt genommen. Beispiel: Wird in der obigen Tabelle 0.2 eine Milch gekauft, liegt die oberste Milch in Zeile 4 und kostet deswegen 4 Goldstücke. Anschließend kostet die nächste gekaufte Milch 5 Goldstücke.
Der aktive Spieler kann nur Rohstoffe kaufen, wenn er im selben Zug keine Rohstoffe hergestellt und verkauft hat; dann jedoch so viele wie es auf dem Markt gibt und er diese auch aus seinem Goldvorrat bezahlen kann.
Das Kaufen von Rohstoffen erlaubt auf deren Basis das Zubereiten von Gerichten nach Rezept (siehe A.3).

A.5 Kommandozeilenargumente

Ihr Programm erwartet Kommandozeilenargumente.

Das erste Kommandozeilenargument definiert die Anzahl der Spieler x $\in \{2, 3, 4\}$.

Anschließend wird, beginnend mit dem Startfeld `S`, die Reihenfolge der Spielfelder für Mühle `M`, Hühnerstall `H` und Kuhweide `C` festgelegt. Alle Spielfelder werden durch exakt ein Semikolon separiert. So werden die Felder für den Spielverlauf in einem Kreis angeordnet ausgehend vom Startfeld.

Halten Sie dabei für die Reihenfolge der Spielfelder folgende Regeln ein:

- Das erste Feld muss das Startfeld `S` sein. Es gibt nur ein Startfeld.
- Es muss zusätzlich mindestens eine Mühle `M`, einen Hühnerstall `H` und eine Kuhweide `C` geben.
- Es dürfen keine zwei angrenzenden Felder gleich sein. Das Startfeld `S` wird dabei übersprungen.
  Beispiel: Sequenz `M;M` ist verboten, `M;C;M` ist erlaubt.
- Vier-Felder-Regel: Jedes Feld darf nicht mehr als 4 Felder von einem Feld der gleichen Art entfernt sein. Das Startfeld `S` wird dabei übersprungen.
  Beispiel: Sequenz `M;C;H;C;H;M` ist verboten, `M;C;H;C;M` ist erlaubt.

Beispiele für gültige Zusammenstellungen kreisförmiger Spielfelder:

- `S;M;C;H;C;M;C;H`
- `S;M;C;H`

Beispiele für ungültige Zusammenstellungen kreisförmiger Spielfelder:

- `S;M;H;M;H` (Kuhweide `C` fehlt)
- `S;M;C;H;H` (zwei Hühnerställe `H` nebeneinander)
- `S;H;M;C;H` (zwei Hühnerställe `H` nebeneinander)
- `S;M;C;H;C;H` (Vier-Felder-Regel verletzt)
- `S;H;M;C;H;C` (Vier-Felder-Regel verletzt)

Beispiel `java BakersWalz 4 S;M;C;H;C;M;C;H`

Der Programmname `BakersWalz` ist nicht vorgeschrieben.
Tritt beim Verarbeiten des Kommandozeilenargumente ein Fehler auf bzw. entspricht die Eingabe nicht dem hier spezifizierten Format, so wird eine aussagekräftige Fehlermeldung beginnend mit `Error ,␣` ausgegeben und das Programm wird beendet.

A.6 Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole mittels `Terminal.readLine()` Befehle entgegen, die im Folgenden näher spezifiziert werden. Nach Abarbeitung eines Befehls wartet Ihr Programm auf weitere Befehle, bis das Programm irgendwann durch die Eingabe der Zeichenfolge `quit` beendet wird.

Fehlermeldungen

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Spielregeln nicht verletzt werden und geben Sie in diesen Fällen eine aussagekräftige Fehlermeldung aus. Auch wenn die Benutzereingabe nicht dem vorgegebenen Format entspricht, ist 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 `Error,␣` beginnen und darf keine Zeilenumbrüche enthalten. Den weiteren Text der Fehlermeldung dürfen Sie frei wählen, er sollte jedoch sinnvoll sein.

Automatische Tests

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. Geben Sie auch keine zusätzlichen Informationen aus. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären.

Beispielinteraktionen

Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebenen Beispielinteraktionen. Die Eingabezeilen werden mit dem `>` (Größer-als-Zeichen) gefolgt von einem Leerzeichen eingeleitet, diese beiden Zeichen sind ebenfalls kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabezeilen. Beachten Sie, dass die Beispielabläufe der einzelnen Befehle unabhängig voneinander zu betrachten sind.

Platzhalter

Beachten Sie, dass bei der Beschreibung der Eingabe- und Ausgabeformate die Wörter zwischen spitzen Klammen (`<` und `>`) für Platzhalter stehen, welche bei der konkreten Ein- und Ausgabe durch Werte ersetzt werden. Diese eigentlichen Werte enthalten bei der Ein- und Ausgabe keine spitzen Klammern. Vergleichen Sie hierzu auch die jeweiligen Beispielabläufe.

Spielende

Der Spieler, der als Erster einen Goldbetrag von 100 oder mehr erwirtschaftet hat, gewinnt das Spiel *Bäckers Walz*. In diesem Fall wird zusätzlich zu der Ausgabe des aktuellen Befehls `Px wins` mit $x \in \{1, 2, 3, 4\}$ in einer separaten Zeile ausgegeben. Ihr Programm terminiert hierbei nicht. Die Befehle `show-market` und `show-player` sind weiterhin zulässig, die weiteren Befehle `roll`, `harvest`, `buy`, `prepare`, `can-prepare?` und `turn` jedoch nicht. Der Befehl `quit` beendet das Programm vollständig.

A.6.1 Befehle

Der `roll`-Befehl

Mit diesem Befehl kann der aktive Spieler im Rahmen der Regeln eine Zahl `<number>` übergeben, die als das Ergebnis eines Würfelwurfs interpretiert wird.
`<number>` ist in diesem Fall eine natürliche Zahl aus dem abgeschlossenen Intervall [1,6].

Eingabe `roll <number>`

Ausgabe Die Ausgabe des Befehls ist im Erfolgsfall das Kürzel des Spielfelds (`S`, `M`, `H` oder `C`) auf dem der Spieler nun gelandet ist, gefolgt von seinem aktuellen Goldvorrat. Beide Ausgabe werden durch exakt ein Semikolon separiert.

Beispielinteraktion

```
1 > roll 4
2 S;34
```

Der `harvest`-Befehl

Mit dem parameterlosen `harvest`-Befehl kann der aktive Spieler einen Rohstoff herstellen und an dem Markt verkaufen. Es kann nur der Rohstoff `<resource>` hergestellt werden, der dem Spielfeld zugehörig ist und auf dem sich der Spieler nach einem Würfelwurf aktuell befindet.

Eingabe `harvest`

Ausgabeformat `<resource>;<gold>` Im Erfolgsfall wird der verkaufte Rohstoff (`flour`, `egg` oder `milk`) und der aktuelle Goldvorrat des Spielers ausgegeben. Beide Ausgaben werden durch exakt ein Semikolon separiert. Im Fehlerfall (z.B. wenn bereits das Maximum an Rohstoffeinheiten erreicht ist, siehe A.4) wird eine aussagekräftige Fehlermeldung beginnend mit `Error␣␣` ausgegeben.

Beispielinteraktion

```
1 > roll 5
2 H;24
3 > harvest
4 egg;25
```

Der `buy`-Befehl

Mit dem `buy`-Befehl kann der aktive Spieler Rohstoffe vom Markt kaufen, sofern er genügend Goldstücke zur Verfügung hat.

Eingabe `buy <resource>`

Ausgabeformat `<price>;<gold>` Im Erfolgsfall wird der aktuelle Preis des Rohstoff ausgegeben sowie der aktuelle Goldvorrat des Spielers nach dem Kauf des zugehörigen Rohstoffs.

Beispielinteraktion

```
1 > buy egg
2 4;22
3 > buy egg
4 5;17
5 > buy egg
6 Error, no eggs on market available
```

Der `prepare`-Befehl

Mit dem `prepare`-Befehl kann der aktive Spieler ein Gericht `<recipe>` (`yoghurt`, `meringue`, `bread`, `bun`, `crepe`, `pudding` oder `cake`) zubereiten, sofern er genügend Rohstoffe zur Verfügung hat. Sein Goldvorrat wird mit dem jeweiligen Gewinn aufgestockt.

Eingabe `prepare <recipe>`

Ausgabe `<gold>` Im Erfolgsfall wird der aktuelle Goldvorrat des Spielers ausgegeben. Beachten Sie die Besonderheit bei dem Erhalt eines Meisterbriefs (siehe A.3).
Die dafür verwendeten Rohstoffe werden ihm aus seinem Rohstoffvorrat entnommen.

Beispielinteraktion

```
1 > prepare yoghurt
2 87
3 > prepare cake
4 109
5 P1 wins
```

Der `can-prepare?`-Befehl

Mit dem `can-prepare?`-Befehl kann der aktive Spieler feststellen, welche Gerichte er basierend auf seinem Rohstoffvorrat zubereiten kann.

Eingabe `can-prepare?`

Ausgabe Im Erfolgsfall werden alle Gerichte namentlich (`yoghurt`, `meringue`, `bread`, `bun`, `crepe`, `pudding` oder `cake`), die der aktive Spieler zubereiten kann – aufsteigend sortiert nach dem möglichen Gewinn – zeilenweise ausgegeben. Falls nichts zubereitet werden kann, findet keine Ausgabe statt.

Beispielinteraktion

```
> can-prepare?
bun
crepe
pudding
cake
```

Der `show-market`-Befehl

Der parameterlose `show-market`-Befehl listet die drei Rohstoffe aufsteigend sortiert nach deren Vorkommen auf dem Markt auf. Bei gleicher Anzahl werden die Rohstoffe alphabetisch nach ihrer englischen Bezeichnung sortiert.

Eingabe `show-market`

Ausgabe `<number>;<resource>` Im Erfolgsfall werden die Rohstoffe zeilenweise mit der vorhandenen Anzahl `<number>` als natürliche Zahl im abgeschlossenen Intervall [0,5] und mit den Bezeichnern ausgegeben. Alle Ausgaben in einer Zeile werden durch exakt ein Semikolon separiert.

Beispielinteraktion

```
> show-market
2;egg
3;milk
4;flour
```

Der `show-player`-Befehl

Der `show-player`-Befehl gibt den Vorrat an Goldstücken und Rohstoffen eines Spielers `Px` mit $x \in \{1, 2, 3, 4\}$ aus.

Eingabe `show-player <Px>`

Ausgabe `<gold>;<flour>;<egg>;<milk>` Im Erfolgsfall werden zunächst die Anzahl der Goldstücke, dann die Anzahl der Rohstoffe Mehl, Eier und Milch eines Spielers `Px` in einer Zeile ausgegeben. Alle Ausgaben in einer Zeile werden durch exakt ein Semikolon separiert.

Beispielinteraktion

```
> show-player 1
Error, incorrect input format
> show-player P1
25;2;0;2
```

Der `turn`-Befehl

Der parameterlose `turn`-Befehl schließt den Spielzug des aktiven Spielers ab und wechselt anschließend den aktiven Spieler. Dieser Befehl ist nur zulässig, nachdem der bisher aktive Spieler seinen Spielzug vollständig abgeschlossen hat.

Eingabe `turn`

Ausgabe Im Erfolgsfall wird `Px` ausgegeben, falls Spieler `Px` mit $x \in \{1,2,3,4\}$ der neue aktive Spieler ist. Beispiel: `P2` wird ausgegeben, falls Spieler 1 seinen Spielzug mit dem `turn`-Befehl abgeschlossen hat. D.h. im nächsten Spielzug ist Spieler 2 der aktive Spieler.

Beispielinteraktion

```
1 > roll 5
2 H;24
3 > harvest
4 egg;25
5 > prepare cake
6 47
7 > turn
8 P2
```

Der `quit`-Befehl

Der parameterlose `quit`-Befehl ermöglicht es, Ihr Programm jederzeit vollständig zu beenden. Beachten Sie, dass hierfür keine Methoden wie `System.exit()` oder `Runtime.exit()` verwendet werden dürfen.

Eingabe `quit`

Ausgabe Im Erfolgsfall findet keine Ausgabe statt. Im Fehlerfall wird eine aussagekräftige Fehlermeldung beginnend mit `Error,␣` ausgegeben.

Beispielinteraktion

```
1 > quit quit
2 Error, incorrect input format, this command does not accept any parameters
3 > quit
```

A.7 Beispielinteraktion

Im Folgenden finden Sie einen Beispielablauf einer Spielrunde mit 3 Spielern und einem Spielfeld mit der Konfiguration `S;M;C;H`. Es werden hierbei 3 Spielzüge vollständig durchgeführt.

Beispielinteraktion

```
> roll 2
C;20
> harvest
milk;21
> turn
P2
> roll 2
C;20
> buy milk
3;17
> buy egg
4;13
> buy milk
4;9
> prepare pudding
22
> turn
P3
> roll 4
S;25
> turn
P1
> show-player P1
21;0;0;0
> show-player P2
22;0;0;0
> show-player P3
25;0;0;0
> show-market
1;egg
1;milk
2;flour
> quit
```