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 – Wintersemester 2020/21

Übungsblatt 5 Version 1.1	20 Punkte	Ausgabe: 13.01.2021, ca. 08:00 Uhr Praktomat: 20.01.2021, 12:00 Uhr Abgabefrist: 28.01.2021, 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 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 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.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Mittwoch, den 20. Januar 2021 um 13:00 Uhr**, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Praktomat bei der richtigen Aufgabe vor Ablauf der Abgabefrist hochzuladen. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären.

- Geben Sie Ihre Klassen zu Aufgabe A in Einzelarbeit als `*.java`-Dateien ab.

Aufgabe A: Santorini (20 Punkte)

Das Spiel *Santorini* findet auf einem 3-dimensionalen Spielfeld statt. Zwei Spieler, die gegeneinander antreten, stapeln auf einem quadratischen Spielfeld abwechselnd Spielsteine aufeinander, mit dem Ziel einen Turm der Höhe 3 zu errichten, um diesen mit der eigenen Spielfigur zu erklimmen. Dies veranschaulicht Abbildung 0.1. Es gewinnt der Spieler, der mit seiner Spielfigur als erster einen Turm der Höhe 3 erklimmt oder wenn sein Gegner nicht mehr seine Spielfigur ziehen und/oder Türme errichten kann. Dabei gelten einige Einschränkungen – sowohl beim Turmbau als auch beim Ziehen der Spielfigur.

Implementieren Sie Santorini wie nachfolgend beschrieben.

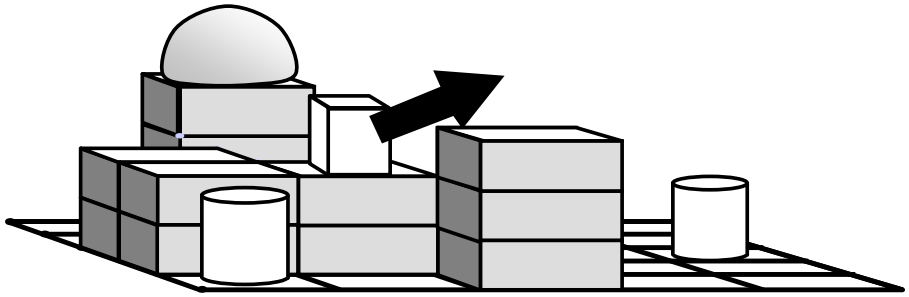

Abbildung 0.1: Santorini-Spielfeld, kurz vor dem Sieg von Spieler 1. Der Pfeil deutet den siegbringenden Spielzug an. Spieler 1 spielt hier mit Figuren in Form eines Würfels, Spielfiguren von Spieler 2 haben die Form eines Zylinders. (Quelle: Gordon Hamilton, *Santorini Instructions*)

A.1 Spielaufbau

Das *Spielfeld* ist quadratisch und besteht aus 5×5 Zellen.

Eine Zelle c_{ij} besitzt 8 *angrenzende Zellen* – wie Abbildung 0.2 zeigt. Zellen am Spielfeldrand besitzen entsprechend weniger angrenzende Zellen. Dabei bezeichnet i die Zeilennummer und j die Spaltennummer einer Zelle. Zeilennummer und Spaltennummer beginnen jeweils bei 0. Somit ist c_{00} die oberste linke Zelle.

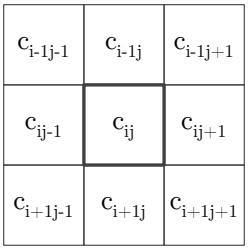

Abbildung 0.2: Eine Zelle c_{ij} samt ihrer angrenzenden Zellen

Jeder der beiden Spieler besitzt zwei *Spielfiguren*, die bei Spielbeginn auf unterschiedlichen Zellen platziert werden.

Auf einer Zelle können *Bauelemente* platziert werden. Es gibt zwei Arten von Bauelementen: Quader und Kuppeln. Es gibt im Spiel insgesamt 54 Quader und 18 Kuppeln. *Quader* („cuboid“) werden aufeinander gestapelt, um Türme zu errichten. Drei aufeinander gestapelte Quader bilden einen *Turm*. Eine *Kuppel* („dome“) kann auf einem Turm platziert werden, um zu verhindern, dass eine Spielfigur den Turm erklimmt.

A.2 Spielregeln

Das Spiel beginnt, indem die Spieler ihre Spielfiguren auf unterschiedlichen Zellen platzieren (siehe Abschnitt A.3).

Spieler 1 und Spieler 2 führen nun abwechselnd Spielzüge aus. Es beginnt Spieler 1. Ein Spielzug besteht aus aufeinanderfolgenden Schritten.

1. **Götterkarte ziehen** (optional) mit weiteren Anweisungen zu Spezialfähigkeiten und Sonderaktionen für den aktuellen Spielzug. Dabei treten für die darauffolgenden Schritte die Kartenregeln in Kraft (Abschnitt A.2.1).
2. **Auswahl** einer (eigenen) Spielfigur und **Ziehen** der ausgewählten Spielfigur auf eine angrenzende, unbesetzte Zelle (nachfolgend „Zielzelle“ genannt). Hierbei kann der Spieler in der Regel beliebig viele Ebenen nach unten springen, aber nur eine Ebene nach oben auf einen Quader klettern. Dabei müssen die Zugregeln (Abschnitt A.2.2) beachtet werden.
3. **Bauen** eines Spielsteins auf eine an die Zielzelle angrenzende, unbesetzte Zelle unabhängig von der Ebene auf der sich die Spielfigur befindet. Dabei müssen die Bauregeln (Abschnitt A.2.3) beachtet werden.

Ein Spieler muss in jedem Spielzug mindestens sowohl eine Spielfigur ziehen als auch bauen. Das Ziehen einer Götterkarte zu Beginn eines Spielzuges ermöglicht dem Spieler gewisse Spezialfähigkeiten (Abschnitt A.2.1). In einem Spiel darf ein Spieler höchstens dreimal eine Götterkarte ziehen. Es darf nicht ausgesetzt werden. Es gewinnt der Spieler, der mit seiner Spielfigur als erster einen Turm erklimmt oder wenn sein Gegner nicht mehr seine Spielfigur ziehen und/oder Türme errichten kann. Das Programm wird danach beendet.

A.2.1 Götterkartenregeln

Ein Spieler kann zu Beginn eines Spielzugs entscheiden, ob er eine Götterkarte ziehen möchte. Innerhalb eines Spiels kann er maximal drei Götterkarten ziehen, die ihm Spezialfähigkeiten und Sonderaktionen erlauben, die im Folgenden erläutert werden:

- Götterkarte `Apollo`: Mit dieser gezogenen Götterkarte kann ein Spieler auf das mit einer fremden Spielfigur besetzte Feld ziehen. Die Spielfigur des Gegners wird auf das eigene Ausgangsfeld gesetzt.
- Götterkarte `Artemis`: Mit dieser gezogenen Götterkarte kann ein Spieler seine Spielfiguren zweimal bewegen. ~~seine Spielfigur zweimal bewegen, aber nicht auf das Ausgangsfeld zurück.~~
- Götterkarte `Athena`: Mit dieser gezogenen Götterkarte kann ein Spieler dafür sorgen – sofern er die eigene Spielfigur in dem Zug nach oben bewegt hat – dass der Gegner im darauffolgenden Zug seine Spielfigur nicht nach oben bewegen kann.
- Götterkarte `Atlas`: Mit dieser gezogenen Götterkarte kann ein Spieler die Kuppel auf eine beliebige Ebene bauen. Dies kann den Gegner in seinen darauffolgende Bewegungen stark einschränken.
- Götterkarte `Demeter`: Mit dieser gezogenen Götterkarte kann ein Spieler in seiner Runde zweimal bauen. ~~, aber nicht auf den gleichen zu bauenden Turm (bzw. auf die gleiche Zelle).~~
- Götterkarte `Hermes`: Dieser gezogene Götterkarte ermöglicht einem Spieler sich beliebig weit mit seiner Spielfigur zu teleportieren – solange sich die Spielfigur auf einer Ebene bewegt (also nicht nach oben oder unten ziehen) – bevor gebaut wird.

A.2.2 Zugregeln

Eine Spielfigur darf beim Ziehen beliebig viele Stufen nach unten springen, aber höchstens eine Stufe nach oben klettern. Steht die Spielfigur z.B. auf einer Zelle mit zwei gestapelten Quadern, darf sie auf eine Nachbarzelle ziehen (herunterspringen), obwohl sich auf der Zielzelle kein einziger Quader befindet. Umgekehrt darf die Spielfigur die Zelle mit den aufeinander gestapelten Quader nicht betreten, wenn sie selbst direkt auf dem Spielfeld steht und somit zwei Stufen hochklettern müsste.

Eine Zelle mit einer Kuppel darf nicht betreten werden.

Eine Spielfigur „besetzt“ die Zelle, auf der sie platziert wurde. Eine besetzte Zelle darf nicht durch andere Spielfiguren betreten werden. Bauelemente hingegen besetzen eine Zelle nicht.

A.2.3 Bauregeln

Es dürfen maximal drei Quader aufeinander gestapelt werden. Bei genau drei aufeinander gestapelten Quadern sprechen wir von einem Turm. Auf jedem Turm darf genau eine Kuppel errichtet werden (Ausnahme Turmhöhe bei Götterkarte `Atlas`). Über einer Kuppel dürfen keine weiteren Bauelelemente platziert werden. Auf einer besetzten Zelle darf nicht gebaut werden.

A.3 Kommandozeilenargumente

Beim Programmstart werden alle vier Spielfiguren mit einem Namen versehen und auf dem Spielfeld positioniert. Hierzu erwartet Ihr Programm genau vier Kommandozeilenargumente, eines je Spielfigur. Das Format jedes Kommandozeilenarguments ist weiter unten beschrieben. Die ersten beiden Kommandozeilenargumente beschreiben Spielfiguren von Spieler 1. Die letzten beiden Kommandozeilenargumente beschreiben Spielfiguren von Spieler 2.

Tritt beim Verarbeiten der Kommandozeilenargumente ein Fehler auf, so wird eine aussagekräftige Fehlermeldung ausgegeben und das Programm wird beendet.

Eingabeformat `<Spielfigur-Bezeichner>;<Zeilennummer>;<Spaltennummer>`

`<Spielfigur-Bezeichner>` ist ein String, der durch den regulären Ausdruck[1] `[0-9a-z]+` beschrieben ist. Der Spielfigur-Bezeichner muss eine Spielfigur eindeutig bezeichnen. `<Zeilennummer>` und `<Spaltennummer>` sind jeweils ein Integer und bezeichnen die Zelle, an der die erzeugte Spielfigur platziert wird.

Beispiel `java Santorini yellow;1;1␣red;3;2␣blue;1;2␣green;2;2`

Der Programmname *Santorini* ist nicht vorgeschrieben.

A.4 Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole mittels `Terminal.readLine()` Befehle entgegen. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm irgendwann durch `quit` beendet wird oder ein Spieler das Spiel gewonnen hat.

Direkt nach Programmstart ist Spieler 1 aktiv.

Fehlermeldungen

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Vorgaben in der Aufgabenstellung (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.

[1] siehe `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html`

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 Ausgabe 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 Eingabe und Ausgabe keine spitzen Klammern. Vergleichen Sie hierzu auch die jeweiligen Beispielabläufe.

A.4.1 Der `draw-card`-Befehl

Mit diesem Befehl kann der Spieler im Rahmen der Regeln ein Symbol übergeben, das als das Ergebnis des Kartenziehens interpretiert wird. Ein Spieler kann nur bei Beginn eines Spielzuges eine Götterkarte ziehen. Nach den oben genannten Regeln, ermöglicht einem Spieler das Ziehen einer Götterkarte Spezialfähigkeiten und Sonderaktionen im Vergleich zu einem herkömmlichen Spielzug mit Auswahl und Ziehen der eigenen Spielfigur sowie Bauen eines Spielsteins. Im gesamten Spiel darf er höchstens dreimal eine Götterkarte ziehen. Sobald eine Karte von einem Spieler gezogen wurde, wird sie aus dem Kartenvorrat entfernt.

Als Eingabeparameter sind hier die Kartensymbole `Apollo`, `Artemis`, `Athena`, `Atlas`, `Demeter` und `Hermes` erlaubt.

Eingabeformat `draw-card <Kartensymbol>`

Ausgabeformat Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall, d.h. bei ungültigen Eingaben, wird eine aussagekräftige Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.4.2 Der `move`-Befehl

Der `move`-Befehl bewegt eine Spielfigur des aktiven Spielers auf die angegebene Zelle.

Eingabeformat `move <Spielfigur-Bezeichner>;<Zeilennummer>;<Spaltennummer>`

`<Spielfigur-Bezeichner>` ist ein String, der die zu bewegende Spielfigur bezeichnet. `<Zeilennummer>` und `<Spaltennummer>` sind jeweils ein Integer und bezeichnen die Zelle, zu der die Spielfigur bewegt werden soll.

Ausgabeformat Wenn der Spielzug erfolgreich durchgeführt wurde, aber noch kein Spieler gewinnt, wird `OK` ausgegeben. Falls dieser Spielzug zum Sieg von Spieler 1 führt, wird `P1 wins` ausgegeben (`P2 wins`, falls Spieler 2 gewinnt) und das Programm beendet sich.

A.4.3 Der `build`-Befehl

Der `build`-Befehl legt einen bestimmten Spielstein zuoberst auf der angegebene Zelle. Der Spielstein wird folglich aus den Spielsteinvorrat entfernt.

Eingabeformat `build <Spielstein-Typ>;<Zeilennummer>;<Spaltennummer>`

`<Spielstein-Typ>` ist entweder `C` (für „cuboid“) oder `D` (für „dome“). `<Zeilennummer>` und `<Spaltennummer>` sind jeweils ein Integer und bezeichnen die Zelle, auf welcher der erzeugte Spielstein platziert werden soll.

Ausgabeformat Wenn der Spielzug erfolgreich durchgeführt wurde, aber noch kein Spieler gewinnt, wird `OK` ausgegeben. Falls dieser Spielzug zum Sieg von Spieler 1 führt, wird `P1 wins` ausgegeben (`P2 wins`, falls Spieler 2 gewinnt) und das Programm beendet sich. Eine Gewinnsituation ergibt sich, wenn z.B. in diesem Spielzug der letzte Quader platziert wurde.

A.4.4 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.

Eingabeformat `turn`

Ausgabeformat Im Erfolgsfall wird `P1` ausgegeben, wenn Spieler 1 der neue aktive Spieler ist; oder `P2`, falls Spieler 2 der neue aktive Spieler ist.

A.4.5 Der `surrender`-Befehl

Der parameterlose `surrender`-Befehl ermöglicht einem aktiven Spieler zu kapitulieren, wenn z.B. keine gültigen `move`- oder `build`-Befehle möglich sind. Das Programm wird anschließend beendet.

Eingabeformat `surrender`

Ausgabeformat Im Erfolgsfall wird der Gewinner mit `Px wins` mit $x \in \{1, 2\}$ ausgegeben.

A.4.6 Der `bag`-Befehl

Der parameterlose `bag`-Befehl gibt den aktuellen Spielsteinvorrat aus (d.h. Bauelemente, die sich aktuell nicht auf den Spielfeld befinden). Zunächst wird die Anzahl der Quader, dann die Anzahl der Kuppeln ausgegeben. Die Ausgabe erfolgt jeweils zeilenweise.

Eingabeformat `bag`

Ausgabeformat `<Spielstein-Typ>;<Anzahl>`

`<Spielstein-Typ>` ist `C` (für „cuboid“) oder `D` (für „dome“). `<Anzahl>` repräsentiert die Anzahl der Spielsteine im aktuellen Spielsteinvorrat.

A.4.7 Der `cellprint`-Befehl

Der `cellprint`-Befehl gibt eine Liste aller Spielelemente, die auf einer bestimmten Zelle platziert sind, auf die Konsole aus. Zu den Spielelementen zählen: Spielfiguren, Quader und Kuppeln.

Eingabeformat `cellprint <Zeilennummer>;<Spaltennummer>`

`<Zeilennummer>` und `<Spaltennummer>` sind jeweils ein Integer und bezeichnen die Zelle, deren Spielelemente ausgegeben werden sollen.

Ausgabeformat Die Spielelemente werden als komma-separierte Liste ausgegeben. Für eine Spielfigur wird deren Bezeichner ausgegeben. Für einen Quader wird `C` ausgegeben. Für eine Kuppel wird `D` ausgegeben. Das unterste Spielelement mit direktem Kontakt zum Spielfeld wird zuerst ausgegeben, anschließend das darüber liegende Spielelement, und so weiter.

Falls die Zelle keine Spielelemente enthält, wird `Empty` ausgegeben.

Beispielinteraktion

```
> cellprint 1;1
C,C,C,D
> cellprint 3;2
Empty
```

A.4.8 Der `print`-Befehl

Der parameterlose `print`-Befehl gibt das 5×5-große Spielfeld auf die Konsole aus.

Eingabeformat `print`

Ausgabeformat Das Spielfeld wird zeilenweise ausgegeben. Aufeinanderfolgende Zellen der gleichen Zeile werden durch ein Komma separiert. Eine Zelle wird durch ihr oberstes Spielelement repräsentiert: Für eine Spielfigur wird deren Bezeichner ausgegeben. Für einen Quader wird `C` ausgegeben. Für eine Kuppel wird `D` ausgegeben. Für eine leere Zelle wird `.` ausgegeben.

Beispielinteraktion

```
.,.,.,.,.
.,.,.,blue,.
C,yellow,green,C,.
.,red,.,.,.
.,.,.,.,.
```

A.4.9 Der `quit`-Befehl

Der parameterlose `quit`-Befehl ermöglicht es, das 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.

Beispielinteraktion

```
> quit quit
Error, incorrect input format, this command does not accept any parameters.
> quit
```

A.5 Beispielinteraktion

Beachten Sie im Folgenden, dass Eingabezeilen mit dem >-Zeichen eingeleitet werden, gefolgt von einem Leerzeichen. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabe. Der Beispielablauf erfolgt mit folgender Startkonfiguration: `Santorini yellow;1;1 red;3;2 blue;1;2 green;2;2`. Der Programmname Santorini ist nicht vorgeschrieben.

Beispielinteraktion

```
> move yellow;2;1
OK
> build C;3;1
OK
> turn
P2
> move blue;1;3
OK
> build C;2;3
OK
> turn
P1
> move red;3;1
OK
> build C;2;0
OK
> print
.,.,.,.,.
.,.,.,blue,.
C,yellow,green,C,.
.,red,.,.,.
.,.,.,.,.
> turn
P2
> draw-card Hermes
OK
> move blue;1;0
OK
> build C;0;0
OK
> quit
```