Karlsruher Institut für Technologie

Programmieren – Wintersemester 2018/19

Prof. Dr. Ralf H. Reussner
Software Design und Quality (SDQ)
https://sdqweb.ipd.kit.edu/wiki/Programmieren
programmieren-vorlesung@ipd.kit.edu

Abschlussaufgabe 1

Ausgabe: 08.02.2019 – 13:00 Uhr
Abgabe: 09.03.2019 – 06:00 Uhr

Bearbeitungshinweise

- Achten Sie darauf, nicht zu lange Zeilen, Methoden, Klassen und Schnittstellen zu erstellen[1].
- 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 geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete java.lang, java.util und java.util.regex, es sei denn, die Aufgabenstellung erlaubt ausdrücklich weitere Pakete[1].
- Achten Sie auf fehlerfrei kompilierenden Programmcode[1].
- Halten Sie alle Whitespace-Regeln ein[1].
- Halten Sie die Regeln zu Variablen-, Methoden- und Paketbenennung ein und wählen Sie aussagekräftige Namen[1].
- Halten Sie die Regeln zur Javadoc-Dokumentation ein[1].
- Nutzen Sie nicht das default-Package[1].
- Halten Sie auch alle anderen Checkstyle-Regeln ein[1].
- `System.exit()` und `Runtime.exit()` dürfen nicht verwendet werden[1].
- Halten Sie die Hinweise zur Modellierung im Ilias-Wiki ein.

Wichtig: Das Einreichen fremder Lösungen, seien es auch teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen wie beispielsweise der Lehrveranstaltung oder dem Übungsbetrieb selbst, ist ein Täuschungsversuch und führt zur Bewertung nicht bestanden. Dies beinhaltet unter anderem auch die Lösungsvorschläge des Übungsbetriebes. Es werden nur rein selbstständig erarbeitete Lösungen akzeptiert. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

[1] Der Praktomat wird die Abgabe zurückweisen, falls diese Regel verletzt ist.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Freitag, den 22.02.2019 um 13:00 Uhr**, freigeschaltet.
Laden Sie die `Terminal`-Klasse nicht mit hoch.

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

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`. Fehlermeldungen werden ausschließlich über die Terminal-Klasse ausgegeben und müssen aus technischen Gründen unbedingt mit `Error,` beginnen. Dies kann z.B. über die Terminal.printError-Methode erfolgen. **Modifizieren Sie niemals die Terminal-Klasse und laden Sie diese auch niemals zusammen mit Ihrer Abgabe hoch.**

Fehlerbehandlung
Ihre Programme sollen auf ungültige Benutzereingaben mit einer aussagekräftigen Fehlermeldung reagieren. Aus technischen Gründen muss eine Fehlermeldung unbedingt mit `Error,` beginnen. Eine Fehlermeldung führt nicht dazu, dass das Programm beendet wird; es sei denn, die nachfolgende Aufgabenstellung verlangt dies ausdrücklich. Achten Sie insbesondere auch darauf, dass unbehandelte `RuntimeExceptions`, bzw. Subklassen davon – sogenannte *Unchecked Exceptions* – nicht zum Abbruch Ihres Programms führen sollen.

Objektorientierte Modellierung
Achten Sie darauf, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden.

Öffentliche Tests
Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe nötig ist. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

Dawn 11/15 (20 Punkte)

In dieser Aufgabenstellung soll eine modifizierte Variante des Brettspiels *DAWN 11/15*[2] von Prof. Dr. Ingo Althöfer implementiert werden. Das Spiel ist durch eine NASA-Mission[3] inspiriert. Hierbei wurden die beiden Himmelskörper Vesta (2011) und Ceres (2015) von der am 27.09.2007 gestarteten NASA-Sonde *Dawn* für mehrere Monate wissenschaftlich untersucht.

Bei diesem Spiel treten immer zwei Spieler gegeneinander an. Für das Spiel braucht man ein rechteckiges 11×15 großes Spielbrett, zwei Spielsteine, die die Himmelskörper Vesta und Ceres repräsentieren, sowie weitere Spielsteine in unterschiedlicher Größe und einen Würfel.

Einer der Spieler – als *Nature* bezeichnet – führt die beiden Himmelskörper, der andere – als *Mission Control* bezeichnet – setzt Spielsteine, die die *Dawn*-Sonden symbolisieren. Per Würfel mit den Symbolen *2, 3, 4, 5,*

[2] `https://www.althofer.de/dawn-11-15.html`
[3] `https://solarsystem.nasa.gov/missions/dawn/overview/`

6 und *DAWN* wird ermittelt, welcher *Dawn*-Spielstein gesetzt werden darf und zugleich, wie viele Felder die Himmelskörper Vesta und Ceres für ihre Flucht nutzen können.

Beachten Sie, dass für die Abschlussaufgabe nur die hier genannten Regeln gültig sind und nicht die auf der ursprünglichen Homepage.

Spielregeln und Spielphasen

Das 11×15 große Spielbrett besteht aus insgesamt 165 Feldern. Jedes Feld kann mit Hilfe von Koordinaten (m, n) mit $0 \leq m \leq 10$ und $0 \leq n \leq 14$ beschrieben werden. Die Ausrichtung des Spielbrettes und die Festlegung der Koordinaten ist in Abbildung 1 zu finden.

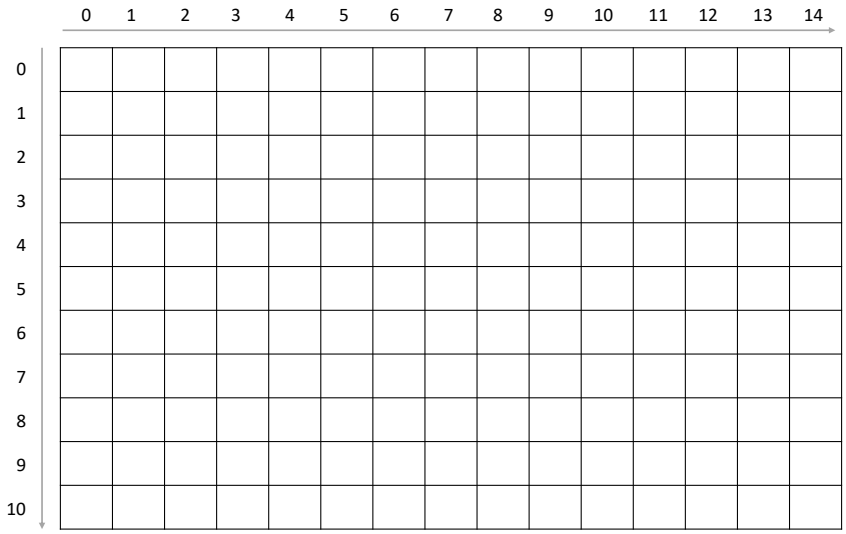

Abbildung 1: Festgelegte Koordinaten für das Spielbrett

Zu Beginn des Spiels ist das Spielbrett leer. Ein Spieler übernimmt die Rolle von *Nature*, der andere Spieler übernimmt *Mission Control*. *Nature* besitzt die Spielsteine *Vesta* und *Ceres*, welche jeweils eine Größe von 1×1 haben. *Mission Control* besitzt zwei gleiche Sätze an Spielsteinen. Jeder Spielstein hat immer eine Breite von einem Feld. Ein Satz Spielsteine enthält genau sechs verschiedene Spielsteine, die je eine Länge von 2, 3, 4, 5, 6 und 7 haben. Der Spielstein mit der Länge 7 wird DAWN genannt. Der erste Satz an Spielsteinen (ein Spielstein *Vesta* und sechs *Mission Control*-Spielsteine) wird in Spielphase 1 verwendet, der zweite Satz an Spielsteinen (ein Spielstein *Ceres* und sechs *Mission Control*-Spielsteine) in Spielphase 2.

Spielsteine dürfen sich beim Platzieren nicht überschneiden und müssen immer vollständig auf dem Spielbrett sein. Die einzige Ausnahme ist der Spielstein DAWN, dieser darf über das Spielbrett hinausragen aber sich nicht mit anderen überschneiden. Alle Spielsteine müssen flach hingelegt werden. Dies heißt, dass z.B. ein Spielstein der Länge 5 genau 5 Spielfelder in horizontaler oder vertikaler Richtung benötigt. Die *Vesta* und *Ceres* Spielsteine von Spieler *Nature* dürfen bei ihrer Flucht nicht über andere Spielsteine (*Vesta*, *Ceres* und *Mission Control*-Spielsteine) steigen.

Das Brettspiel *DAWN 11/15* setzt sich aus zwei Spielphasen zusammen. Eine Spielphase endet wenn der Spieler *Mission Control* seinen letzten Spielstein für diese Phase platziert hat und der Spieler *Nature* seinen Spielzug beendet hat. Ein Spiel endet nach Ende der zweiten Spielphase.

Spielphase 1 – Jagd auf Vesta

Diese Spielphase besteht aus insgesamt sechs Runden. Nach der sechsten Runde beginnt Spielphase 2. Jede Runde beinhaltet drei Aktionen i), ii) und iii), die nacheinander ausgeführt werden. Initial zu Beginn der

Spielphase muss der Spieler *Nature* seinen Spielstein *Vesta* auf dem Spielbrett platzieren. Die drei Aktionen werden im Folgenden näher erläutert:

i) Der Spieler *Nature* würfelt. Die Seiten des Würfels sind wie oben beschrieben mit den Symbolen *2, 3, 4, 5, 6* und *DAWN* versehen. Das Würfelsymbol *DAWN* steht für die Spielsteinlänge 7.

ii) Der Spieler *Mission Control* platziert gemäß des Würfelsymbols den jeweiligen Spielstein auf dem Spielbrett. Wenn der Würfel ein Symbol anzeigt, deren Spielsteinlänge noch nicht im Spielbrett platziert ist, soll genau dieser Spielstein von Spieler *Mission Control* platziert werden. Wenn sich dieser Spielstein bereits auf dem Spielbrett befindet, muss aus dem Satz der verbleibenden Spielsteine derjenige mit der nächst kleineren Länge oder der nächst größeren Länge auf dem Spielbrett platziert werden. Nachdem ein *Mission Control*-Spielstein platziert wurde, darf dieser nicht mehr bewegt werden.

iii) Der Spieler *Nature* bewegt den Spielstein *Vesta* gemäß der Länge des ausgewählten *Mission Control*-Spielstein über die unmittelbaren Nachbarfelder auf dem Spielbrett in horizontaler oder vertikaler Richtung. Die maximale Anzahl der Schritte ergibt sich demnach aus der Länge des Spielsteins, den die *Mission Control* gerade eingefügt hat. Eine Bewegung des Spielsteins *Vesta* besteht aus einer Folge von elementaren Schritten. Jeder Elementarschritt erfolgt auf einem freien Feld in horizontaler oder vertikaler Richtung. In diagonaler Richtung ist ein Elementarschritt nicht zulässig. Die Länge des gerade in ii) verwendeten Spielsteins gibt die maximale Anzahl der Elementarschritte an. Bei der Aktion muss mindestens ein Elementarschritt durchgeführt werden. Falls kein Schritt mehr möglich ist, wird diese Aktion iii) übersprungen.

Spielphase 2 – Jagd auf Vesta und Ceres

Nachdem Spielphase 1 beendet ist, platziert der Spieler *Nature* den Spielstein *Ceres* auf einem freien Feld des Spielfeldes. *Vesta* und alle anderen Spielsteine aus Spielphase 1 bleiben auf ihren Positionen. Auch Spielphase 2 besteht aus sechs Runden. Jede Runde hat ebenso wie Spielphase 1 drei Aktionen i), ii), iii), die nacheinander ausgeführt werden. Es gelten hier analog dieselben Ausführungen wie in Spielphase 1.

i) Der Spieler *Nature* würfelt. Die Seiten des Würfels sind wie oben beschrieben mit den Symbolen *2, 3, 4, 5, 6* und *DAWN* versehen. Das Würfelsymbol *DAWN* steht für die Spielsteinlänge 7.

ii) Der Spieler *Mission Control* platziert gemäß des Würfelsymbols den jeweiligen Spielstein aus dem zweiten Satz auf dem Spielbrett. Wenn der Würfel ein Symbol anzeigt, deren Spielsteinlänge noch nicht im Spielbrett platziert ist, soll genau dieser Spielstein von Spieler *Mission Control* platziert werden. Wenn dieser Spielstein sich bereits auf dem Spielbrett befindet, muss aus dem Satz der verbleibenden Spielsteine derjenige mit der nächst kleineren Länge oder der nächst größeren Länge auf dem Spielbrett platziert werden.

iii) Der Spieler *Nature* bewegt den Spielstein *Ceres* gemäß der Länge des ausgewählten *Mission Control*-Spielsteins über die unmittelbaren Nachbarfelder auf dem Spielbrett in horizontaler oder vertikaler Richtung. Die maximale Anzahl der Schritte ergibt sich demnach aus der Länge des Spielsteins, den die *Mission Control* gerade eingefügt hat. Eine Bewegung des Spielsteins *Ceres* besteht aus einer Folge von elementaren Schritten. Jeder Elementarschritt erfolgt auf einem freien Feld in vertikaler oder horizontaler Richtung. In diagonaler Richtung ist ein Elementarschritt nicht zulässig. Die Länge des gerade in ii) verwendeten Spielsteins gibt die maximale Anzahl der Elementarschritte an. Bei der Aktion muss mindestens ein Elementarschritt durchgeführt werden, außer es ist kein Schritt mehr möglich.

Nach dem Ende der Spielphase 2 kann bis zum Neustarten des Spieles, nicht mehr gewürfelt, oder Spielsteine bewegt werden. Erlaubte Aktionen sind: Status einer Zelle ausgeben, Spielfeld ausgeben, Ergebnis ausgeben, Anwendung beenden, oder ein neues Spiel starten;

Berechnung der freien Felder

Um das Ergebnis des Spiels zu bestimmen, müssen nach Ablauf der zweiten Spielphase die freien Felder um *Vesta* und *Ceres* bestimmt werden. Die Anzahl der freien Felder werden für *Vesta* mit F(V) und für *Ceres* mit

F(C) angegeben. Es wird hiernach die Anzahl der freien Felder um einen Himmelskörper bestimmt, ohne über die Himmelskörper selbst oder über irgendwelche weiteren Spielsteine zu steigen.

Abb. 2 zeigt einen Beispiel Spielstand. Dieser ist jedoch kein möglicher Endstand, sondern dient nur als Demonstration. Dabei sind die schwarzen Kästchen die Spielsteine von *Mission Control*, der rote Spielstein ist *Vesta* und der gelbe Spielstein ist *Ceres*. In diesem Beispiel wäre F(V) = 7 und F(C) = 6. Der Unterschied ergibt sich, dass *Ceres* nicht in das Feld (5,2) gelangen kann, da *Vesta* den Weg dahin blockiert. Das Ergebnis des Spiels lässt sich formal wie folgt ausdrücken:
$E = max\{F(C), F(V)\} + [max\{F(C), F(V)\} - min\{F(C), F(V)\}]$
Nach dieser Formel wäre das Ergebnis in dem angegebenen Beispiel $E = 8$.

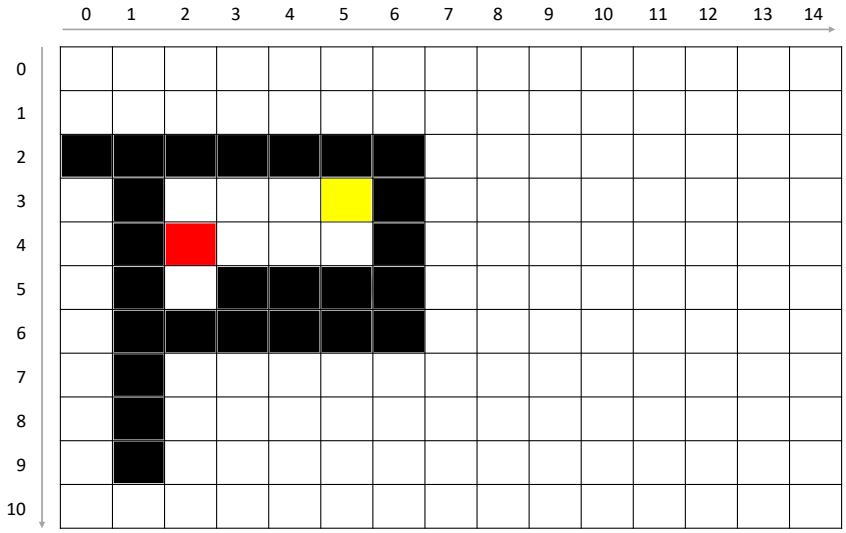

Abbildung 2: Beispiel Spielbrett

Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole mittels `Terminal.readLine()` verschiedene Arten von Befehlen entgegen, die im Folgenden näher spezifiziert werden. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm irgendwann durch `quit` beendet wird. Alle Befehle werden auf dem aktuellen Zustand des Spiels ausgeführt. Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Spielregeln nicht verletzt werden und geben Sie in diesen Fällen immer eine Fehlermeldung beginnend mit `Error,` aus. Direkt nach Programmstart ist der Spieler *Nature* aktiv bzw. an der Reihe. Bei der Beschreibung der Eingabeformate der Befehle stehen die Wörter in spitzen Klammern (< und >) für Platzhalter, welche bei Eingabe durch Parameter ersetzt werden. Die eigentlichen Parametern enthalten keine Klammern (vgl. Parameter Beispielablauf).

Der `state`-Befehl

Der Befehle `state` gibt den Zustand eines Spielfeldes mit bestimmten Koordinaten (m, n) aus, wobei gilt $0 \leq m \leq 10$ und $0 \leq n \leq 14$. Die Zeilen- und Spaltennummern sind durch exakt ein `;` (Semikolon) separiert.

Eingabeformat `state <m>;<n>`

Ausgabeformat Ein Feld auf dem Spielbrett kann sich in einem der folgenden Zustände befinden:

- Handelt es sich um ein leeres Feld, wird dies durch `-` dargestellt.

- Befindet sich auf einem Feld der Spielstein *Ceres* wird `C` ausgegeben. Der Spielstein *Vesta* wird durch `V` dargestellt.
- `+` wird ausgegeben, wenn sich auf einem Feld ein Spielstein des Spielers *Mission Control* befindet.

Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben, inkorrekte Koordinatenangaben, usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Der `print`-Befehl

Der parameterlose Befehl `print` ermöglicht es, das aktuelle Spielbrett auf der Kommandozeile auszugeben. Es findet sonst keine weitere Ausgabe statt. Dieser Befehl ist in jeder Spielphase ausführbar, auch nachdem das Ergebnis des Spiels ermittelt wurde.

Eingabeformat `print`

Ausgabeformat Das Spielbrett wird zeilenweise von oben nach unten und pro Zeile die Felder von links nach rechts ausgegeben. ~~Die Felder sind jeweils durch exakt ein ; (Semikolon) getrennt.~~ Die Zeilen sind jeweils durch einen Zeilenumbruch getrennt. Es existieren hierbei folgende Fallunterscheidungen:

- Ein leeres Feld wird durch `-` dargestellt.
- Befindet sich auf einem Feld der Spielstein *Ceres* wird `C` ausgegeben. Der Spielstein *Vesta* wird durch `V` dargestellt.
- `+` wird ausgegeben, wenn sich auf einem Feld ein Spielstein des Spielers *Mission Control* befindet.

Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Nachfolgend eine Beispielausgabe für Abb. 2.

Beispielausgabe

```
---------------
---------------
++++++--------
-+---C+--------
-+V---+--------
-+-++++--------
-++++++--------
-+-------------
-+-------------
-+-------------
---------------
```

Der `set-vc`-Befehl

Dieser Befehl wird verwendet um die Spielsteine *Vesta* und *Ceres* auf ein leeres Feld im 11×15 großen Spielbrett zum Beginn der jeweiligen Spielphasen zu setzen. Für das Setzen der Spielsteine werden in Eingabeformat die jeweiligen Koordinaten (m, n) angegeben, für die gilt: $0 \leq m \leq 10$ und $0 \leq n \leq 14$. Die Koordinaten sind jeweils durch exakt ein **;** (Semikolon) separiert.

Eingabeformat `set-vc <m>;<n>`

Ausgabeformat `OK`
Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B. bei einem falsch spezifizierten Eingabeformat oder wenn die Koordinaten nicht im gültigen Wertebereich liegen usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Der `roll`-Befehl

Mit diesem Befehl kann der Benutzer im Rahmen der Regeln ein Symbol übergeben, das als das Ergebnis eines Würfelwurfs interpretiert wird. Als Eingabeparameter sind hier die Würfelsymbole *2, 3, 4, 5, 6* und *DAWN* erlaubt.

Eingabeformat `roll <symbol>`

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

Der `place`-Befehl

Der `place`-Befehl ermöglicht es, einen Spielstein von *Mission Control* in den jeweiligen Spielphasen zu platzieren. Die Parameter legen die Anfangs- und Endkoordinaten des zu setzenden Spielsteines nach den oben angegebenen Regeln fest. Die einzelnen Koordinaten sind durch **:** (Doppelpunkt) getrennt.

Eingabeformat `place <x1>;<y1>:<x2>;<y2>`

Ausgabeformat `OK`
Im Erfolgsfall wird `OK` ausgeben. Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben, falscher Auswahl der Spielsteinlänge, inkorrekte Koordinatenangaben, Spielsteinplatzierung auf nicht freien Felder, inkorrekte Verwendung des Befehl usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Nachfolgend 4 Beispiel-Szenarien, die das Verhalten von *place* genauer zeigen. Die Szenarien zeigen **kein** vollständiges Spiel, sondern greifen nur beispielhaft den *place*-Befehl heraus.

Beispiel-Szenarien

```
/* Beispiel Interaktion um einen DAWN Stein zu legen */
> place 5;-6:5;0
OK
> print
---------------
---------------
---------------
---------------
---------------
+--------------
---------------
---------------
---------------
---------------
---------------

/* Beispiel Interaktion um einen Spielstein der Länge 5 zu legen. */
> place 0;-4:0;1
Error, the token is not completely on the board.

/* Beispiel Interaktion um einen Spielstein der Länge 5 zu legen */
> place 0;0:0;4
OK
> print
+++++----------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
/* Beispiel Interaktion um einen Spielstein der Länge 5 zu legen */
> place 0;4:0;0
OK
> print
+++++----------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
---------------
```

Der `move`-Befehl

Der Befehl ermöglicht es dem Spieler *Nature* seine Spielsteine *Vesta* und *Ceres* in den jeweilig vorgesehenen Spielphasen gemäß der Länge des ausgewählten *Mission Control*-Spielsteins auf ein freies Feld zu bewegen. Die Koordinaten (m_i, n_i) mit $0 \leq m_i \leq 10$ und $0 \leq n_i \leq 14$ geben hier den Pfad an. Die einzelnen Koordinaten sind durch `:` (Doppelpunkt) getrennt.

Eingabeformat `move <`m_1`>;<`n_1`>:<`m_i`>;<`n_i`>`

Ausgabeformat Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.b. bei falsch spezifizierten Eingaben oder bei inkorrekter, nicht-spielregelkonformer Koordinateneingabe, kein Spielzug möglich, usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispiel-Szenarien für `move`-Befehlt

```
/* Vesta steht auf 0;0 und darf maximal 5 Schritte gehen */
> move 0;1:0;2:0;3:0;4:0;5
OK

/* Vesta steht 0;1 und darf maximal 4 Schritte gehen */
> move 0;1:0;2:0;3
Error, field 0;1 is occupied.

/* Vesta steht 0;1 und darf maximal 4 Schritte gehen */
> move 0;0:0;1:0;2:0;3
OK

/* Vesta steht auf 0;1 und darf maximal 7 Schritte gehen */
> move 0;0
OK
```

Der `show-result`-Befehl

Der parameterlose Befehl `show-result` ist nur nach Beendigung der zweiten Spielphase ausführbar. Er ermittelt nach der Berechnungsvorschrift das Ergebnis eines Spiels.

Eingabeformat `show-result`

Ausgabeformat *I*

Das Ergebnis eines Spiels wird im Ausgabeformat durch eine Ganzzahl angegeben. Wobei *I* einen Integer darstellt.

Hinweis: Sie können zur Berechnung der freien Felder unter anderem eine modifizierte Breitensuche verwenden.

Der `reset`-Befehl

Der parameterlose Befehl `reset` bricht das aktuelle Spiel ab, beendet jedoch nicht das Programm. Das Spiel wird erneut initilisiert und das Spiel beginnt von neuem.

Eingabeformat `reset`

Ausgabeformat `OK`

Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B. bei einem falsch spezifiziertem Eingabeformat) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Der `quit`-Befehl

Der parameterlose Befehl ermöglicht es, das laufende Programm vollständig zu beenden. Beachten Sie, dass hierfür keine Methoden wie `System.exit()` oder `Runtime.exit()` verwendet werden dürfen. Die Eingabe des `quit`-Befehls ist immer möglich, unabhängig davon, in welcher Phase sich das Spiel gerade befindet.

Eingabeformat `quit`

Ausgabeformat Im Erfolgsfall findet keine Konsolenausgabe statt. Im Fehlerfall (z.B. bei einem falsch spezifiziertem Eingabeformat) wird eine aussagekräftige Fehlermeldung beginnend mit `Error, ` ausgegeben.

Beispielablauf

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.

Beispielablauf 1/3

```
> set-vc -5;2
Error, the given coordinate -5;2 is not on the board.
> set-vc 5;2
OK
> state 5;2
V
> roll DAWN
OK
> reset
OK
> set-vc 5;2
OK
> roll 5
OK
> place 6;2:6;6
OK
> move 4;2:3;2
OK
> roll 5
OK
> place 2;0:2;5
OK
> move 4;2:3;2
OK
> roll DAWN
OK
> place 3;1:9;1
OK
> move 4;2
OK
> roll 3
OK
> place 5;3:5;5
OK
> move 5;2:4;2:5;2
OK
> roll 2
OK
> place 0;0:0;1
OK
> move 4;2:5;2
OK
> roll 3
OK
> place 2;6:5;6
OK
```

Beispielablauf 2/3

```
> move 4;2:3;2:4;2
OK
> print
++-------------
---------------
++++++++--------
-+----+--------
-+V---+--------
-+-++++--------
-++++++--------
-+-------------
-+-------------
-+-------------
---------------
> set-vc 3;5
OK
> roll DAWN
OK
> place 0;14:0;20
OK
> move 4;5:4;4:4;3:4;2:3;2:3;3:3;4
Error, field 4;2 is occupied.
> move 4;5:4;4:4;3:3;3:3;2:3;3:3;4
OK
> roll DAWN
OK
> place 10;0:10;5
OK
> move 4;4:4;3:3;3:3;4:3;5:4;5
OK
> roll 2
OK
> place 1;13:1;14
OK
> move 3;5:3;4
OK
> roll 5
OK
> place 2;10:2;14
OK
> move 4;4:4;5:3;5
OK
> roll 5
OK
> place 3;11:3;14
OK
> move 4;5:3;5
OK
> roll 3
OK
```

Beispielablauf 3/3

```
> place 4;12:4;10
OK
> move 3;4:4;4:4;5:3;5
OK
> show-result
8
> print
++------------+
-------------++
+++++++---+++++
-+---C+----++++
-+V---+---+++--
-+-++++--------
-++++++--------
-+--------------
-+--------------
-+--------------
++++++---------
> reset
OK
> set-vc 0;0
OK
> print
V---------------
----------------
----------------
----------------
----------------
----------------
----------------
----------------
----------------
----------------
----------------
> quit
```