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 2019/20

Übungsblatt 5 Version 1.1	20 Punkte	Ausgabe: 15.01.2020, ca. 13:00 Uhr Praktomat: 22.01.2020, 13:00 Uhr Abgabefrist: 30.01.2020, 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` 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()` 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 erarbeitete Lösungen akzeptiert. Das Einreichen fremder Lösungen, seien es auch nur teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen wie beispielsweise der Lösungsvorschläge des Übungsbetriebes, ist ein Täuschungsversuch und führt zur Bewertung „nicht bestanden“. Ausdrücklich ausgenommen hiervon sind Quelltextschnipsel von den Vorlesungsfolien. 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. 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 (ILIAS) beschreibt, wie das funktioniert.

>_ 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. Laden Sie die Terminal-Klasse **niemals** zusammen mit Ihrer Abgabe hoch.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Mittwoch, den 22. Januar 2020 um 13:00 Uhr**, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Praktomaten bei der richtigen Aufgabe vor Ablauf der Abgabefrist hochzuladen.

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

Lernziele

Der Fokus dieses Übungsblattes liegt in den nachfolgenden Lernzielen aus der Vorlesung, inklusive der Übung. Beachten Sie, dass alle anderen bisherigen Lernziele ebenso gelten. Berücksichtigen Sie daher generell den gesamten bisherigen Inhalt aus Vorlesung und Übung.

- Java-API
  - Sie wissen, dass Sie in der Java API häufig benötigte Funktionalität finden
  - Sie haben einen Überblick über häufig verwendete und hilfreiche Klassen wie Object, String, Math, Enum, Comparable, Wrapper-Klassen sowie die Java Collections
  - Sie haben einen Überblick über wichtige Pakete
- Prinzipien des objektorientierte Entwurfs
  - Studierende können Prinzipien des objektorientierten Entwurfs bei der Programmierung von Java anwenden
  - Studierende können Programmcode anhand der Prinzipien bewerten und verbessern
  - Studierende können gut wartbaren und gut lesbaren Programmcode erstellen
- Best Practices
  - Studierende kennen Best Practices für oft auftretende Entscheidungen im Entwurf von Java-Programmen
  - Studierende können die fünf vorgestellten Prinzipien anwenden
  - Studierende kennen den Einsatzzweck der Prinzipien
  - Studierende können gut lesbaren und gut wartbaren Code in Java schreiben

Aufgabe A: Brettspiel (20 Punkte)

In dieser Aufgabe implementieren Sie ein Brettspiel, bei dem zwei Spieler abwechselnd ihre Spielsteine auf die Zellen eines 6×6 Spielbrettes platzieren. Es gibt 16 Spielsteine, worauf beide Spieler Zugriff haben. Der erste Spieler beginnt immer das Spiel. Der Spieler, der als erster vier Spielsteine, die alle mindestens eine gemeinsame Eigenschaft haben, in eine Zeile, Spalte oder Diagonale platzieren kann, gewinnt.

Ein wichtiges Ziel dieser Aufgabe ist es, dass Sie eine saubere objektorientierte Modellierung planen und umsetzen. Überlegen Sie sich, welche Klassen hierzu nötig sind und welche Verbindungen zwischen ihnen bestehen müssen. Achten Sie dabei auf die Trennung von Ein-/Ausgabe und Funktionalität sowie die anderen aus der Vorlesung bekannten Kriterien für guten objektorientierten Entwurf.

A.1 Spielaufbau

Das *Spielfeld* ist quadratisch und besteht aus 6×6 Zellen. Eine Zelle $c_{i,j}$, die sich nicht am Rand befindet, besitzt 8 *angrenzende Zellen*, wie Abb. 0.1 zeigt; Zellen am Spielfeldrand besitzen entsprechend weniger angrenzende Zellen. Dabei bezeichnet i die Zeilennummer und j die Spaltennummer einer Zelle. Zeilennummer (von Oben nach Unten) und Spaltennumer (von Links nach Rechts) beginnen jeweils bei 0. Somit ist $c_{0,0}$die oberste linke Zelle. Außerdem ist das Spielfeld zu Beginn des Spiels leer.

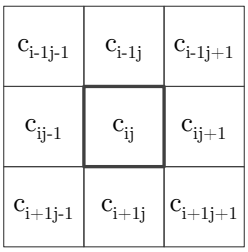

Abbildung 0.1: Eine Zelle $c_{i,j}$ samt ihrer angrenzenden Zellen

Auf jeder Zelle kann maximal ein Spielstein platziert werden. Es gibt 16 verschiedene *Spielsteine*. Jeder Spielstein muss entweder groß oder klein, entweder schwarz oder weiß, entweder eckig oder zylinderförmig und entweder hohl oder massiv sein. Somit haben keine zwei Spielsteine identische Eigenschaften. Beispiel: Ein Spielstein kann groß, schwarz, zylinderförmig und massiv sein. Für den Rest des Aufgabenblattes werden die 16 Spielsteine wie folgt durchnummeriert:

- Spielstein 0: schwarz, eckig, klein, hohl.
- Spielstein 1: schwarz, eckig, klein, massiv.
- Spielstein 2: schwarz, eckig, groß, hohl.
- Spielstein 3: schwarz, eckig, groß, massiv.
- Spielstein 4: schwarz, zylinderförmig, klein, hohl.

- Spielstein 5: schwarz, zylinderförmig, klein, massiv.
- Spielstein 6: schwarz, zylinderförmig, groß, hohl.
- Spielstein 7: schwarz, zylinderförmig, groß, massiv.
- Spielstein 8: weiß, eckig, klein, hohl.
- Spielstein 9: weiß, eckig, klein, massiv.
- Spielstein 10: weiß, eckig, groß, hohl.
- Spielstein 11: weiß, eckig, groß, massiv.
- Spielstein 12: weiß, zylinderförmig, klein, hohl.
- Spielstein 13: weiß, zylinderförmig, klein, massiv.
- Spielstein 14: weiß, zylinderförmig, groß, hohl.
- Spielstein 15: weiß, zylinderförmig, groß, massiv.

A.2 Spielregeln

Zwei Spieler treten gegeneinander an. Jeder Spieler kann auf die noch nicht auf dem Spielfeld platzierten Spielsteine zugreifen. Das Spiel endet, wenn alle Spielsteine auf dem Spielfeld platziert sind (und das Spiel unentschieden ausgegangen ist) oder wenn einer der Spieler gewonnen hat.

A.2.1 Spielzug

Der Spiel beginnt, indem der erste Spieler im ersten Zug einen Spielstein auswählt, den der zweite Spieler im selben Zug auf das Spielfeld platziert. Beim nächsten Zug wählt nun der zweite Spieler einen Spielstein aus, den der erste Spieler auf das Feld platzieren muss, usw. Dies bedeutet, dass ein Zug aus dem Wählen des Spielsteines durch den einen Spieler und dem Platzieren des Spielsteines durch den anderen Spieler besteht. Gelingt es dem Spieler, der im aktuellen Zug den Spielstein platzieren muss, den Spielstein so zu platzieren, dass er als erster 4 Spielsteine, die alle mindestens eine gemeinsame Eigenschaft haben, in eine Zeile, Spalte oder Diagonale platzieren kann, dann gewinnt dieser Spieler. Hierbei müssen diese vier Steine direkt nebeneinander liegen. Somit kann nur der Spieler, der in einem Zug den Spielstein platziert, gewinnen. Die Züge sind in dieser Aufgabe beginnend mit 0 durchnummeriert.

A.2.2 Standardspielfeld

Die Ränder des Spielfeldes sind abgeschlossen. Außerhalb des Spielfeldes ist die Welt undefiniert. Dies bedeutet, dass keiner der Spieler seinen Spielstein außerhalb des Spielfeldes platzieren darf.

A.2.3 Torusspielfeld

Bei einem Torusspielfeld sind die äußersten Zellen in der Nachbarschaft der gegenüber liegenden äußersten Zellen. Abbildung 0.2 illustriert beispielhalft ein Torusspielfeld mit 4 Zeilen und Spalten. Stellt man sich das Spielfeld analog zu einer Matrix $A = (a_{i,j})$ mit insgesamt n Zeilen und m Spalten, beginnend mit 0 durchnummeriert, vor, dann sind die Ränder des Spielfeldes wie folgt definiert: $a_{-1,j} = a_{n-1,j}$, $a_{i,-1} = a_{i,m-1}$, $a_{n,j} = a_{0,j}$, $a_{i,m} = a_{i,0}$.

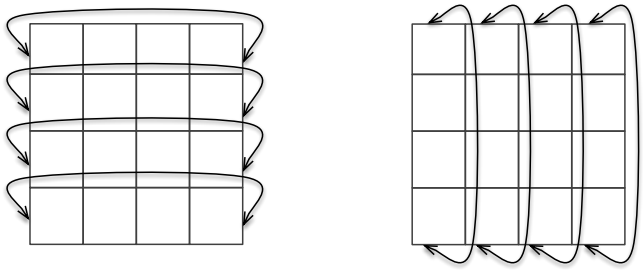

Abbildung 0.2: Abbildung eines exemplarischen 4×4 Spielfeldes mit einem torusartigen Rahmen

Die Ränder des Spielfeldes sind offen. Versucht ein Spieler seinen Spielstein außerhalb des Spielfeldes auf die Zelle mit Zeilennummer i und Spaltennummer j zu platzieren, wird der Spielstein automatisch auf die Zelle $i\%6$ und $j\%6$ platziert, falls i und $j > 5$ sind. Für negative i und j, wird der Spielstein auf die Zelle ~~$6 - (|i|\%6)$~~$((i\%6) + 6)\%6$ und ~~$6 - (|j|\%6)$~~$((j\%6) + 6)\%6$ platziert. ~~$|i|$ bezeichnet den Betrag der Zahl i.~~

Bei einem torusartigen Spielfeld gewinnt somit der Spieler, der als erster vier Spielsteine, die alle mindestens eine gemeinsame Eigenschaft haben, in eine Zeile, Spalte oder Diagonale auch über den Spielrahmen hinweg platziert hat. Abbildung 0.3 illustiriert diesen Fall anhand eines Beispiels.

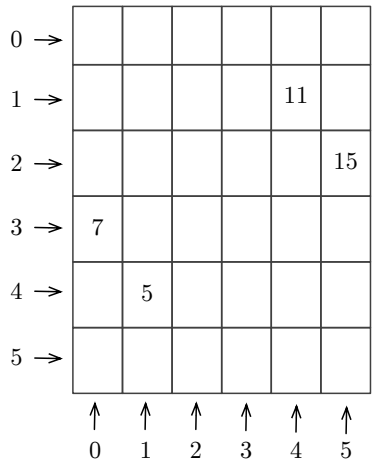

Abbildung 0.3: Beispiel eines Spielstandes mit einem Torusspielfeld, in dem der erste Spieler gewonnen hat, da alle Spielsteine massiv sind.

A.3 Interaktive Benutzerschnittstelle

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

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die zuvor definierten Spielregeln nicht verletzt werden und geben Sie in diesen Fällen immer 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.

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.

A.3.1 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.

`<Modus>` Identifiziert die Variante des Spielfelds, siehe Abschnitt A.2.1. Eine Zeichenkettte, welche entweder `standard` oder `torus` ist.

`<Spielstein>` Eine `int`-Zahl $n \in [0, 15]$. Diese Zahl repräsentiert einen auf dem Spielfeld zu platzierenden Spielstein, der zuvor aus dem Vorrat entnommen wird, siehe hierzu Abschnitt A.1.

`<Spaltennummer>` und `<Zeilennummer>` Beides sind `int`-Zahlen aus dem abgeschlossenes Intervall $[0, 5]$. Beachten Sie für die Zellen außerhalb des Spielfeldes den Abschnitt A.2.1.

A.3.2 Der `start`-Befehl

Der Befehl startet ein neues aktives Spiel. Dabei bedeutet `standard`, dass ein Standardspielfeld für das Spiel eingesetzt wird und `torus`, dass ein Torusspielfeld verwendet wird. Wird der Befehl während eines laufenden Spiels eingebeben, wird dieses abgebrochen und ein neues Spiel gestartet. Direkt nach Programmstart ist Spieler 1 *aktiv* bzw. an der Reihe. In jedem Zug gibt es immer zwei aktive Spieler, die sich in darauffolgenden Schritten abwechseln.

Eingabeformat `start <Modus>`

Ausgabeformat Wenn der Spielzug erfolgreich durchgeführt wurde, wird `OK` ausgegeben. Im Fehlerfall wird eine aussagekräftige Fehlermeldung ausgegeben.

A.3.3 `select`-Befehl

Mit dem `select`-Befehl entnimmt der aktive Spieler einen Spielstein aus dem Vorrat der beiden Spieler. Achten Sie darauf, wenn ein Spielstein ausgewählt wurde, ist er für das weitere Spiel nicht mehr vorhanden. Sobald dieser Befehl erfolgreich durchgeführt wurde, wird automatisch der aktive Spieler in diesem Zug gewechselt.

Nach einem erfolgreichen `select`-Befehl darf keine weiteren select-Befehle ausgeführt werden, bis ein place-Befehl ausgeführt wurde. Alle anderen Befehle können jedoch ausgeführt werden.

Eingabeformat `select <Spielstein>`

Ausgabeformat Wenn der Spielzug erfolgreich durchgeführt wurde, wird `OK` ausgegeben. Im Fehlerfall, beispielsweise wenn der Spielstein nicht vorhanden ist, wird eine aussagekräftige Fehlermeldung ausgegeben.

A.3.4 `place`-Befehl

Der `place`-Befehl wird immer durch den Spieler aufgerufen, der nicht den `select`-Befehl aufgerufen hat. Mit diesem Befehl wird der Spielstein, der im selben Zug aus dem Vorrat ausgewählt wurde, auf dem Spielbrett platziert. Sobald dieser Befehl erfolgreich durchgeführt wurde, beginnt automatisch der nächste Zug. Im nächsten Zug wird der Spieler, der im aktuellen Zug den `place`-Befehl aufgerufen hat, den `select`-Befehl aufrufen, usw. Schlägt der Befehl fehl, wird der entnommene Spielstein in diesem Zug in den Vorrat zurückgegeben. In diesem Fall wird der Zug rückgängig gemacht, d.h., dass sich die Zugnummer dementsprechend nicht erhöht wird. Die Reihenfolge der Spieler entspricht aber dem nicht ausgeführten Zug. Nach einem `place`-Befehl können keine weiteren `place`-Befehle ausgeführt werden, bis ein erfolgreicher `select`-Befehl ausgeführt wurde. Alle anderen Befehle können jedoch nach einem `place`-Befehl ausgeführt werden.

Eingabeformat `place <Zeilennummer>;<Spaltennummer>`

Ausgabeformat Wenn der Spielzug erfolgreich durchgeführt wurde, keiner der Spieler gewonnen hat und noch weitere Spielsteine im Vorrat existieren, wird `OK` ausgegeben. Wenn der Spielzug erfolgreich durchgeführt wurde und keiner der Spieler gewonnen hat, jedoch keine Spielsteine mehr im Vorrat vorhanden sind, wird `draw` ausgegeben. Gewinnt einer der Spieler in diesem Zug, wird in der ersten Zeile `P1 wins` ausgegeben, falls Spieler 1 gewinnt; `P2 wins`, falls Spieler 2 gewinnt. In der zweiten Zeile wird die Zugnummer ausgegeben. Die Züge sind beginnend mit 0 und fortlaufend durchnummeriert. Beachten Sie, dass die Zugnummer nur dann ausgegeben wird, wenn das Spiel beendet ist und einer der Spieler gewonnen hat. Im Fehlerfall, beispielsweise wenn die Zelle bereits durch einen anderen Spielstein besetzt ist oder zuvor kein erfolgreicher `select`-Befehl ausgeführt wurde, wird eine aussagekräftige Fehlermeldung ausgegeben. Ist das Spiel beendet (weil einer der Spieler gewonnen hat oder weil das Spiel unentschieden ausging), so dürfen alle Befehle außer dem `select`- und dem `place`-Befehl ausgeführt werden.

A.3.5 `bag`-Befehl

Der parameterlose `bag`-Befehl gibt den aktuellen Spielsteinvorrat der Spieler auf die Konsole aus. Somit sind alle Spielsteine, die zuvor mit einem `select`-Befehl ausgeführt wurden, nicht Teil der Ausgabe.

Eingabeformat `bag`

Ausgabeformat Der Spielsteinvorrat wird in einer eigenständigen Zeile ausgegeben, indem alle Spielsteine hintereinander, separiert durch ein Leerzeichen, geschrieben werden. Die Reihenfolge ist beliebig. Ein Spielstein wird durch eine `int`-Zahl n, wobei gilt $0 \leq n \leq 15$, repräsentiert.

A.3.6 `rowprint`-Befehl

Der `rowprint`-Befehl gibt eine bestimmte Zeile des Spielfelds auf die Konsole aus.

Eingabeformat `rowprint <Zeilennummer>`

Ausgabeformat Die Felder der angegebenen Zeile werden aufsteigend nach ihrer Spaltennummer in einer einzigen Zeile ausgegeben, indem alle Felder, separiert durch genau ein Leerzeichen, hintereinander geschrieben werden. Für ein leeres Feld wird `#` ausgegeben. Befindet sich auf dem Feld ein Spielstein, wird die zum Spielstein passende Zahl ausgegeben.

Beispielausgabe `# 5 # # # #`

A.3.7 `colprint`-Befehl

Der `colprint`-Befehl gibt eine bestimmte Spalte des Spielfelds auf die Konsole aus.

Eingabeformat `colprint <Spaltennummer>`

Ausgabeformat Die Felder der angegebenen Spalte werden aufsteigend nach ihrer Zeilennummer in einer einzigen Zeile ausgegeben, indem alle Felder, separiert durch genau ein Leerzeichen, hintereinander geschrieben werden. Für ein leeres Feld wird `#` ausgegeben. Befindet sich auf dem Feld ein Spielstein, wird die zum Spielstein passende Zahl ausgegeben.

Beispielausgabe `# # # # 5 #`

A.3.8 `quit`-Befehl

Der parameterlose Befehl ermöglicht es, das Brettspiel vollständig zu beenden. Beachten Sie, dass hierfür keine Methoden wie `System.exit()` oder `Runtime.exit()` verwendet werden dürfen. Hierbei findet keine Konsolenausgabe statt.

Eingabeformat `quit`

A.3.9 Beispielinteraktion

Beachten Sie auch, dass bei den folgenden Beispielabläufen die Eingabezeilen mit dem `>`-Zeichen gefolgt von einem Leerzeichen eingeleitet werden. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabezeilen.

```
> start torus
OK
> select 11
OK
> place 1;4
OK
> select 7
OK
> place 3;0
OK
> select 5
OK
> place 4;1
OK
> select 15
OK
> place 2;5
P1 wins
3
> rowprint 4
# 5 # # # #
> quit
```