Karlsruher Institut für Technologie

Programmieren – Wintersemester 2015/15

Software-Design und Qualität (SDQ)
`https://sdqweb.ipd.kit.edu/wiki/Programmieren`
Prof. Ralf Reussner · Kiana Rostami · Michael Langhammer

Abschlussaufgabe 2

Ausgabe: 25.02.2016 – 13:00
Abgabe: 24.03.2016 – 13:00

Bearbeitungshinweise

- Achten Sie darauf nicht zu lange Zeilen, Methoden und Dateien zu erstellen[1]
- Programmcode muss in englischer Sprache verfasst sein
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete `java.lang`, `java.io` und `java.util`, 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 zu Javadoc-Dokumentation ein[1]
- Nutzen Sie nicht das default-Package[1]
- Halten Sie auch alle anderen Checkstyle-Regeln ein

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Donnerstag, den 10. März**, freigeschaltet. Geben Sie die Java-Klassen als `*.java`-Dateien ab. Laden Sie die Terminal-Klasse nicht mit hoch.

Planen Sie für die Abgabe ausreichend Zeit ein, sollte der Praktomat Ihre Abgabe wegen einer Regelverletzung ablehnen.

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 inbesondere auch darauf, dass unbehandelte `RuntimeExceptions`, bzw. Subklassen davon—sogenannte *Unchecked Exceptions*—nicht zum Abbruch des Programms führen.

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

Terminal-Klasse

Laden Sie für **diese Aufgabe** die `Terminal`-Klasse[a] von unserer Homepage 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 `Terminal.printLine()` ausgegeben und müssen aus technischen Gründen unbedingt mit `Error,` beginnen.
Laden Sie die Terminal-Klasse niemals zusammen mit Ihrer Abgabe hoch.

[a] https://sdqweb.ipd.kit.edu/lehre/WS1516-Programmieren/Terminal.java

Brettspiel (20 Punkte)

In dieser Aufgabe implementieren Sie ein Brettspiel, bei dem zwei Spieler abwechselnd ihre Spielsteine auf die Zellen eines 6x6 Spielbrettes platzieren. Es gibt 16 Spielsteine, worauf beide Spieler Zugriff haben. Der erste Spieler beginnt immer das Spiel. Der Spieler, der als erster 4 Spielsteine, die alle mindestens eine gemeinsame Eigenschaft haben, in eine Zeile, Spalte oder Diagonale platzieren kann, gewinnt.

Implementieren Sie dieses Brettspiel wie nachfolgend beschrieben.

A 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 Abbildung 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.

c_{i-1j-1}	c_{i-1j}	c_{i-1j+1}
c_{ij-1}	c_{ij}	c_{ij+1}
c_{i+1j-1}	c_{i+1j}	c_{i+1j+1}

Abbildung 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. 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.

Somit haben keine zwei Spielsteine identische Eigenschaften.

B 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.

B.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. Somit kann nur der Spieler, der in einem Zug den Spielstein platziert, gewinnen. Die Züge sind in dieser Aufgabe beginnend mit 0 durchnummeriert.

B.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.

B.3 Torusspielfeld

Bei einem Torusspielfeld sind die äußersten Zellen in der Nachbarschaft der gegenüber liegenden äußersten Zellen. 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 2 illustriert beispielhalft ein Torusspielfeld mit 4 Zeilen und Spalten.

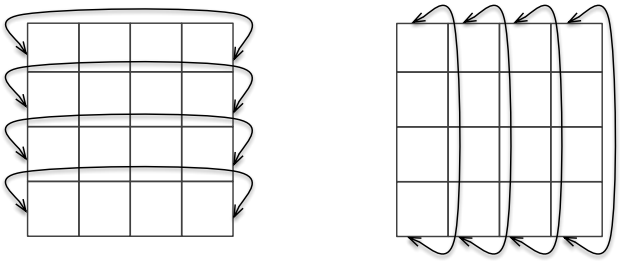

Abbildung 2: Abbildung eines exemplarischen 4x4 Spielfeldes mit einem torusartigen Rahmen

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 $6 - (|i|\%6)$ und $6 - (|j|\%6)$ platziert. $|i|$ bezeichnet den Betrag der Zahl i.

Bei einem torusartigen Spielfeld gewinnt somit der Spieler, der als erster 4 Spielsteine, die alle mindestens eine gemeinsame Eigenschaft haben, in eine Zeile, Spalte oder Diagonale auch über den Spielrahmen hinweg platziert hat. Abbildung 3 illustriert diesen Fall anhand eines Beispiels.

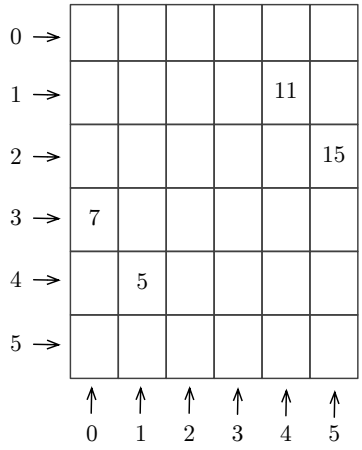

Abbildung 3: Beispiel eines Spielstandes mit einem Torusspielfeld, in dem der erste Spieler gewonnen hat, da alle Spielsteine massiv sind.

C Kommandozeilenargumente

Ihr Programm nimmt als erstes und einziges Kommandozeilenargument entweder `standard` oder `torus` entgegen. Dabei bedeutet `standard`, dass ein Standardspielfeld für das Spiel eingesetzt wird und `torus`, dass ein Torusspielfeld verwendet wird.

Tritt beim Verarbeiten des Kommandozeilenargumentes ein Fehler auf, so wird eine Fehlermeldung ausgegeben und das Programm beendet sich mittels `System.exit(1)`.

D Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole mittels `Terminal.readLine()` 6 Befehle entgegen. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm irgendwann durch `quit` beendet wird.

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.

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Spielregeln nicht verletzt werden und geben Sie ansonsten eine Fehlermeldung aus. Beispiel: Dass der Spieler den Spielstein bei einem Standardspielfeld außerhalb des Spielfeldes platziert, ist ein Beispiel für die Verletzung der Spielregeln.

D.1 `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>`

`<Spielstein>` ist eine `Integer`-Zahl n, wobei gilt $0 \leq n \leq 15$, siehe hierzu Abschnitt A. Diese Zahl repräsentiert einen auf dem Spielfeld zu platzierenden Spielstein, der zuvor aus dem Vorrat entnommen wird.

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

D.2 `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>`

`<Zeilennummer>` und `<Spaltennummer>` sind `Integer`-Zahlen n und m, wobei gilt $0 \leq n \leq 5$ und $0 \leq m \leq 5$ und bezeichnen die Zelle, auf welcher der erste Spielstein der `<Spielstein-Liste>` platziert werden soll. Beachten Sie für die Zellen außerhalb des Spielfeldes den Abschnitt B.1.

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 (z.B., 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.

Beispielausgabe

```
P1 wins
3
```

D.3 `bag`-Befehl

Der `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 `Integer`-Zahl n, wobei gilt $0 \leq n \leq 15$, repräsentiert.

D.4 `rowprint`-Befehl

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

Eingabeformat `rowprint <Zeilennummer>`

`<Zeilennummer>` ist eine ganze Zahl zwischen 0 und 5 und bezeichnet die auszugebende 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 # # # #`

D.5 `colprint`-Befehl

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

Eingabeformat `colprint <Spaltennummer>`

`<Spaltennummer>` ist eine ganze Zahl zwischen 0 und 5 und bezeichnet die auszugebende 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 #`

D.6 `quit`-Befehl

Dieser Befehl beendet das Programm. Dabei findet keine Konsolenausgabe statt.

D.7 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 [2].

java BoardGame torus

```
> 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
```

[2] Der Klassenname `BoardGame` ist nicht vorgeschrieben.