Karlsruher Institut für Technologie

Programmieren – Wintersemester 2017/18

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

Abschlussaufgabe 1

Ausgabe: 09.02.2018 – 13:00 Uhr
Abgabe: 10.03.2018 – 06:00 Uhr

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. Die Kommentare sollen einheitlich entweder in englischer oder in 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.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 zur Javadoc-Dokumentation ein.[1]
- Nutzen Sie nicht das default-Package.[1]
- Halten Sie auch alle anderen Checkstyle-Regeln ein.
  Prinzipiell kann ein Nichteinhalten der Checkstyle-Regeln zu Punktabzug führen.
- `System.exit` und `Runtime.exit` dürfen nicht verwendet werden.[1]
- Achten Sie darauf, genau die vorgegebenen Ein- und Ausgabeformate einzuhalten.
- Allgemeiner Hinweis: Bei Regelverletzung wird der Praktomat Ihre Abgabe zurückweisen.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Freitag, den 23.02.2018, um 13:00 Uhr** freigeschaltet. Laden Sie die `Terminal`-Klasse nicht mit hoch.

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

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

Planen Sie für die Abgabe ausreichend Zeit ein, für den Fall, dass der Praktomat Ihre Abgabe wegen einer Regelverletzung ablehnen sollte. Hinweis: Beachten Sie, dass die öffentlichen Tests für eine erfolgreiche Abgabe erfüllt sein müssen.

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.

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.

A Brettspiel – Connect6

In dieser Aufgabe soll eine Variante des Brettspiels *Connect6*[2], das 2005 auf der Konferenz *Advances in Computer Games* vorgestellt wurde, implementiert werden.

Implementieren Sie dieses Brettspiel wie nachfolgend beschrieben.

A.1 Spielaufbau und -regeln

Spieler der Anzahl p (mit $1 < p < 5$) treten gegeneinander an und legen reihum in jedem gültigen Zug zwei ihrer Spielsteine, die mit `P1` für Spieler 1, `P2` für Spieler 2, `P3` für Spieler 3 bzw. `P4` für Spieler 4 zur Unterscheidung markiert sind, auf die Felder eines quadratischen N×N-Spielbrettes. Auf jedem freien Feld darf maximal ein Spielstein platziert werden. Die Anzahl der Spielsteine eines Spielers ist nicht beschränkt. Der Spieler, der als erster mindestens sechs seiner Zeichen in einer ununterbrochenen Reihe - in horizontaler, vertikaler oder diagonaler Richtung - anordnen kann, gewinnt das Spiel.

Ein Feld $c_{i,j}$, das sich nicht am Rand des Spielbrettes befindet, besitzt 8 angrenzende Felder. Abbildung 1 illustriert diesen Sachverhalt. Felder am Rande des Spielbrettes besitzen weniger angrenzende Felder. Dabei bezeichnet i die Zeilennummer und j die Spaltennummer eines Feldes.

[2] Wu IC., Huang DY. (2006) A New Family of k-in-a-Row Games. In: van den Herik H.J., Hsu SC., Hsu T., Donkers H.H.L.M.. (eds) Advances in Computer Games. ACG 2005. Lecture Notes in Computer Science, vol 4250. Springer, Berlin, Heidelberg

Zeilennummer (zählweise von oben nach unten) und Spaltennummer (zählweise von links nach rechts) beginnen immer bei 0. Somit ist $c_{0,0}$ das oberste linke Feld im Spielbrett. Das Spielbrett ist zu Beginn des Spiels leer; keine Steine belegen die Felder.

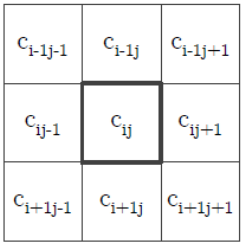

Abbildung 1: Ein Feld $c_{i,j}$ innerhalb des Spielbrettes mit den angrenzenden Feldern

Ein Spieldurchgang kann entweder mit einem Standardspielbrett (eingeleitet mit dem Kommandozeilenargument `standard`, siehe A.2) oder mit einem Torusspielbrett (eingeleitet mit dem Kommandozeilenargument `torus`, siehe A.2) durchgeführt werden.
Bei einem Standardspielbrett sind die Ränder des Spielbrettes abgeschlossen. Keiner der Spieler darf seinen Spielstein außerhalb des Spielbrettes platzieren. Anderfalls gilt dies als eine Spielregelverletzung.

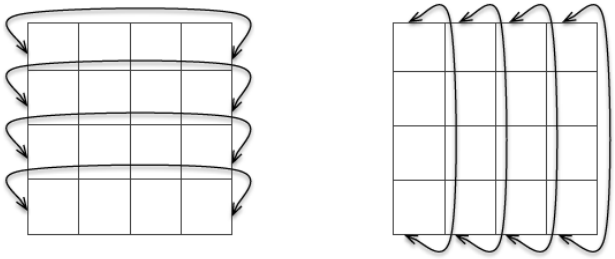

Abbildung 2: Ein 4×4 Spielbrett mit einem torusartigen Rahmen

Bei einem Torusspielbrett sind die äußersten Felder in der Nachbarschaft der gegenüber liegenden äußersten Felder. Stellt man sich das Spielbrett analog zu einer Matrix $A = a_{i,j}$ mit insgesamt m Zeilen und n Spalten – beginnend mit 0 durchnummeriert – vor, dann sind die Ränder des Spielbrettes wie folgt definiert:
$a_{-1,j} = a_{m-1,j}, a_{i,-1} = a_{i,n-1}$, $a_{m,j} = a_{0,j}$ und $a_{i,n} = a_{i,0}$.
Abbildung 2 illustriert beispielhaft ein Torusspielbrett mit 4 Zeilen und 4 Spalten.
Die Ränder des Spielbrettes sind offen. Versucht ein Spieler seinen Spielstein außerhalb des Spielbrettes auf das Feld mit Zeilennummer i und Spaltennummer j (mit i < 0 oder i > N-1, j < 0 oder j > N-1) zu platzieren, wird der Spielstein automatisch auf das Feld i%N und j%N gesetzt.
Bei einem torusartigen Spielfeld gewinnt somit der Spieler, der als erster mindestens sechs seiner Spielsteine in eine Zeile, Spalte oder Diagonale – auch über den Spielrahmen hinweg – platziert hat.

A.2 Kommandozeilenargumente

Ihr Programm nimmt als erstes Kommandozeilenargument entweder `standard` oder `torus` entgegen. Das Argument `standard` legt fest, dass ein Standardspielbrett für das Spiel eingesetzt wird und das Argument `torus`, dass ein Torusspielbrett verwendet wird.
Das zweite Kommandozeilenargument entspricht der Anzahl N der Zeilen und Spalten des quadratischen Spielbrettes N×N mit $N \in \mathbb{N}$, $17 < N < 21$ und N durch zwei teilbar.
Das dritte Kommandozeilenargument entspricht der Anzahl p der Spieler, wobei gilt $p \in \mathbb{N}$ und $1 < p < 5$.

Bitte beachten Sie, dass alle Kommandozeilenargumente vorhanden sein müssen. Tritt beim Verarbeiten der Kommandozeilenargumente ein Fehler auf bzw. entspricht die Eingabe nicht dem hier spezifizierten Format, so wird eine aussagekräftige Fehlermeldung ausgegeben und das Programm wird beendet.

A.3 Interaktive Benutzerschnittstelle

Ihr Programm arbeitet ausschließlich mit Befehlen, die nach dem Programmstart über die Konsole mittels `Terminal.readLine()` eingelesen werden. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm durch die Eingabe des `quit`-Befehls beendet wird.
Direkt nach Programmstart ist Spieler 1 aktiv bzw. an der Reihe. In jedem Zug gibt es immer genau einen aktiven Spieler. Die Spieler sind in aufsteigender Reihenfolge aktiv bis einer der Spieler gewinnt.

A.4 Befehle

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Spielregeln nicht verletzt werden und geben Sie ansonsten eine Fehlermeldung aus. Beispiel: Setzt ein Spieler im Modus `standard` seinen Spielstein außerhalb des Spielbrettes, so gilt diese Handlung als Spielregelverletzung. Die folgenden Befehle sind sowohl im Modus `standard`, als auch `torus` anwendbar. Im Folgenden wird die Ausgabe für einen Erfolgsfall spezifiziert. Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben) muss eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben werden.

Hinweise zu den Beispielen

Beachten Sie, dass bei den folgenden Beispielen 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 Ausgabe.

A.4.1 `place`-Befehl

Mit dem `place`-Befehl werden immer zwei Spielsteine des aktiven Spielers auf ein freies Feld des N×N-Spielbrettes platziert. Nach einem `place`-Befehl terminiert Ihr Programm nicht.

Eingabeformat:

```
place <Zeilennummer1>;<Spaltennummer1>;<Zeilennummer2>;<Spaltennummer2>
```

`<Zeilennummer1>;<Spaltennummer1>` und `<Zeilennummer2>;<Spaltennummer2>` sind jeweils Tupelangaben und bezeichnen die Koordinaten des Feldes, auf dem die Spielsteine des aktiven Spielers gesetzt werden sollen. Ein Feld darf prinzipiell nur einmal belegt werden. Ein Spieler ist so lange aktiv bzw. an der Reihe bis er einen gültigen Zug gespielt hat.

Ausgabeformat:
Für das Ausgabeformat ergeben sich folgende Fallunterscheidungen:

- Wenn der Spielzug erfolgreich durchgeführt wurde u. keiner der Spieler gewonnen hat, wird `OK` ausgegeben.
- Wenn das Spiel unentschieden durch diesen Spielzug (d.h. alle Spielfelder des Spielbrettes sind mit Steinen belegt) ausgeht, wird `draw` ausgegeben.
- Gewinnt einer der Spieler in diesem Zug, wird `P1 wins` ausgegeben, falls Spieler 1 gewinnt; `P2 wins` wird ausgegeben, falls Spieler 2 gewinnt usw.
- Im Fehlerfall (z.B. wenn das Spielfeld bereits durch einen Spielstein besetzt ist) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Des Weiteren ist dieser Befehl in den folgenden Fällen unzulässig:
(i) Nachdem ein Spieler anhand eines vorherigen `place`-Befehls das Spiel gewonnen hat.
(ii) Nachdem alle Spielfelder auf dem Spielbrett anhand eines vorherigen `place`-Befehls belegt sind. Es existiert somit kein freies Feld auf dem Spielbrett mehr.
Bemerkung: Alle anderen Befehle dürfen jedoch in diesen beiden Fällen ausgeführt werden.

Beispielablauf für Modus `standard 18 2`

```
> place 4;3;6;1
OK
> place 6;4;5;2
OK
```

A.4.2 `rowprint`-Befehl

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

Eingabeformat:

```
rowprint <Zeilennummer>
```

`<Zeilennummer>` ist eine ganze Zahl m mit $0 \leq$ m $\leq N-1$ und bezeichnet die Zeilennummer der Zeile des Spielbrettes, die ausgegeben werden soll.

Ausgabeformat:
Die Felder der angegebenen Zeile werden aufsteigend nach ihrer Spaltennummer in einer einzigen Zeile ausgegeben, indem alle Felder, die jeweils durch exakt ein Leerzeichen separiert werden, hintereinander geschrieben werden. Für ein leeres Feld wird ** ausgegeben. Befindet sich auf dem Feld ein Spielstein eines Spielers, wird entsprechend der Markierung { `P1`, `P2`, `P3`, `P4` } ausgegeben.
Im Fehlerfall (z.B. wenn die Zeilennummer negativ oder größer als die Anzahl der Zeilen und Spalten ist) wird eine aussagekräftige Fehlermeldung ausgegeben.

Beispielablauf mit N = 18 und p = 2

```
> rowprint 5
** ** ** ** ** ** P2 ** ** ** ** ** P1 ** P2 P1 ** P1
```

A.4.3 `colprint`-Befehl

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

Eingabeformat:

```
colprint <Spaltennummer>
```

`<Spaltennummer>` ist eine ganze Zahl n mit $0 \leq$ n $\leq N-1$ und bezeichnet die Spaltennummer der Spalte des Spielbrettes, die ausgegeben werden soll.

Ausgabeformat:
Die Felder der angegebenen Spalte werden aufsteigend nach ihrer Zeilennummer in einer einzigen Zeile ausgegeben, indem alle Felder, die jeweils durch exakt ein Leerzeichen separiert werden, hintereinander geschrieben werden. Für ein leeres Feld wird ** ausgegeben. Befindet sich auf dem Feld ein Spielstein eines Spielers, wird entsprechend der Markierung { `P1`, `P2`, `P3`, `P4` } ausgegeben.
Im Fehlerfall (z.B. wenn die Spaltennummer negativ oder größer als die Anzahl der Zeilen und Spalten ist) wird eine aussagekräftige Fehlermeldung ausgegeben.

Beispielablauf mit N = 18 und p = 2

```
> colprint 3
** ** ** P1 ** ** ** ** ** ** ** ** P1 P2 P2 ** P1 P1
```

A.4.4 `print`-Befehl

Der `print`-Befehl gibt das aktuelle Spielbrett mit den Spielsteinbelegungen aus. Es findet keine weitere Ausgabe statt.

```
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 Leerzeichen getrennt. Es existieren folgende Fallunterscheidungen:

- Ein leeres Feld wird durch ** dargestellt.
- Befindet sich auf einem Feld ein Spielstein eines Spielers x, wird `Px` ausgegeben.
  Es gilt x $\in$ {1, 2, 3, 4}, wobei x abhängig von der Anzahl der Spieler ist. Hinweis: Wird anhand eines `place`-Befehls das Spiel mit der ersten Tupeleingabe gewonnen, werden auch beide Tupeleingaben bei den jeweiligen relevanten `print`-Befehlen ausgegeben.

Beispielablauf mit N = 18 und p = 2

```
> print
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** P1 ** ** ** ** ** ** ** ** P1 P2 ** ** ** ** **
** ** ** ** ** ** P1 ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** P2 ** ** **
** ** ** ** ** ** ** ** ** P2 ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** P1 ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** P2 ** ** ** ** ** ** ** ** **
** ** ** ** ** P1 ** ** ** ** P2 ** P2 ** ** ** ** ** **
** ** ** ** P1 ** ** ** ** ** ** P2 ** ** ** ** ** ** **
** ** ** ** ** ** ** ** P1 ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
```

A.4.5 `state`-Befehl

Der Befehl `state <Zeilennummer>;<Spaltennummer>` gibt den Zustand eines Feldes auf dem Spielbrett an. Die Zeilen- und Spaltennummern sind durch ein Semikolon separiert. Dieser Befehl ist sowohl im Modus `standard`, als auch `torus` anwendbar.

Eingabeformat:

```
state <Zeilennummer>;<Spaltennummer>
```

Ausgabeformat:
Ein Feld auf dem Spielbrett kann sich in einem der folgenden Zustände befinden:
(i) Handelt es sich um ein leeres Feld, wird dies durch ** dargestellt.
(ii) Befindet sich auf dem angegebenen Feld ein Spielstein eines Spielers x, wird `Px` ausgegeben.
Es gilt x $\in$ { 1, 2, 3, 4 }, wobei x abhängig von der Anzahl der Spieler ist.

Beispielablauf

```
> state 3;4
P1
```

A.4.6 `reset`-Befehl

Der Befehl `reset` initialisiert das Brettspiel in dem ausgewählten Spielmodus neu. Im folgenden Zug fängt wieder Spieler 1 mit der Platzierung seiner Steine an.

Eingabeformat:

```
reset
```

Ausgabeformat:

```
OK
```

`OK` lautet die Ausgabe zu diesem Befehl.

Beispielablauf

```
> reset
OK
```

A.4.7 `quit` -Befehl

Dieser Befehl beendet das Programm. Dabei findet keine Konsolenausgabe statt.
Hinweis: Denken Sie daran, dass hierfür keine Methoden wie `System.exit` oder `Runtime.exit` verwendet werden dürfen.

Beispielablauf

```
> quit
```

A.4.8 Beispielinteraktion

Beachten Sie im Folgenden, dass 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 Ausgabe. Der Klassenname `BoardGame` ist nicht vorgeschrieben.

java BoardGame standard 18 2

```
> place 6;3;6;8
OK
> place 3;2;1;7
OK
> place 6;4;6;7
OK
> reset
OK
> place 6;3;6;8
OK
> place 3;2;1;7
OK
> place 6;4;6;7
OK
> place 6;9;6;2
OK
> place 6;5;6;6
P1 wins
> rowprint 6
** ** P2 P1 P1 P1 P1 P1 P1 P2 ** ** ** ** ** ** ** **
> quit
```