Karlsruher Institut für Technologie

Programmieren – Sommersemester 2018

Architecture-driven Requirements Engineering (ARE)
`https://sdqweb.ipd.kit.edu/wiki/Übung_Programmieren_SS18`
Jun.-Prof. Dr.-Ing. Anne Koziolek

Abschlussaufgabe 1

Ausgabe: 06.07.2018 – 13:00 Uhr
Abgabe: 04.08.2018 – 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 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.io`, 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.
- **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**. Es werden nur rein selbstständig erarbeitete Lösungen akzeptiert. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Freitag, den 20.07.2018 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.

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 (20 Punkte)

Bei dem Strategiespiel *Vier gewinnt* lassen zwei Spieler ihre Spielsteine immer abwechselnd in ein senkrecht stehendes 8×8 Spielbrett fallen. Die Spielsteine sind entweder mit der Markierung `P1` für Spieler 1 oder `P2` für Spieler 2 versehen. Die Anzahl der Spielsteine ist beschränkt.

Spieler 1 beginnt immer dieses Strategiespiel. Der Spieler, der als erster mindestens 4 seiner Zeichen in eine Zeile, Spalte oder Diagonale platzieren kann, gewinnt. In dieser Aufgabe sind alle Spalten des Spielfeldes eindeutig von links nach rechts – beginnend mit 0 – durchnummeriert.

Abbildung 1 illustriert 3 Beispiele, bei denen Spieler 1 gewinnt.

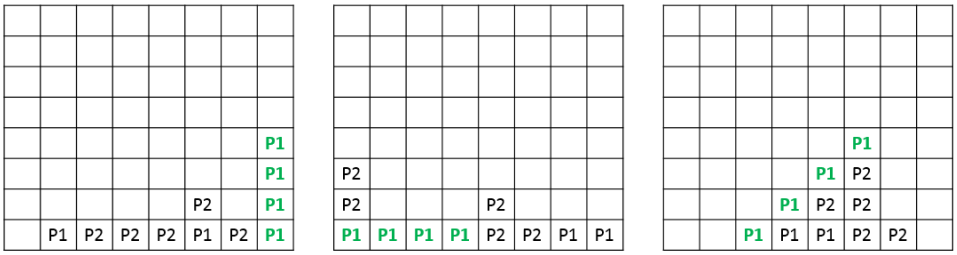

Abbildung 1: Beispiele für Gewinnsituationen

Implementieren Sie dieses Brettspiel wie nachfolgend beschrieben.

B Kommandozeilenargumente

Ihr Programm nimmt als erstes Kommandozeilenargument `standard`, `flip` oder `remove` entgegen. Diese Argumente legen die unterschiedlichen Spielvarianten des Brettspiels *Vier gewinnt* fest. Das zweite Kommandozeilenargument entspricht der Anzahl der Spielsteine p für jeden Spieler mit $p \in \mathbb{N}$ und $27 < p < 33$.
Tritt beim Verarbeiten der Kommandozeilenargumente ein Fehler auf bzw. entspricht die Eingabe nicht dem hier spezifizierten Format, so wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben und das Programm wird beendet.

C 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 und Festlegung des Spielmodus ist Spieler 1 aktiv bzw. an der Reihe. Nach jedem gültigen Zug (siehe `throwin`-Befehl in Abschnitt C.1, `flip`-Befehl in Abschnitt C.2, `remove`-Befehl in Abschnitt C.3) ist der jeweils andere Spieler an der Reihe.

Im Zuge dieser Aufgabenstellung werden Beispielinteraktionen zu Illustrationszwecken angegeben. Beachten Sie, dass bei diesen Beispielen 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. Im Folgenden wird die Ausgabe für einen Erfolgsfall spezifiziert. Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben, Spielregelverletzung usw.) muss eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben werden. Der Klassenname `BoardGame` ist nicht vorgeschrieben.

C.1 Spielmodus `standard`

Mit dem `throwin`-Befehl kann ein Spieler seinen Stein mit einer Markierung $\in$ {`P1`, `P2`} in eine Spalte mit einer Nummer `<value>` $\in \{0, 1, 2, 3, 4, 5, 6, 7\}$ fallen lassen. Nach einem gültigen Zug ist der jeweils andere Spieler an der Reihe.

Eingabeformat:

```
throwin <value>
```

Ausgabeformat:
Für das Ausgabeformat ergeben sich folgende Fallunterscheidungen:

- Gewinnt durch diesen Zug weder einer der Spieler, noch geht das Spiel unentschieden aus, wird nach einem erfolgreichen Zug `OK` ausgegeben.
- Gewinnt Spieler `P1` oder Spieler `P2`, wird nach diesem Zug `Px wins` mit x $\in \{1, 2\}$ ausgegeben.
- Geht das Spiel durch diesen Zug unentschieden aus, wird `draw` ausgegeben. Dies ist der Fall wenn entweder alle Felder auf dem Spielbrett belegt sind oder wenn der danach folgende Spieler keine Steine mehr besitzt und keiner der Spieler mindestens vier seiner Zeichen in einer Reihe hat.
- Ist der Zug ungültig, wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben. Derselbe Spieler kann nach einem ungültigen Zug erneut einen Zug durchführen.

Nach einem `throwin`-Befehl terminiert Ihr Programm nicht.

Dieser Befehl ist in folgenden Fällen nicht zulässig:

- nachdem ein Spieler das Spiel gewonnen hat,
- nachdem das Spiel unentschieden ausgegangen ist,
- ~~wenn ein Spieler keine Spielsteine mehr besitzt.~~

Beispielablauf für `java BoardGame standard 32`

```
> throwin 1
OK
> throwin 5
OK
```

C.2 Spielmodus `flip`

Im diesem Spielmodus hat ein aktiver Spieler die Möglichkeit das Spielbrett mit Hilfe des `flip`-Befehls zu drehen. Die Anzahl der noch vorhandenen Spielsteine des aktiven Spielers bleibt gleich und wird nicht um eins verringert. Nach einem gültigen Zug ist der jeweils andere Spieler an der Reihe.
Der `throwin`-Befehl ist ebenfalls in diesem Spielmodus erlaubt; der `remove`-Befehl (siehe C.3) jedoch nicht.

Eingabeformat:

```
flip
```

Ausgabeformat:
Es existieren folgende Fallunterscheidungen:

- Gewinnt durch diesen Zug weder einer der Spieler, noch geht das Spiel unentschieden aus, wird `OK` ausgegeben.
- Gewinnt einer der Spieler durch diesen Zug, wird `Px wins` mit x $\in \{1, 2\}$ ausgegeben. Werden durch diesen Zug mehrere Reihen aus mindestens vier gleicher Spielsteine gebildet, so gewinnt der Spieler, der die höhere Anzahl an Gewinnerreihen seiner Spielsteine hat.
- Haben beide Spieler durch diesen Zug jeweils eine gleiche Anzahl an Gewinnerreihen (mindestens vier ihrer Spielsteine in einer Reihe), so wird `draw` ausgegeben. Das Spiel geht in diesem Fall unentschieden aus. Besitzt der nachfolgende Spieler keine Spielsteine mehr und keiner der Spieler hat das Spiel gewonnen, so geht das Spiel ebenfalls unentschieden aus.

Die Abbildung 2 illustriert das Spielbrett vor einem `flip`-Befehl (i) und nach einem `flip`-Befehl (ii).

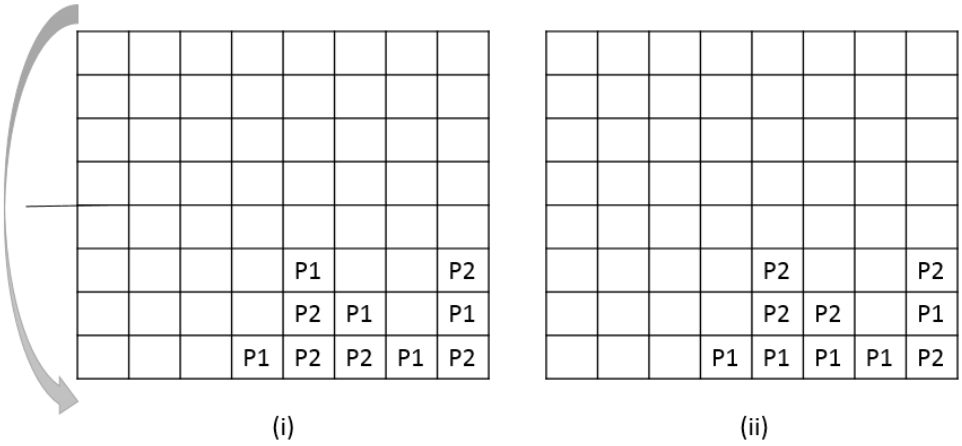

Abbildung 2: Spielbrett vor (i) und nach (ii) einem `flip`-Befehl

Nach einem `flip`-Befehl terminiert Ihr Programm nicht.

Dieser Befehl ist in folgenden Fällen nicht zulässig:

- nachdem ein Spieler das Spiel gewonnen hat,

- nachdem das Spiel unentschieden ausgegangen ist,
- ~~wenn ein Spieler keine Spielsteine mehr besitzt.~~

C.3 Spielmodus `remove`

Im Spielmodus `remove` hat ein aktiver Spieler die Möglichkeit einen eigenen Spielstein aus der untersten Reihe zu entfernen. Mit diesem Befehl werden alle weiteren Spielsteine in dieser Spalte nach unten fallen gelassen. Der entnommene Spielstein darf nicht für einen weiteren Zug verwendet werden. Nach einem gültigen Zug ist der jeweils andere Spieler an der Reihe.
Der `throwin`-Befehl ist ebenfalls in diesem Spielmodus erlaubt; der `flip`-Befehl jedoch nicht.

Eingabeformat:

```
remove <value>
```

Ausgabeformat:
Es existieren folgende Fallunterscheidungen:

- Wird der Spielstein aus der untersten Reihe mit Spaltennummer `<value>` $\in \{0, 1, 2, 3, 4, 5, 6, 7\}$ des aktiven Spielers entfernt und gewinnt durch diesen Zug kein Spieler, wird `OK` ausgegeben.
- Wird der Spielstein aus der untersten Reihe mit Spaltennummer `<value>` $\in \{0, 1, 2, 3, 4, 5, 6, 7\}$) des aktiven Spielers entfernt und gewinnt durch diesen Zug ein Spieler `Px` mit x $\in \{1, 2\}$, wird `Px wins` ausgegeben.
- Wird der Spielstein aus der untersten Reihe mit Spaltennummer `<value>` $\in \{0, 1, 2, 3, 4, 5, 6, 7\}$) des aktiven Spielers entfernt und haben beide Spieler durch diesen Zug mindestens vier ihrer Spielsteine in einer horizontalen, vertikalen oder diagonalen Reihe, wird `draw` ausgegeben. Die jeweilige Anzahl und Längen der Reihen ist in diesem Fall unerheblich. Besitzt der nachfolgende Spieler keine Spielsteine mehr und keiner der Spieler hat das Spiel gewonnen, so geht das Spiel ebenfalls unentschieden aus.
- Im Fehlerfall (z.B. wenn die angegebene Spaltennummer nicht existiert oder der aktive Spieler einen Spielstein des Kontrahenten entfernen möchte) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Nach einem `remove`-Befehl terminiert Ihr Programm nicht.

Dieser Befehl ist in folgenden Fällen nicht zulässig:

- nachdem ein Spieler das Spiel gewonnen hat,
- nachdem das Spiel unentschieden ausgegangen ist,
- ~~wenn ein Spieler keine Spielsteine mehr besitzt.~~

C.4 Weitere Befehle und Beendigung des Programms

Im Folgenden werden weitere Befehle spezifiziert, die in allen Spielmodi erlaubt sind. Durch diese Befehle werden die Spielsteinbelegungen des Spielbretts nicht modifiziert.

C.4.1 Der `token`-Befehl

Der Befehl `token` gibt die Anzahl der verbleibenden Spielsteine des aktiven Spielers an.

Beispielablauf für `java BoardGame standard 32`

```
> token
3
```

C.4.2 Der `state`-Befehl

Der Befehl `state` gibt den Zustand eines Spielfeldes mit Koordinaten x (Spaltennummer des Feldes beginnend mit 0) und y (Zeilennummer des Feldes beginnend mit 0) aus. Die Zeilen- und Spaltennummern sind durch exakt ein Semikolon separiert. Abbildung 3 illustriert beispielhaft die Markierung des Spielfeldes `3;6`.

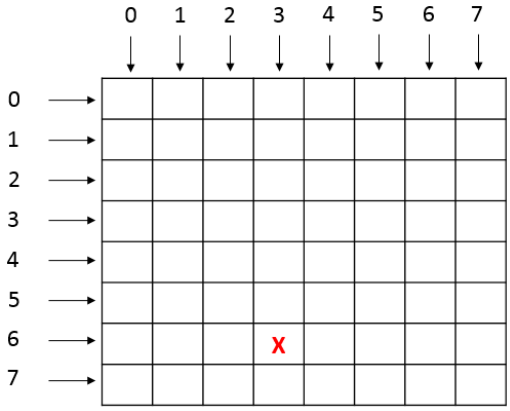

Abbildung 3: Spielbrett mit festgelegten Koordinaten und nummerierten Feldern

Eingabeformat:

```
state x;y
```

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 dem angegebenen Spielfeld ein Spielstein des ersten Spielers, wird `P1` ausgegeben.
- Befindet sich auf dem angegebenen Spielfeld ein Spielstein des zweiten Spielers, wird `P2` ausgegeben.
- Im Fehlerfall (z.B. wenn die angegebenen Koordinaten nicht existieren) wird eine aussagekräftige Fehlermeldung beginnend mit `Error, ` ausgegeben.

Beispielablauf für `java BoardGame standard 32`

```
> state 6;3
P1
```

C.4.3 Der `print`-Befehl

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

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 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 mit x ∈ {1, 2}.

Beispielablauf für `java BoardGame standard 32`

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

C.4.4 Der `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
```

D 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 32

```
> throwin 2
OK
> throwin 3
OK
> throwin 5
OK
> throwin 4
OK
> throwin 3
OK
> throwin 4
OK
> throwin 4
OK
> throwin 7
OK
> throwin 6
OK
> throwin 7
OK
> throwin 5
OK
> state 0;0
**
> throwin 5
OK
> throwin 5
P1 wins
> state 4;7
P2
> print
** ** ** ** ** ** ** **
** ** ** ** ** ** ** **
** ** ** ** ** ** ** **
** ** ** ** ** ** ** **
** ** ** ** ** P1 ** **
** ** ** ** P1 P2 ** **
** ** ** P1 P2 P1 ** P2
** ** P1 P2 P2 P1 P1 P2
> quit
```