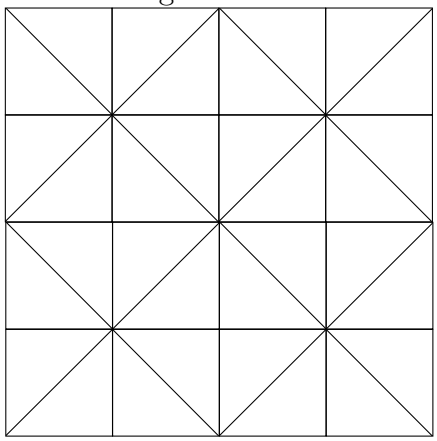
Pflichtaufgabe Informatik Bachelor - Informatik 1 Rechnerübungen - Hochschule Karlsruhe - WS16/17

Abgabe in den Rechnerübungen bei Ihrem Übungsgruppenleiter bis spätestens letzte Vorlesungswoche.

Sie benötigen 25 von den maximal 50 zu erreichenden Punkten. Das Programm muss im jeden Fall voll funktionsfähig, also spielbar, sein.

Aufgabe 1 (Bag Chal, 50 Punkte)

Es gibt zwei Spieler. Der eine bewegt Tiger (Spieler T) und der andere (Spieler Z) Ziegen als Spielfiguren. Das Spielfeld besteht aus zusammenhängenden Kanten. Die Spielfiguren stehen auf den Verbindungsknoten zwischen Kanten.



Es gibt vier Tiger, die zu Beginn des Spiels in den vier Ecken des Spielfelds stehen. Der Spieler mit den Ziegen beginnt. Er hat genau 20 Ziegen.

- Das Spiel ist beendet, wenn Spieler T fünf Ziegen geschlagen (gefressen) hat oder keine Tiger mehr gezogen werden können.
- Spieler T hat gewonnen, wenn fünf Ziegen gefressen wurden.
- Spieler Z hat gewonnen, wenn Spieler T keinen Zug durchführen kann.

Beide Spieler wechseln sich mit den Zügen ab.
Spieler T:

- Er darf einen Tiger entlang einer Kante auf ein benachbartes freies Feld verschieben oder
- er darf über eine Ziege mit einem Tiger entlang zweier Kante springen (fressen). Die Ziege wird aus dem Spiel genommen. Die beiden Kanten müssen die gleiche Ausrichtung haben, es darf also nicht um die Ecke oder die Kurve gesprungen werden.
- Tiger dürfen nicht übersprungen werden.

Spieler Z:

- Da zu anfangs keine Ziegen auf dem Spielfeld stehen, muss Spieler Z zuerst seine Ziegen auf ein freies Feld setzen.
- Erst wenn alle 20 Ziegen gesetzt wurden, darf Spieler Z eine Ziege entlang einer Kante ziehen.
- Ziegen dürfen andere Spielfiguren nicht überspringen.

Die nachfolgenden Seiten am Ende des Übungsblattes zeigen einen Teil eines Spielablauf als Beispiel mit einer Textausgabe.

Das Programm soll objekt-orientiert und nach dem MVC-Paradigma aufgebaut sein.

- Die Klassen des Modells enthalten die Implementierung der Spiellogik: das Spielfeld, aktueller Spieler, Spielzustand sowie Methoden, um das Spiel korrekt zu spielen. Insbesondere werden hier die Spielregeln überprüft. Nicht erlaubte Züge dürfen nicht durchgeführt werden, egal wie oft oder in welcher Reihenfolge die Methoden des Modells aufgerufen werden, zum Beispiel, weil sie von einem anderen Programmierer wiederverwendet wurden, der sich nicht an den Ablauf ihrer Implementierung der Steuerung hält.
- Die Klassen der Sicht enthalten Methoden, um die Daten des Modells auf den Bildschirm auszugeben. Das darzustellende Modell wird als Objekt im Konstruktor dieser Klasse übergeben.
- Die Klassen der Steuerung enthalten die Ablaufsteuerung des Spiels. Hier werden Benutzereingaben entgegengenommen und mit den Methoden des Modells ausgewertet. Für Textaus- und eingaben werden Methoden der Sicht aufgerufen. Modell und Sicht werden als Objekte im Konstruktor übergeben.

Die main-Methode, sollte in einer eigenen Klasse implementiert werden. Hier werden die Objekte erzeugt und der Ablauf des Controllers gestartet. Die main-Methode soll die einzige Klassenmethode sein.

Folgendes ist zu beachten (in Klammern die Verteilung der Punkte):

- Alle öffentlichen Methoden, Variablen und Klassen müssen mit einem kurzen und spezifischen Javadoc-Kommentar versehen werden. (10 Punkte)
- Jede öffentliche Methode des Modells muß mit JUnit getestet werden. Die Spielregeln dürfen durch diese Tests nicht umgangen werden. Implementieren Sie beim Modell keine zusätzlichen Methode oder Konstruktoren, die Sie nur zum Testen benötigen. (10 Punkte)
- Das Programm muss objekt-orientiert, funktional korrekt sein und redundanzfrei sein. (20 Punkte)
- Es muss streng nach dem MVC-Paradigma implementiert sein. (10 Punkte)

Zur Eingabe der Zahlen verwenden Sie bitte die Klasse Eingabe.java (Ilias) oder – wie in der Vorlesung gezeigt – die Klasse Scanner.

Hilfen und Hinweise

- Für den Entwurf der Klassen des Modells stellt man sich am besten eine Schiedsrichter vor, der auf Zuruf, das Spiel für die beiden Spieler spielt. Ebenso stelle man sich vor, dass das Spielbrett von den Spielern nicht direkt gesehen wird: die Spieler müssen die Information vom Schiedsrichter erfragen. Diese Zurufe und das Erfragen von Informationen sind Methodenaufrufe.

- Es ist sinnvoll eine Methode zu implementieren, die prüft, ob eine Spieler gemäß den Spielregeln einen Zug durchführen darf. Sie können diese Methode verwenden, um durch Ausprobieren aller potentiellen Züge von Spieler T, herauszufinden, ob Spieler Z gewonnen hat.

- Die erlaubten Züge entlang den Kanten herauszufinden ist algorithmisch etwas schwierig. Es ist einfacher, in einer Tabelle nach zu schauen, ob ein Zug von einer Position u zu einer anderen Position v erlaubt ist. Mit einem zweidimensionalem 25 x 25 Feld *zuege* kann dies wie folgt realisiert werden: Das Feld enthält nur die Werte 0 bis 2. $u = zeile * 5 + spalte$ und $v = zuZeile * 5 + zuSpalte$, wobei $zeile, spalte$ das Ursprungs- und $zuZeile, zuSpalte$ das Zielfeld angeben. In dem Feld $zuege[u][v]$ steht

  - 0, wenn kein Zug von u nach v möglich ist.
  - 1, wenn potentiell ein direkter Zug ohne Sprung möglich ist.
  - 2, wenn potentiell ein Sprung von u nach v möglich ist.

  Diese Tabelle wird Adjazenzmatrix genannt. Sie können folgenden Java-Code für die Matrix verwenden. Beachten Sie, dass die Einträge manuell erstellt sind und vielleicht noch Fehler enthalten können.

Die folgenden Adjazenzmatrix gibt es zum Hochladen im ILIAS.

```
/*
 Adjazenzmatrix für die gültigen Züge dieses
 Spielbrettteils:

 0---1---2---3---4
 |\  |  /|\  |  /|
 | \ | / | \ | / |
 |  \|/  |  \|/  |
 5---6---7---8---9
 |  /|\  |  /|\  |
 | / | \ | / | \ |
 |/  |  \|/  |  \|
 10--11--12-13--14
 |\  |  /|\  |  /|
 | \ | / | \ | / |
 |  \|/  |  \|/  |
 15-16--17--18--19
 |  /|\  |  /|\  |
 | / | \ | / | \ |
 |/  |  \|/  |  \|
 20--21--22-23--24
 */
private static int [] [] zuege =
      { {0, 1, 2, 0, 0, 1, 1, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {1, 0, 1, 2, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {2, 1, 0, 1, 2, 0, 1, 1, 1, 0, 2, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 2, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 2, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {1, 1, 1, 0, 0, 1, 0, 1, 2, 0, 1, 1, 1, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0},
        {0, 0, 1, 0, 0, 2, 1 ,0, 1, 2, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 1, 1, 1, 0, 2, 1, 0, 1, 0, 0, 1, 1, 1, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0},
        {2, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 1, 2, 0, 0, 1, 1, 0, 0, 0, 2, 0, 2, 0, 0},
        {0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0},
        {2, 0, 2, 0, 2, 0, 1, 1, 1, 0, 2, 1, 0, 1, 2, 0, 1, 1, 1, 0, 2, 0, 2, 0, 2},
        {0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0},
        {0, 0, 2, 0, 2, 0, 0, 0, 1, 1, 0, 0, 2, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 0, 2},
        {0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 1, 0, 0, 0, 0},
```

```
        {0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 1, 1, 1, 0, 0, 1, 0, 1, 2, 0, 1, 1, 1, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 1, 2, 0, 0, 1, 0, 0},
        {0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 1, 1, 1, 0, 2, 1, 0, 1, 0, 0, 1, 1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 0, 0, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 1, 2, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 1, 1, 1, 0, 2, 1, 0, 1, 2},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 1, 1, 0, 0, 2, 1, 0}
    };
```

Beispielausgabe

```
T-+-+-+-T
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
+-+-+-+-+
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
T-+-+-+-T
Anzahl gefressener Ziegen : 0
Anzahl zu setzender Ziegen : 20
Spieler Z ist am Zug.
Zeile (Ziel): 2
Spalte (Ziel): 0
T-+-+-+-T
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
Z-+-+-+-+
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
T-+-+-+-T
Anzahl gefressener Ziegen : 0
Anzahl zu setzender Ziegen : 19
Spieler T ist am Zug.
Zeile (Start): 0
```

```
Spalte (Start): 0
Zeile (Ziel): 1
Spalte (Ziel): 1
+-+-+-+-T
|\|/|\|/|
+-T-+-+-+
|/|\|/|\|
Z-+-+-+-+
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
T-+-+-+-T
Anzahl gefressener Ziegen : 0
Anzahl zu setzender Ziegen : 19
Spieler Z ist am Zug.
Zeile (Ziel): 1
Spalte (Ziel): 2
+-+-+-+-T
|\|/|\|/|
+-T-Z-+-+
|/|\|/|\|
Z-+-+-+-+
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
T-+-+-+-T
Anzahl gefressener Ziegen : 0
Anzahl zu setzender Ziegen : 18
Spieler T ist am Zug.
Zeile (Start): 1
Spalte (Start): 1
Zeile (Ziel): 1
Spalte (Ziel): 3
+-+-+-+-T
|\|/|\|/|
+-+-+-T-+
|/|\|/|\|
Z-+-+-+-+
|\|/|\|/|
+-+-+-+-+
|/|\|/|\|
```

```
T-+-+-+-T
Anzahl gefressener Ziegen : 1
Anzahl zu setzender Ziegen : 18
Spieler Z ist am Zug.
Zeile (Ziel):
```

Sie können auch eine kompaktere Eingabe wie 1113 für eine Tigersprung von 1,1 nach 1,3 implementieren.