Prof. Dr. Volker Linnemann
Dr. Sven Groppe
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 23.11.2007

Vorlesung „Programmieren“

Wintersemester 2007/2008

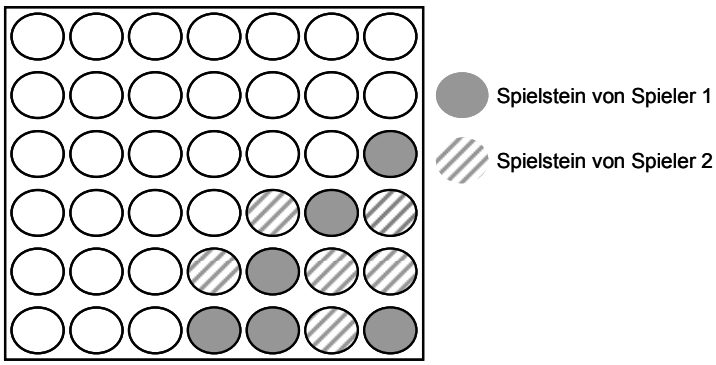
5. Übungsblatt

1. Vier gewinnt (Programmieraufgabe) (40 Punkte)

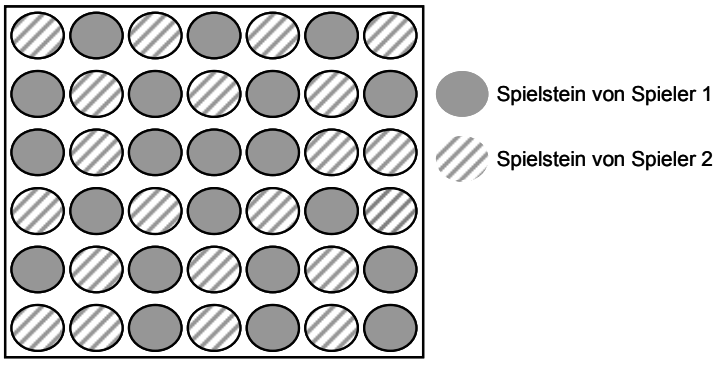
Das Spiel „Vier gewinnt“ wird auf einem senkrecht stehenden hohlen Spielbrett mit sieben Spalten (senkrecht) und sechs Reihen (waagrecht) gespielt, in das zwei Spieler abwechselnd ihre Spielsteine fallen lassen. Wenn ein Spieler einen Spielstein in eine Spalte fallen lässt, besetzt dieser Spielstein den untersten freien Platz der Spalte. Gewinner ist der Spieler, der es als erster schafft, mindestens vier seiner Spielsteine waagerecht, senkrecht oder diagonal in eine Linie zu bringen. Das Spiel endet unentschieden, wenn das Spielbrett komplett gefüllt ist, ohne dass ein Spieler eine Viererlinie gebildet hat.

Beispiele:

In der folgenden Spielsituation hat Spieler 1 gewonnen, da vier Spielsteine von Spieler 1 diagonal zusammenhängen:



In der folgenden Spielsituation endet das Spiel unentschieden:



Aufgabe:

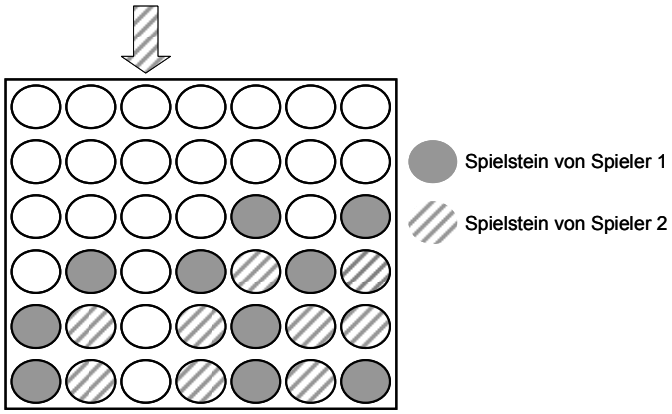
Implementieren Sie eine Java Klasse `ConnectFour`, die das Spiel „Vier gewinnt“ nachbildet.

a) Implementieren Sie eine Methode `void reset()`, die das Spielfeld zurücksetzt für ein neues Spiel. (5 Punkte)
b) Implementieren Sie eine Methode `print()`, die das Spielfeld auf dem Bildschirm ausgibt. (5 Punkte)
c) Implementieren Sie eine Methode `boolean positionWon(int x, int y)`, die `true` zurückgibt, falls der Spielstein in Zeile `x`, Reihe `y` mit anderen Spielsteinen desselben Spielers mindestens 4 zusammenhängende Spielsteine (vertikal, horizontal oder diagonal) ergeben. Ansonsten gibt die Methode falsch zurück. (5 Punkte)
d) Implementieren Sie eine Methode `int endOfGame()`, die 0 zurückgibt, falls das Spiel noch nicht zu Ende ist, die 1 zurückgibt, falls Spieler 1 gewinnt, die 2 zurückgibt, falls Spieler 2 gewinnt und die 3 zurückgibt, falls das Spiel unentschieden endet. (5 Punkte)
e) Implementieren Sie eine Methode `int dropDisc(int x, int number)`, die den Einwurf eines Spielsteines des Spielers mit der Nummer `number` in der Reihe `x` simuliert. (5 Punkte)
f) Implementieren Sie eine Methode `void nextRoundPlayer(int number)`, die die Runde eines menschlichen Spielers mit Nummer `number` simuliert, in dem die Spalte von der Tastatur eingelesen wird, in der der menschliche Spieler mit Nummer `number` seinen Spielstein einwirft. (5 Punkte)
g) Implementieren Sie eine Methode `void nextRoundComputer(int number)`, die die Runde eines Computerspielers mit Nummer `number` simuliert. Der Computergegner soll die Spalte, in die er seinen Spielstein einwirft, folgendermaßen wählen:

i) Falls das Gewinnen des anderen Spielers durch Einwurf eines Spielsteines in eine bestimmte Spalte verhindert wird, so soll diese Spalte gewählt werden.
ii) Ansonsten soll die Spalte gewählt werden mit der höchsten Bewertung, die folgendermaßen ermittelt wird: Die Bewertung einer Spalte ergibt sich, in dem die maximale Anzahl zusammenhängenden Spielsteine (vertikal, horizontal, diagonal) mit der Anzahl der Auftreten solcher Reihen mit maximaler Anzahl zusammenhängender Spielsteine multipliziert wird.

Beispiel:

In der folgenden Spielsituation würde die gekennzeichnete Spalte 3 die Bewertung 2*3=6 für einen Spielstein von Spieler 2 ergeben, da zwei zusammenhängende Reihen von Spielsteinen (waagrecht und diagonal) mit der maximalen Anzahl 3 von zusammenhängenden Spielsteinen auf dem Spielbrett zu finden sind:



(5 Punkte)

h) Implementieren Sie eine Methode `void play()`, die den gesamten Spielablauf simuliert. Es soll nachgefragt werden, ob Spieler 1 ein menschlicher Spieler oder ein Computergegner ist und ob Spieler 2 ein menschlicher Spieler oder ein Computergegner ist. Danach soll solange abwechselnd Spieler 1 und Spieler 2 ihre Spielsteine einwerfen durch Aufruf der entsprechenden Methoden für den menschlichen Spieler bzw. Computergegner, bis das Spiel zu Ende ist. (5 Punkte)

Bemerkungen:

- Natürlich gibt es bessere Strategien für den Computergegner bis zu Vorausberechnungen für ein perfektes Spiel.
- Dieses ist eine Programmieraufgabe. Bitte geben Sie die Lösung *sowohl* auf **Papier** *als auch* den Java-Quelltext in elektronischer Form per **Email** an ihren Tutor ab.

Bemerkungen:

- Jede Seite soll oben rechts den Namen der Abgebenden und die Übungsgruppennummer (wichtig!) enthalten.
- Lösungen für die Übungsaufgaben sind (in der Regel) zu zweit abzugeben.
- Kommentieren Sie Ihre Lösungen! Besteht eine Lösung aus mehreren Zetteln, so sind diese zusammen zu heften. Bitte keine Hüllen, Mappen, o.ä..
- Bitte schicken Sie *Programmieraufgaben zusätzlich zur Abgabe auf Papier in elektronischer Form per Email* an ihren jeweiligen Tutor.
- Kommentieren Sie ihren Quelltext bei Programmieraufgaben. Dabei sollen keine Trivialitäten kommentiert werden, also bitte keine Kommentare wie

  ```
  x=5; // Wir weisen nun der Variablen x den Wert 5 zu
  ```

  sondern sinnvolle Kommentare, die Ideen des Quelltextabschnittes beschreiben oder auf Unteraufgaben (z. B. a), b), …) hinweisen.
- **Hinreichende Bedingung für die Zulassung zur Klausur:** 50% der erreichbaren Punkte bei jedem Übungszettel (bis auf zwei) und einmaliges Vorrechnen in der Übung
- **Zertifikatskriterium:** Das Bestehen der Klausur am Ende des Semesters

Abgabetermin: Freitag, 30.11.2007, nach der Vorlesung