Technische Universität Berlin **WS 2012/2013**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Frau Kiefer

Dr. Britta Peis
Martin Groß
Dr. Max Klimm
Madeleine Theile

Aylin Acikel, Katharina Bütow, Christian Döblin,
Alexander Hopp, Daniel Kuske, Olivia Röhrig, Robert Rudow,
Daniel Schmand, Hendrik Schrezenmaier, Judith Simon,
Sebastian Spies, Steffen Suerbier, Fabian Wegscheider, Jan Zur

Computerorientierte Mathematik I

Tic-Tac-Toe

7. Programmieraufgabe

Abnahme: spätestens am 20./21.12.2012 (je nach Gruppennummer).

Ungerade Gruppennummern geben spätestens am Donnerstag, Gerade Gruppennummern geben spätestens am Freitag ab.

Dies ist die erste Programmieraufgabe der zweiten Semesterhälfte.

Das Spiel Tic-Tac-Toe, wird von zwei Spielern auf einem Spielfeld der Größe 3×3 gespielt. Die Spieler markieren abwechselnd ein noch nicht markiertes Feld mit ihrem Symbol, üblicherweise einem „X“ oder einem „O“. Es gewinnt derjenige Spieler, der als erstes drei eigene Symbole in einer (horizontalen, vertikalen oder diagonalen) Reihe des Spielfeldes hat.

Für beliebige Parameter $n, m \in \{1, \ldots, 10\}$ und $k \in \{3, \ldots, \min\{n, m\}\}$, wollen wir das Spiel auf Spielfelder der Größe $n \times m$ verallgemeinern. Es soll derjenige Spieler gewinnen, der als erstes k aufeinander folgende eigene Symbole in einer (horizontalen, vertikalen oder diagonalen) Reihe hat, dabei soll $k \in \{3, \ldots, \min\{n, m\}\}$ gelten.

Schreibt eine Klasse `TicTacToe`, die ein verallgemeinertes Tic-Tac-Toe-Spiel modelliert.

(a) Legt die Klasse `TicTacToe` mit einem privatem Datenfeld `char[][] gameboard` an. Die Anzahl aufeinanderfolgender Symbole, die jeder Spieler zum Sieg benötgt, soll in der privaten Variablen `int wins` gespeichert werden.

(b) Stellt zwei Konstruktoren bereit. Der leere Konstruktor `TicTacToe()` soll ein normales 3×3 Spiel erstellen, in dem der Spieler gewinnt, der als erster 3 eigene Symbole in einer Reihe hat. Der Konstruktor `TicTacToe(int rows, int columns, int win)` erstellt ein verallgemeinertes Spiel auf einem Spielfeld mit `rows` Zeilen und `columns` Spalten, in dem der Spieler gewinnt, der als erstes `wins` aufeinander folgende eigene Symbole in einer Reihe hat. Zu Beginn sollen alle Felder mit einem Leerzeichen initailisiert werden. Werft eine `IllegalArgumentException`, falls die übergebenen Daten kein sinnvolles Spiel zulassen.

(c) Die Methoden `numberOfRows()` und `numberOfColumns()` soll die Anzahl der Zeilen beziehungsweise der Spalten des Spielfeldes zurück geben. Die Methode `getWinCondition()` soll zurück geben, wie viele konsekutive Symbole jeder Spieler zum Sieg benötigt. Die Methode `symbolAt(int row, int column)` gibt das Symbol in der Zeile `row` und der Spalte `column` zurück.

(d) Implemetiert die Methode `outputBoard()`, die den momentanen Spielstand ausgibt. Dabei sollen die Zeilen mit A, B, C,... und die Zeilen mit 1, 2, 3,... durchnummeriert sein.

(e) Implementiert die Methode

`public boolean setMove(int row, int column, char player)`.

Die Methode soll das Symbol `player` an die Stelle (`row`,`column`) des Spielfeldes setzen. Es soll eine `IllegalArgumentException` geworfen werden, wenn das Feld nicht existiert oder bereits gewählt wurde. Anschließend soll geprüft werden, ob der Spieler durch diesen Zug gewonnen hat.

Hinweis: Um die Siegbedinungen zu überprüfen, ist es sinnvoll, sich Zeile, Spalte und die Diagonalen, die das Feld (`row`, `column`) enthalten, als `String` ausgeben zu lassen. Anschließend kann man mit der Methode `substring` überprüfen, ob der String `wins` aufeinander folgende Symbole des Spielers enthält.

(f) Legt im gleichen Verzeichnis eine Datei `HumanPlayer.java` an, die die Klasse `HumanPlayer` enhält. Die Klasse soll eine private Referenz `TicTacToe game` und die zwei Symbole `char mySymbol` und `char otherSymbol` enthalten. In `mySymbol` soll das eigene Symbol, in `otherSymbol` das des Gegenspielers gespeichert werden.

Legt einen Konstruktor an, dem die drei Daten übergeben werden.

Implementiert anschließend die Methode `int[] makeMove()`. In dieser soll von der Kommandozeile ein Feld eingelesen werden. Die Eingabe soll so erfolgen, dass erst der Buchstabe der gewünschten Zeile und dann (ohne Leerzeichen) die Nummer der jeweiligen Spalte angegeben werde. Zum Beispiel sind in einem 3×3-Spiel A0, B1 und B2 gültige Eingaben. Die Eingabe soll so lange wiederholt werden, bis ein noch nicht besetztes Feld eingegeben wurde. Anschließend sollen der Index der gewählten Zeile und Spalte in einem `int[]` der Länge 2 zurück gegeben werden.

(e) Lasst nun in der `main`-Methode der Klasse `TicTacToe` das Spiel ablaufen. Fragt zunächst die Anzahl Zeilen, die Anzahl Spalten und die Siegbedingung vom Benutzer ab. Legt dann zwei `HumanPlayer` an, die abwechselnd ziehen. Nach jedem Spielzug soll das aktualisierte Spielfeld mit Hilfe von `outputBoard()` ausgegeben werden. Beendet das Spiel, wenn ein Spieler gewonnen hat oder wenn alle Felder belegt sind mit einer entsprechenden Meldung. Exceptions müssen in `main` nicht gefangen werden.

Viel Spaß und Erfolg!