c# Programmieren (c) HTL-Leonding

Gomoku

Lehrziele

- mehrdimensionale Arrays
- Datei Lesen/Schreiben
- Board

Aufgabenstellung

Gomoku ist ein Spiel für zwei Spieler, dass in verschiedenen Ländern unter verschiedenen Namen gespielt wird. In China z.B. **Wuziqi**, in Korea **Omok**, in Europa **Fünf in einer Reihe** und international **Gomoku**. Auf einen 15x15, 17x17 oder 19x19 dürfen die beiden Spieler abwechselnd Steine an eine beliebige, leere Spielfeldposition setzen. Gewinner ist derjenige, der als ersters fünf Steine in einer Reihe, Spalte oder Diagonale hat.

Programmablauf

Schreiben Sie ein Programm mit dem **Gomoku** gespielt werden kann.

- Beim Programmstart wird die Dimension des Spielfelds eingegeben.
  Da es sich um ein quadratisches Spielfeld handelt, muss nur eine Dimension eingegeben werden, z.B.: `17`.
  Erlaubte Größen sind 15x15, 17x17 oder 19x19.
- Anschließend wird das **Board** mit den eingegebenen Dimensionen erstellt.
- Zwei Benutzer können abwechselnd eine Position (Zeile/Spalte) eingeben, in die der nächste Stein platziert werden soll.
  Beispiel für die Eingabe `3,7`
- Das Spiel ist beendet, wenn ein Spieler gewonnen hat **oder** kein Stein mehr eingeworfen werden kann (das Spielfeld ist voll => unentschieden).
- Ein Spieler kann **aufgeben**.
  Mit der Eingabe eine Rufzeichens hat der andere Spieler gewonnen und das Programm wird beendet.
- Mit der Eingabe von `s` (für `save`) wird der Status des Spiels in eine CSV Datei gespeichert.
- Gibt ein Spieler `l` (für `load`) ein, wird das aktuelle Spiel beendet und das Spiel aus der CSV Datei geladen.
- Bei einer fehlerhaften Eingabe muss diese wiederholt werden: z.B. einer falsche Spielfeldgröße, bei einer ungültigen Zeile/Spalte, bei einem bereits besetztem Feld, ...

Programmdesign

Achten Sie bei der Umsetzung auf ein sauberes Design Ihres Programms.
In dem bereitgestellten Programm-Template sind Unittests vorhanden. Implementieren Sie die Methoden so, dass **alle** Unittests erfolgreich ausgeführt werden können. Änderungen an den *Unittests* sind nicht erlaubt.
Damit die Unittests ausgeführt werden können, müssen sie folgende (Hilfs-)Methoden umsetzen:

- `bool IsWinner(int[,] field, int row, int col)`
  Überprüft, ob sich durch die Belegung eines Feldes ein Sieger ergeben hat. Dabei werden nicht alle Felder überprüfen, sondern nur die an die neu gesetzte Position angrenzenden.
- `int GetStoneCount(int[,] field)`
  Zählt alle am Spielfeld platzierte Steine.
- `bool SetStone(int[,] field, int row, int col, int player)`
  Die Methode setzt einen Stein am Spielfeld. Der Rückgabewert legt fest, ob mit dem Setzen des Steines der Spieler gewonnen hat (bool).
- `void SaveGame(int[,] field, string fileName)`
  Das Spielfeld (gespeichert im `field`) wird in eine Datei geschrieben. Die CSV Datei hat folgende Spalten: `"No;Row;Col;Player"` - "No" ist eine fortlaufende Nummer. Hinweis: es werden nur **Steine** in die Datei geschrieben, die Position der leeren Felder jedoch nicht.
- `int[,] LoadGame(int boardSize, string fileName)`
  Ein zuvor gespeichertes Spiel wird wieder geladen. Das Format der Datei entspricht der CSV Datei der Methode `SaveGame`.

Testen Sie das Programm ausführlich.

Verwenden Sie im Programm (für das Feld **field**) folgende Codierung:
`-1` = frei, `0` = Spieler 1 (rotes X), `1` = Spieler 2 (grünes O)

Bildschirmausgabe

```
Gomoku
=========
Board size [15,17 or 19]: 15
"row,col" to set stone, e.g. 5,7
! to resign game (quit program)
s to save the game (and continue)
l resign the current game and load the store game - continue with the stored game.
Player 1: 5,5
Player 2: 4,4
Player 1: 5,5
Illegal input, please try again
Player 1: s
Game saved.
Player 1: 6,6
Player 2: l
Game loaded.
Player 1: 6,6
Player 2: !
```