Übungen zu Softwareentwicklung 1 **WS 2000/2001**
Übung 9

Name: _____ Tutor: _____

Matrikelnummer: _____ Punkte: _____

Gruppe: _____ **Abzugeben bis: 17. 1. 2001, 12:00**

Übungsleiter: _____ Bearbeitungsdauer: _____

Tic Tac Toe (7 + 4 + 5 + 8 Punkte)

Implementieren Sie das Spiel Tic Tac Toe. Das Spielfeld besteht aus 3 x 3 Feldern, in das zwei Spieler nacheinander ein X und ein O in ein freies Feld setzen können. Hat ein Spieler 3 gleiche in einer Reihe, so hat er gewonnen, kann keiner der beiden Spieler mehr ziehen ist das Spiel unentschieden.

a) Implementieren Sie das Spielfeld mit folgender Schnittstelle. Die Klasse soll sich im Package **swe1.tictactoe** befinden. Die Klasse und alle Methoden sollen 'package-private' (=> kein Modifier) sein. Sie dürfen Attribute oder zusätzliche Methoden verwenden, sofern diese für die Klasse 'private' sind:

```
class Field extends Basic {
   // Use these constants whereever possible!!
   final static char PLAYER_X = 'X', PLAYER_O = 'O', NONE = ' ';
   final static char GAME_DRAW = '-', GAME_WAIT_MOVE = ' ';
   final static String ERROR_XXX = ..., ERROR_YYY = ..., ...;

   Field ();
   // Deletes the field, next player is 'nextPlayer'
   void reset (char nextPlayer);
   // Player moves (returns an error message (use constants, see
   // above) if move is not allowed, null otherwise)
   String move (char player, int row, int column);
   // Checks if move is valid (returns the same as method 'move')
   String checkMovePossible (char player, int row, int column);
   // Prints the field (e.g. as shown below)
   void printField ();
   // returns PLAYER_X, PLAYER_O, GAME_DRAW, or GAME_WAIT_MOVE
   char theWinnerIs ();
   // returns PLAYER_X, PLAYER_O, or NONE
   char getElement (int row, int column);
}
```

b) Implementieren Sie die Benutzerschnittstelle (Klasse TicTacToe), sodass Sie folgende (oder eine bessere) Ausgabe erhalten. Es müssen mehrere Spiele hintereinander möglich sein ("New game (y/n)?"). Diese Klasse enthält die Methode 'main' (d.h. diese Klasse entspricht dem Testprogramm):

Beispiel (Ausschnitt):

```
 X |   |
---+---+---
   | O |
---+---+---
 O | X | X
```

```
Next player is O. O's move (row column): 1 3

 X |   | O
---+---+---
   | O |
---+---+---
 O | X | X

Player O wins! Congratulations!
New game (y/n)? y
```

c) Protokollieren Sie alle Züge (als Klasse?) mit (nur Züge, nicht das Spielfeld mitspeichern). Es genügt ein fixes Array der Länge 9 - mehr Züge können nicht gemacht werden.
Implementieren Sie die Methode

```
void undoMove (int nOfMoves);
```

in der Klasse Field, mit der Züge zurückgenommen werden können. Achten Sie darauf, dass das Spielfeld immer in einem gültigen Zustand bleibt (Nimmt man z.B. einen Zug zurück, ist der andere Spieler als nächstes dran).
Erweitern Sie die Klasse TicTacToe, sodass man (möglichst bequem) Züge zurücknehmen kann.

d) Implementieren Sie eine Klasse Player, die von nun an die Spielzüge liefern soll:

```
class Player extends Basic {
   // Types of players
   final static int USER = 0, RANDOM = 1, COMPUTER_SIMPLE = 2;

   Player (int playerType, char playerName);
   // Returns true, if the game should be stopped (by User request).
   // To make a move, call method 'move'
   boolean doMove (Field field);
   char getPlayerName (); // Field.PLAYER_X or Field.PLAYER_O
}
```

In der Klasse Field sind noch folgende Methoden zu implementieren

```
void setPlayers (Player playerX, Player playerO);
Player getNextPlayer ();
void play ();
```

zu implementieren. Sobald das Spiel gestartet wurde, sollen die beiden Spieler abwechselnd einen Zug ziehen, bis das Spiel beendet ist (gesteuert in der Methode play). Je nach gesetztem Spielertyp wird (in doMove) automatisch gezogen, oder von der Kommandozeile eingelesen.

- Die Klasse TicTacToe soll jetzt nur mehr die Spieler erzeugen und setzen, sowie das Spiel starten, das Spielergebnis ausgeben und nachfragen, ob noch ein Spiel gespielt werden soll. Undo muss weiterhin funktionieren.
- Der User-Spieler (doMove - USER) kann das Spiel stoppen, um einen Zug zurueckzunehmen
- Für den Zufalls-Spieler (doMove - RANDOM) können Sie den Zufallsgenerator (Random) aus dem Package java.util verwenden.
- Für den einfachen Computer-Spieler (doMove - COMPUTER_SIMPLE) genügt es z.B. festzustellen, ob der nächste Zug einen Sieg des Gegners verhindern kann. Ansonsten wie Zufalls-Spieler.
- Es ist nur jeweils die letzte Version der Klassen abzugeben (d.h. keine Zwischenversionen).
- Schnittstellen sind unbedingt einzuhalten! Das Verhalten (Eingabe, Ausgabe, ...) des Spiels kann selbst bestimmt werden (Versuchen Sie sinnvolle Entscheidungen zu treffen!). Spielen Sie das Spiel eine Weile selbst, gefällt Ihnen das Spiel? Ist es einfach zu bedienen? Könnte das Spiel jemand anders schnell begreifen?
- Jede Methode muss verwendet werden. Notfalls ausserhalb des Spielablaufs testen!