Prof. Dr. Raymond Bimazubute
Duale Hochschule Baden-Württemberg - Stuttgart

Werkzeuge und Methode der Softwareentwicklung

Assignment zu fortgeschrittenen Techniken der Softwareentwicklung

DHBW Stuttgart

Entwicklung des Spiels Tic-Tac-Toe

In diesem Assignment soll das Spiel Tic-Tac-Toe realisiert werden. Das Spiel wird auf einem quadratischen, 3×3 großen Gitter von zwei Spielern gespielt. Die beiden Spieler setzen abwechselnd ihr Zeichen (z.B. „X“ und „O“) in ein freies Feld. Der Spieler, der als Erster drei Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt. Weitere Hintergrundinformationen und Regeln zu diesem Spiel finden Sie auf Wikipedia (https://de.wikipedia.org/wiki/Tic-Tac-Toe).

Ihre Implementierung soll auf der Basis der bereitgestellten Klasse „TicTacToe“ erfolgen.

```
public class TicTacToe {

	private int ROWS = 3;
	private int COLS = 3;
	public Player[][] board;

	public enum Player {
		Unknown, X, O
	}
	private Player currentPlayer;
	private Player winner;
	public TicTacToe() {}
	public void initialiazeBoard() {}
	public void play(int row, int col) {}
	public boolean isBoardEmpty() {
	public boolean isGameWon() {}
	public boolean isDraw() {}
	public boolean isGameOver() {}
	public Player getCurrentPlayer() {}
	public Player getWinner() {}


}
```

Folgende Methoden sind zu implementieren:

`public TicTacToe() {}`	Konstruktor der Klasse. • Im Konstruktor wird ein zweidimensionales Array initialisiert. Jedes Feld des Arrays erhält als Inhalt den Enum-Wert **Player.Unknown**. • Im Konstructor wird festgelegt, dass der Spieler „X“ anfängt. • Im Konstruktor wird festgelegt, dass zu Beginn eines Spiels, kein Gewinner feststeht. Dies wird durch Initialisierung des **Attributs winner mit dem Wert null** festgelegt.
`int ROWS = 3;` `int COLS = 3;`	Das Gitter hat 3 Zeilen und 3 Spalten.
`public enum Player` `{Unknown, X, O}`	Enumeration, um Spieler abzubilden: • Unknown: Spieler unbekannt/steht nicht fest • X: Spieler mit dem Zeichen X • O: Spieler mit dem Zeichen O
`private Player currentPlayer;`	• Der Spieler, der aktuell am Zug ist. • Der Spieler X startet immer das Spiel.
`private Player winner;`	Das Attribut gibt den Spieler zurück, der das Spiel gewonnen hat. Mögliche Werte: • X: Spieler X hat gewonnen • O: Spieler O hat gewonnenen • Unknown: Das Spiel ist unentschieden • null: Gewinner steht noch nicht fest, da das Spiel noch läuft. Anmerkungen • Das Attribut hat den Wert null, solange das Spiel nicht beendet ist. • Die Initialisierung mit dem Wert null wird im Konstruktor vorgenommen
`public void` `play(int row, int col) {}`	Der Spieler, der an der Reihe ist, gibt die Zelle an, die er belegen möchte. Spielregeln in der Methode play: • Wenn die Methode aufgerufen wird, dann wird eine Markierung des jeweiligen Spielers in der entsprechenden Zelle des Gitters platziert Beispiel: Spieler X spielt (0,0), dann wird das Zeichen X in der Zelle (0,0) platziert. • Falls die Koordinaten **row oder col** außerhalb des Gitters liegen, dann soll eine Runtime-Exception ausgelöst werden

	• Falls die durch die Koordinaten **row** und **col** angegebene Zelle besetzt ist, dann soll eine RuntimeException ausgelöst werden • Wird die Methode play aufgerufen nachdem ein Spiel beendet ist, dann soll eine Runtime-Exception ausgelöst werden (Die Methode **isGameOver** gibt an, ob ein Spiel beendet ist) • Falls das Spiel durch den aktuellen Zug **nicht gewonnen wurde**, dann ist der andere Spieler dran. Falls durch den Zug eine Zeile, eine Spalte oder eine Diagonale des Gitters 3 Zeichen des gleichen Spielers enthält, **hat der aktuelle Spieler gewonnen**. Dann muss der Spielstand aktualisiert werden (X hat gewonnen, O hat gewonnen, unentschieden).
`public void initialiazeBoard() {}`	Die Methode initialisiert das zweidimensionale Array. Jedes Feld des Arrays erhält als Inhalt den Enum-Wert **Player.Unknown**.
`public boolean isBoardEmpty() {}`	Die Methode prüft, ob jedes Feld des zweidimensionalen Arrays den Wert **Player.Unknown** hat**.**
`public boolean isGameWon() {}`	Die Methode prüft, ob das Spiel gewonnen wurde. • Rückgabewert true bedeutet, dass das Spiel gewonnen wurde • Rückgabewert false bedeutet, dass das Spiel nicht gewonnen wurde • Solange ein Spiel nicht gewonnen wurde, liefert die Methode den Wert false.
`public boolean isDraw() {}`	Die Methode prüft, ob das Spiel unentschieden ist. Dies ist der Fall, falls **alle Zellen des Gitters besetzt sind**, ohne dass 3 Zeichen des gleichen Spielers in einer Zeile, Spalte oder Diagonale sind. Falls das Spiel unentschieden ist, dann muss die Methode getWinner den Wert **Unknown** zurück liefern.

`public boolean isGameOver() {}`	• Die Methode überprüft, ob das Spiel beendet ist. Ein Spiel ist beendet, wenn
	  ○ das Spiel gewonnen wurde oder
	  ○ unentschieden ist.

`public Player getCurrentPlayer() {}`	Methode gibt an, welcher Spieler an der Reihe ist. Die Spieler müssen sich abwechseln. Beispiel: 1) X spielt 2) Nach X ist nun O an der Reihe. Die Methode getCurrentPlayer liefert **Player.O** zurück
`public Player getWinner() {}`	Die Methode liefert den aktuellen Gewinner des Spiels. Dies ist eine Getter-Methode für das Attribut **winner**.