IFM 2.2 Programmiermethoden

Blatt 02: Strategy-Pattern, Git Branches

Praktikum: KW16 (18./22.04.2016)

Birgit C. George, Carsten Gips, FH Bielefeld

FH Bielefeld
University of
Applied Sciences

Laden Sie Ihre Lösung bis spätestens zum **18.04.2016 (08:00 Uhr)** als Zip-Archiv in die ILIAS-Aufgabe hoch. Geben Sie dabei im ILIAS die jeweiligen Team-Mitglieder an!

Zur Erlangung der Punkte müssen die Aufgabenteile im jeweiligen Praktikum vorgeführt und erklärt werden. Jedes Team-Mitglied muss Auskunft über jeden Teil der Bearbeitung geben können.

1 Strategy-Pattern: Eigenständige Spieler (6 Punkte)

In dieser Aufgabe erweitern Sie Ihre Implementierung des Spiels „Tic Tac Toe“ vom letzten Blatt.

Laden Sie sich die Vorgaben herunter (Zip-Archiv im ILIAS) und binden Sie diese in Ihr Programm ein.[1]

Bauen Sie Ihr Programm so um, daß es eine eigene Klasse für die Spieler gibt, die das vorgegebene Interface `game.Player` implementiert. Ihr Spiel muss das vorgegebene Interface `game.Game` implementieren. Nutzen Sie für die Darstellung der Spielzüge das Interface `game.Move` aus der Vorgabe.

Nutzen Sie das Strategy-Pattern, um den Spielerinstanzen zur Laufzeit eine konkrete Spielstrategie mit dem Interface `strategy.GameStrategy` mitzugeben, nach denen die Spieler ihre Züge *berechnen*. Implementieren Sie mindestens drei unterschiedliche konkrete Strategien.[2] Nutzen Sie als weitere Strategie die Vorgabe zum Minimax-Algorithmus (`strategy.MinMaxStrategy`). Damit steht eine stets perfekt spielende Spielstrategie zur Verfügung.[3]

Stellen Sie die jetzt vorhandenen Klassen und ihre Beziehungen in einem **manuell** erstellten UML-Klassendiagramm dar.

Hinweis: Ein Spieler soll seinen nächsten Zug nur **berechnen** und darf nicht direkt den Spielstand (Spielbrett) modifizieren! Dies geschieht über die Methode `game.doMove()` aus dem Spiel (Interface `game.Game`). Die Berechnung im Spieler wird direkt an die zur Laufzeit übergebene Strategie delegiert.

Ziel: Nutzung des Strategie-Entwurfsmusters

2 Git Branches und Mergen – Kommandozeile (4 Punkte)

Üben Sie den Umgang mit Git auf der Kommandozeile:

a) Legen Sie in Ihrem Projekt einen Branch an. Ändern Sie einige Dateien und committen Sie die Änderungen. Checken Sie den Master-Branch aus und mergen Sie die Änderungen. Was beobachten Sie?

b) Legen Sie einen weiteren Branch an. Ändern Sie einige Dateien und committen Sie die Änderungen. Checken Sie den Master-Branch aus und ändern Sie dort ebenfalls:

- Ändern Sie eine Datei an einer Stelle, die nicht bereits im Branch modifiziert wurde.
- Ändern Sie eine Datei an einer Stelle, die bereits im Branch manipuliert wurde.

Committen Sie die Änderungen.

Mergen Sie den Branch jetzt in den Master-Branch. Was beobachten Sie? Wie lösen Sie Konflikte auf?

Hinweis: Dieses Vorgehen ist in der Abgabe live auf der Kommandozeile (Konsole/Git Bash) vorzuführen!

Ziel: Sicherer Umgang mit den grundlegenden Arbeitsabläufen in Git

[1] Gehen Sie im Package Explorer auf das .jar-File und wählen Sie im Kontextmenü „Build Path > Add to Build Path“.
[2] Eine mögliche Strategie könnte sein, den Nutzer via Tastatureingabe nach dem nächsten Zug zu fragen :-)
[3] Auf die Funktionsweise des Minimax-Algorithmus wird im Wahlmodul „Künstliche Intelligenz“ genauer eingegangen :-)