Rückblick zum Programmieren Projekt “LOGO Interpreter”

Die Gruppenarbeit:

Diese lief beinahe Reibungslos ab, da wir bereits in früheren Projekten schon mehrmals zusammengearbeitet haben. Daher kannten wir auch bereits im vorhinein die Vorlieben des Anderen, welches die Vergabe der Teilaufgaben sehr leicht machte. Auch bei in der Planung “vergessene” Klassen, wie zum Beispiel einer Klasse, welche Dateien einliest oder speichert, traten aufgrund der eindeutigen Arbeitsteilung keine Probleme auf.

Kommunikation und Schnittstellen:

Für unser Projekt haben wir eine Git Repository genutzt, welche uns, mit dem dazugehörigen Eclipse Plugin, erlaubte immer mit den aktuellsten Versionen des Anderen zu arbeiten. Als Schnittstelle zwischen unseren Teilgebieten haben wir eine Controller Klasse genutzt, in welcher jeder seine benötigten bzw. verfügbaren Ressourcen angeben konnte. Lediglich in dieser Klasse kam es zu kleineren Unstimmigkeiten, beispielsweise aufgrund der identischen Bezeichnung eines Buttons und eines Befehls (siehe “Clear”). Welche sich aber sehr schnell beheben lassen konnten. In anderen Fällen haben sich unsere Aufgabenteile nicht überschnitten. Hier wurden sämtliche Aktualisierungen Autonom von Git übernommen.

Das Programmieren:

Im Allgemeinen gab es bei dieser Programmieraufgabe keine größeren Schwierigkeiten. Sämtliche in der Planung festgelegten Aufgaben und Meilensteine konnten vor den jeweiligen Deadlines beendet werden. Unsicherheiten bereiteten uns im Allgemeinen nur, dass wir uns nicht immer im Klaren waren, wie die Javakonventionen in bestimmten Bereichen (beispielsweise die GUI Programmierung) aussahen. In diesen Fällen haben wir uns dazu entschieden den Quellcode so leserlich und verständlich wie möglich zu gestalten.

Design & Erweiterbarkeit

Als Design wurde das Model-View-Controller Prinzip verwendet, um die Erweiterbarkeit so gut wie möglich zu gewährleisten.
Controller: Hierzu gehören die Klassen Parser, Interpreter, FileHandler und Controller. Parser und Interpreter, da diese das Kernstück des Logo Interpreters bilden, der Controller da er für die Aufgabendelegation verantwortlich ist und somit ebenfalls unbedingt benötigt wird. Der FileHandler könnte sowohl im View, wie auch im Controller liegen, wir haben uns für den Controller entschieden, da das Aufgabengebiet, des FileHandler's, nicht unbedingt etwas mit der View zu tun hat.
Model: Ist in diesem Fall eindeutig unsere Turtle Klasse.
View: dazu gehören die Klassen GUI, GuiListener, GraphPane und die HelpPage, da all diese für die Benutzeroberfläche verantwortlich sind.
Zwischen Model, View und Controller haben wir auf eine strikte Trennung geachtet. Sämtliche Änderungen in einer dieser Gebiete kann/muss in der Controller-Klasse angepasst werden. Weitere Veränderungen werden nicht benötigt.

Erweiterungsmöglichkeiten

Die Benutzerfreundlichkeit könnte in folgenden Punkten verbessert werden:

- Einführung von Pop-Up Fenstern, beispielsweise, wenn beim Speichern eine Datei überschrieben werden sollte.
- Eine Zeilenangabe im Editor Fenster.

Fazit:

Dieses Projekt lief für unser empfinden reibungslos ab. Im Team sind während des gesamten Ablaufes weder Streitigkeiten noch größere Unklarheiten aufgekommen. Zeitlich wäre es durchaus möglich gewesen ein größeres, umfangreiches Projekt durchzuführen. Allerdings war es unserem Empfinden nach nicht optimal, dass das Projekt erst während der Prüfungsperiode begann. Dadurch war es einem nur in selten Fällen möglich, für längere Zeit durchgehen an dem Projekt zu arbeiten, was den Workflow meines Erachtens deutlich verminderte. Dennoch war es durchaus möglich zusätzliche, eigene Ideen und Zusatzfunktionen zu implementieren.