Software Engineering 1
Abgabedokument
Teilaufgabe 2
(Reflexion, Plan vs. Wirklichkeit)

Nachname, Vorname:	Srdanovic, Nemanja
Matrikelnummer:	01576891
E-Mail-Adresse:	a01576891@unet.univie.ac.at
Datum:	11.12.2020

Unteraufgabe 1: Reflexion, Plan vs. Wirklichkeit

Aspekt - Kommunikationsreinfolge

- **Änderung:** MainClient kommuniziert direkt mit dem GameController

- **Begründung:** Durch das Feedback und auch das MVC Prinzip ist es klar geworden, dass es nicht üblich ist das Datenklassen (Game) eine Logiklasse (GameController) instanziiert.

Aspekt - Kapselung und Information Hiding

- **Änderung:** Wo immer es möglich war, sind Methoden von public auf private geändert und nur Schnittstellen sind public geblieben

- **Begründung:** Um diesem Designprinzip gerecht zu werden, sind die meisten Methoden, welche im Plan public waren, auf private umgeändert. Die Logik der Methodenaufrufe ist dadurch in einzelnen Methoden (Schnittstellen) geblieben. Dadurch sind die Methoden nach außen nicht mehr sichtbar bzw. der interne Aufbau der Controller ist versteckt. Es ist natürlich die Möglichkeit geblieben, bei Bedarf die Methoden jederzeit sichtbar zu machen, ohne dass diese Änderung negative Auswirkungen hat, da die nur innerhalb der Klassen eingesetzt werden konnten.

    - *GameController*: startGame - einzige public Methode, enthält Logik (welche private Methoden in welcher reinfolge aufgerufen werden).
    - *Cli:* PropertyChangeListener - kann die verschiedenen print Methoden aufrufen auch wenn die Methoden private sind, da er Teil der Klasse ist.
    - *MapController:* generateMap - einzige public Methode, welche alle anderen internen private Methoden aufruft.
    - *MapRules:* validateHalfMap - einzige public Methode, welche alle anderen private Validation Methoden aufruft.

Aspekt - Dekomposition und Modularisierung

- **Änderung:** Komplexe Methoden sind im Gegensatz zum Plan in kleinere unabhängige Teilprobleme zerlegt. Meistens werden dann alle diese Methoden in der Schnittstelle zur Lösung eines komplexen Problems (Route Kalkulation) verwendet.

- **Begründung:** Diese Methoden mussten in der Implementierung in mehrere einzelnen Methoden aufgeteilt werden, die alle wiederum zur Bewältigung einer Aufgabe eingesetzt werden. Dieser Prozess ist ganz natürlich in Laufe der Implementierung passiert, da sonst einzelne Methoden zu unübersichtlich und komplex geworden wären. Beispiel dafür ist die calculateRoute Methode, welche im Plan einzeln in der RuteCalculation Klasse zu finden ist.

Aspekt – Kohäsion

- **Änderung:** Zusammengehörige Methoden, die sich im Laufe der Implementierung definiert habe, sind in einzelne Klassen aufgeteilt wurden.

- **Begründung:** Diese Änderung ist der Übersicht halber passiert und man wollte damit vermeiden das einzelne Klassen zu komplex werden. So ist der Converter aus dem Plan in weitere Klassen aufgeteilt worden (EnumConverter und MapConverter) und Funktionalitäten, die Map-Objekte bearbeiten oder sie extrahieren, wurden in gleiche Komponenten (Klassen) gepackt.

Aspekt - Überflüssige Komponenten

- **Änderung:** Einige im Plan definierte Klassen sind im Laufe der Implementierung zusammengeschmolzen oder wurden ganz entfernt.

- **Begründung:** Es handelt sich dabei um Klassen, die ähnliche Daten gespeichert haben (HalfMap und CompleteMap) und wo es daher logischer war, eine Klasse mit ein paar mehr Funktionalitäten zu erstellen, als Daten unnötig zu verteilen. Dazu sind auch Klassen, welche im Planning noch als sinnvoll empfunden wurden (MoveRules und Move) wegen der geringen Anzahl an Daten und Funktionalitäten in andere Klasen eingebunden, um nicht gleiche Funktionalitäten auf zu viel Klassen zu verteilen und die Übersicht zu bewahren.