Reflexion, Plan vs. Wirklichkeit

Mit meinem Klassendiagramm habe ich versucht, SOLID Prinzipien zu folgen und könnte eine gute Basis für das Projekt legen. Jedoch später bei der Implementierung habe ich bemerkt, dass ich nicht wirklich den SOLID Prinzipien folge und das hat meine Implementierung mit jeder neuen Klasse oder Methode erschwert.

Für mich war es eine Herausforderung sich zu abstrahieren und Klassen mit Single Responsibility zu bauen. Ich habe während der Implementierung viel Gedanken gemacht, wie und was man ändern könnte. Ich wünschte mir, ich hätte das gleich beim Klassendiagramm besser gemacht.

Mit meinem Klassendiagramm waren Klassen sehr stark miteinander gekoppelt und später bei der Vorlesung habe ich erfahren, dass ich unbewusst „Stair"-Ansatz angewendet habe. Jedoch beim Erfahren von "Fork"-Ansatz habe ich dann meinen Code geändert und es war meine beste Entscheidung. Der Vorteil davon war, dass meine Client-Klasse die Steuerung übernommen hat, jedoch war es sehr schwierig zu verstehen, was gemacht wird. Mit „Stair"-Ansatz hat meine Implementierung wie eine große Verkettung ausgeschaut und es war sehr schwierig zu debuggen und nachzuvollziehen, was in dem Code passiert.

Auch hatte ich Kommentare in meinem Code, jedoch später habe ich erfahren, dass der Code selbsterklärend sein soll und wenn es zu schwer oder unverständlich ist, dann sollte man eine Methode daraus machen, mit passenden Namen. Kommentare habe nur dort gelassen, wo meine Entscheidung erklärt wird.

Außerdem habe ich die Design Patterns angeschaut und bin auf Facade Pattern gestoßen und es hat super in mein Projekt gepasst. Das Problem war, dass mein Client Controller sehr Umfangreich war und man viele Methoden bräuchte. Mit Facade Pattern hat mein Client-Controller nur noch ein paar Methoden.

Interfaces haben mir sehr bei der Implementierung von Move Strategy geholfen. Ich habe ein Interface für die Suchung von Feld erstellt und ich könnte dynamisch meine Strategie wechseln. Zuerst habe ich versucht, alle 4 Ecken der Karte zu besuchen, jedoch war es nicht sehr effizient. Dann habe ich versucht, alle Berge zu besuchen und dann alle Grass. Es war unkompliziert, etwas dazuzugeben, ohne den tatsächlichen Code zu ändern.

Jedoch waren die größte Herausforderung die Unit-Tests. Ich könnte an manchen Klassen kein Mockito verwenden und es gab viele private Methoden. Es hat mir Hinweis gegeben, dass manche Klassen nicht ganz Single Responsibilty folgten und ich habe es behoben, indem ich neue Klassen erstellt habe und eine Dependency-Injektion angewendet.

Zusammengefasst hat die SOLID Prinzipe sehr stark mein Projekt beeinflusst und erleichtert. Obwohl ich davon gelesen habe, habe ich es bei der Implementierung wirklich „hoffentlich" verstanden und angewendet.