Prosa
Der am stärksten veränderte Teil meiner Implementierung war dieses Mal der MoveGenerator-Teil. Um die Effizienz der KI beim Auffinden von Schätzen und gegnerischen Burgen zu verbessern, habe ich beschlossen, Methoden hinzuzufügen, die zuvor in Klassendiagrammen nicht berücksichtigt wurden, einschließlich Heuristik, BFS und anderen Methoden . Gleichzeitig habe ich zur Einhaltung des Single-Responsibility-Prinzips auch die MoveGenerator-Klasse unterteilt, einschließlich der Erstellung einer neuen PathFinder-Klasse und der Unterteilung verschiedener großer Methoden in kleine Methoden mit unabhängigen Aufgaben.

Um die Unterscheidung der Datenobjekte von Client und Server zu erleichtern, habe ich außerdem beschlossen, eine neue NetworkConverter-Klasse in die ClientNetwork-Klasse zu unterteilen. Die gesamte Kommunikation mit dem Server wird auf die ClientNetwork-Klasse beschränkt. Die vom Server erhaltenen Datenobjekte werden in Client-Datenobjekte in der NetworkConverter-Klasse konvertiert und dann auf andere Klassen auf dem Client angewendet, was dem Single-Responsibility-Prinzip sehr gut folgt.

Um die Kopplung zu reduzieren, habe ich die CLI geändert und einen PropertyChangedListener hinzugefügt, um auf Daten zu warten, die rechtzeitig aktualisiert werden müssen, und diese anzuzeigen. Gleichzeitig habe ich auch andere anzeigebezogene Methoden in Klassdiagram (z. B. die displayMap()-Methode in HalfMapGenerator) zur Integration in die CLI verschoben, um dem Single-Responsibility-Prinzip zu entsprechen.

Y-Statements
1. Im Kontext der Effizienzverbesserung der KI haben wir uns angesichts der Notwendigkeit, das Single-Responsibility-Prinzip einzuhalten, dafür entschieden, den MoveGenerator zu überarbeiten und neue Methoden hinzuzufügen, die zuvor nicht berücksichtigt wurden. Dies beinhaltete die Integration von Heuristik, BFS und anderen Methoden. Dadurch wurde die Effizienz der KI verbessert und dem Single-Responsibility-Prinzip gefolgt, jedoch wurden zusätzliche Klassen und Methoden hinzugefügt, was zu einer erhöhten Komplexität des Systems führte.

2. Im Kontext der Verbesserung der Datenobjekttrennung zwischen Client und Server haben wir uns entschieden, eine neue NetworkConverter-Klasse in die ClientNetwork-Klasse zu integrieren. Dadurch wird die Kommunikation mit dem Server auf die ClientNetwork-Klasse beschränkt, und die erhaltenen Server-Datenobjekte werden in Client-Datenobjekte in der NetworkConverter-Klasse konvertiert. Diese Entscheidung erleichtert die Unterscheidung zwischen Client- und Server-Datenobjekten und folgt dem Single-Responsibility-Prinzip. Allerdings führt dies zu einer erhöhten Anzahl von Klassen und einer zusätzlichen Komplexität des Systems.

3. Im Kontext der Reduzierung der Kopplung und der Einhaltung des Single-Responsibility-Prinzips haben wir entschieden, die CLI zu ändern und einen PropertyChangedListener hinzuzufügen, um auf rechtzeitig aktualisierte Daten zu warten und sie anzuzeigen. Außerdem haben wir anzeigebezogene Methoden in Klassendiagramme verschoben, um sie in die CLI zu integrieren. Diese Entscheidung führte zu einer Verringerung der Kopplung und einer besseren Einhaltung des Single-Responsibility-Prinzips. Allerdings führte sie auch zu einer erhöhten Komplexität der CLI und des Systems.