Prosa
Eine der größten Änderungen ist die Strukturierung der Klassen in separaten Paketen. In meinem ursprünglichen Klassendiagramm hatte ich alle Klassen wie Game, Player und Map in einem einzigen Game-Paket organisiert. In der Implementierung habe ich jedoch entschieden, jede dieser Klassen in eigene Pakete auszulagern, z.B. in das map-Paket und das player-Paket, weil ich die Verantwortlichkeiten von jeder Klasse klarer verteilen wollte. Gleichzeitig wird dadurch die Wartbarkeit des Codes verbessert.

Ein weiterer wesentlicher Aspekt war die Einführung der NetworkConverter-Klasse. In meinem ursprünglichen Klassendiagramm hatte ich nur eine ServerNetwork-Klasse, die die Netzwerk-Konvertierungen ebenfalls innerhalb der Klasse behandelte. Um das Single Responsibility Principle (SRP) besser umzusetzen, habe ich diese Konvertierungslogik in eine separate NetworkConverter-Klasse ausgelagert. Dadurch konnte die Komplexität der ServerNetwork-Klasse reduziert werden, was zu einer verbesserten Testbarkeit und Wartbarkeit des Codes führte.

Zusätzlich habe ich spezifische Controller-Klassen für jedes Paket eingeführt, um die Kommunikation und Geschäftslogik zu verwalten. Diese existierten nicht in meinem ursprünglichen Klassendiagramm. Diese Änderung trennte die Geschäftslogik von der Präsentationsschicht, was die Wiederverwendbarkeit und Testbarkeit der Code-Komponenten deutlich verbesserte.

Y-Statements
1. Im Kontext der Paketstruktur haben wir uns angesichts der Notwendigkeit einer besseren Modularität und Klarheit für die Trennung der Klassen in spezifische Pakete entschieden und die ursprünglich geplante einheitliche Struktur vernachlässigt, um eine klarere Verantwortlichkeitsverteilung und eine einfachere Wartbarkeit zu erreichen, unter Inkaufnahme von einer höheren Komplexität bei der Verwaltung der Pakete, weil dies die langfristige Wartbarkeit und Erweiterbarkeit des Projekts verbessert.

2. Im Kontext der Netzwerk-Konvertierung haben wir uns angesichts der Komplexität und der Wichtigkeit der Trennung der Verantwortlichkeiten für die Einführung einer NetworkConverter-Klasse entschieden und die Integration dieser Logik in die ServerNetwork-Klasse vernachlässigt, um das Single Responsibility Principle (SRP) zu wahren und die Testbarkeit zu verbessern, unter Inkaufnahme von einem zusätzlichen Implementierungs- und Integrationsaufwand, weil dies die Modularität und Wartbarkeit des Codes erhöht.

3. Im Kontext der Controller-Funktionalität haben wir uns angesichts der Notwendigkeit einer klaren Trennung der Geschäftslogik für die Einführung spezifischer Controller-Klassen entschieden und eine zentrale Verwaltung in einer einzigen Klasse vernachlässigt, um die Verantwortlichkeiten zu verteilen und die Testbarkeit zu erhöhen, unter Inkaufnahme von erhöhter Komplexität und dem Bedarf an zusätzlicher Koordination zwischen den Controllern, weil dies die Wiederverwendbarkeit und Erweiterbarkeit der einzelnen Komponenten verbessert.