Readme

*** Ich habe eine weitere gradle.properties file erstellt, wo ich den Pfad zu JDK gestellt habe. Ohne das funktioniert die Gradle tasks bei mir nicht, und ich bitte um Entschuldigung fur die Unannehmlichkeit. (Ich weiß, Sie haben schon das beim Client erwähnt).**

Ich habe alle Endpoints implementiert:

Game Creation

Es wird ein UniqueGameIdentifier erstellt und zum Client geschickt.

Player Registration

Client kann sich fur ein bestimmtes Spiel registrieren und bekommt danach ein UniquePlayerIdentifier zurück.

Es wird geprüft ob die erwünschte gameId existiert UND ob es nicht schon 2 Players fur das Spiel registriert sind.

HalfMap Receiver

Der Server bekommt den Half Maps von den Clients, prüft ob diese korrekt sind und speichert diese.

Es wird geprüft ob die erwünschte gameId existiert UND ob der Player der sendet existiert UND ob beide Players registriert sind UND ob der Player nur eine Map gesendet hat UND ob der Player der sendet dran ist.

Es werden alle Half Map rules geprüft.

Wenn alles passt, die HalfMap wird zu dem entsprechenden Player gespeichert und die andere Player kommt dran.

GameStateId wird aktualisiert.

Send GameState

Die Clients fragen nach ihren Game Status und bekommen den aktuellen Spielzustand zurück.

Es wird geprüft ob die erwünschte gameId existiert UND ob der Player der fragt existiert.

Wenn beide HalfMaps schon vorhanden sind, werden diese in einem FullMap zusammengestellt und zusammen mit dem Spielzustand zurückgeschickt. Wenn nicht, nur eine HalfMap (+ Zustand) wird zurückgeschickt.

Send Move

Die Clients senden Moves, der Server prüft ob diese korrekt sind und ändert den Spielzustand dementsprechend.

Es wird geprüft ob die erwünschte gameId existiert UND ob der Player der sendet existiert UND ob der Player der sendet dran ist.

Es werden alle Move rules geprüft. Wenn alles passt, die Move wird verarbeitet und die andere Player kommt dran. GameStateId wird aktualisiert.

Implementierung

Ich habe insgesamt **14 Business Rules** implementiert (Package server.validation). Um das **Open-Close Principle** zu respektieren, habe ich eine **Interface IRuleValidation** erstellt, die die generelle gebrauchte Methoden hat. Alle Business Rules Klassen implementieren dieses Interface indem sie nur die benutzte Methode implementieren. In den ServerEndpoints Klasse habe ich eine Liste von diesen Rules erstellt, die dazu hilft, immer nur die geeignete Rules zu überprüfen.

Fur das Validieren der Inseln der Karte habe ich das Floodfill Algorithmus implementiert. Die Idee wie diese implementiert werden muss habe ich von https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint/ genommen. (Ich habe diese von meinem Client genommen).

Das Validieren habe ich schon in den ServerEndpoints Klasse gemacht, sodass es schnellstmöglich gemacht wird und die fehlerhafte Objekte nicht mehr verarbeitet werden.

Wenn die kommende Objekte korrekt sind, dann speichere ich diese.

Bonuspunkte:

Ich habe auch den **Move Endpoint** und die **entsprechende extra Business Rules**: DontMoveIntoWaterRule, DontMoveOutsideMapRule implementiert.