

Client Informationen

Funktionsweise des Clients

Die Kommunikation der Clients mit dem Spielserver erfolgt über Sockets. Ein Client erstellt eine Verbindung zum Spielserver mit gegebener Kombination IP-Adresse/Ports (127.0.0.1, 8080 ist die Default-Kombination) herzustellen.

Zu jeder Zeit könnt ihr die angebotenen Befehle der API verwenden, um den Server anzuweisen, eine Aktionen für euch durchzuführen. Die Befehle werden dabei als einfache Strings übertragen: „JOIN PATRICK PLAYER“.

Server Nachrichten werden mittels JSON serialisiert und über den Socket verschickt und müssen dort von euch deserialisiert werden. Der Server sendet zwei Arten von Nachrichten: Info und Request. Ein Spieler-Client wird über eine Nachricht mit „TYPE“:“REQUEST“ benachrichtigt, falls er an der Runde ist, und welcher Zug von ihm erwartet wird „TURN_TYPE“:“CLAIM_DESTINATION_TICKETS“.

Es gibt zwei Arten von Clients: OBSERVER oder PLAYER. Die Art des Clients wird beim Join-Aufruf angegeben. Ein Observer bekommt niemals Requests, aber alle Nachrichten die den Spielfortschritt protokollieren.

Beispiel für Server und Client-Nachrichten:

```
SERVER>
{"TYPE":"REQUEST", "PLAYER":"BLUE", "TURN_TYPE":"JOIN"}
CLIENT> JOIN PATRICK PLAYER
SERVER>
{"TYPE":"INFO", "PLAYER":"BLUE", "SUCCESS":TRUE, "TURN_TYPE":"JOIN", "PLAYER_NAME":
"PATRICK"}
```

Jede Nachricht vom Client an den Server **musst mit einem Zeilenumbruch '\n' enden** und alle Parameter werden durch Leerzeichen voneinander getrennt!

Starten des Servers:

```
JAVA -JAR SERVER.JAR <NR_OF_PLAYERS> <EXPORT_LOG> <SEED>
```

Starten des Dummy-Clients:

TBA

Der Server schreibt ein Spiele-Log, das ausschließlich die JSON-Antworten des Servers sowie den Spielstand am Ende des Spiels enthält.

Ein typischer Spielaufbau:

```
// DER SERVER WARTET AUF DEN JOIN-BEFEHL  
SERVER>{"TYPE":"REQUEST", "TURNTYPE":"JOIN"}
```

```
// EIN SPIELER VERBINDET SICH ALS PATRICK  
CLIENT> JOIN PATRICK PLAYER
```

```
SERVER>  
{"TYPE":"INFO", "PLAYER":"RED", "SUCCESS":TRUE, "TURNTYPE":"JOIN", "CLIENTTYPE":"PL  
AYER", "PLAYERNAME":"PATRICK"}  
SERVER>  
{"TYPE":"REQUEST", "PLAYER":"RED", "TURNTYPE":"CLAIMDESTINATIONTICKETS", "ACTIVE  
DESTINATIONTICKETS": [{"CITY1":"CHICAGO", "CITY2":"NEWORLEANS", "POINTS":7},  
{"CITY1":"MONTREAL", "CITY2":"NEWORLEANS", "POINTS":13}, {"CITY1":"NEWYORK", "C  
ITY2":"ATLANTA", "POINTS":6}]}
```

```
// DER SPIELER BEHÄLT 2 KARTEN  
CLIENT> CLAIMDESTINATIONTICKETS FALSE TRUE TRUE
```

```
SERVER>  
{"TYPE":"INFO", "PLAYER":"RED", "SUCCESS":TRUE, "TURNTYPE":"CLAIMDESTINATIONTICK  
ETS", "DRAWNCARDS":[{"CITY1":"MONTREAL", "CITY2":"NEWORLEANS", "POINTS":13},  
{"CITY1":"NEWYORK", "CITY2":"ATLANTA", "POINTS":6}]}
```

Übersicht der Client-Befehle

	<i>Befehl (<Parameter>)</i>
Verbindungsaufbau	Join <Name> <ClientType> <PlayerColor>
Wagenkarten „offen“ ziehen	DrawPassengerCars <false> < PassengerCarColor >
Wagenkarten „verdeckt“ ziehen	DrawPassengerCars <true>
Zielkarten ziehen	DrawDestinationTickets
Zielkarten behalten	ClaimDestinationTickets <boolean> <boolean> <boolean>
Eine Strecke nutzen	ClaimRoute < Destination > < Destination > < PassengerCarColor >
Aktuelles Spielbrett	BoardState
Alle Strecken des Spielbretts ausgeben	ListAllRoutes

Beispiele:

- CLAIMDESTINATIONTICKETS FALSE FALSE FALSE
- DRAWPASSENGERCARS FALSE RED
- DRAWPASSENGERCARS TRUE
- CLAIMROUTE SAINTLOUIS LITTLEROCK YELLOW

Server Antwort

Eine Json-Antworten vom Server enthält die folgenden Felder:

TYPE : STATUS-UPDATE ODER REQUEST
 TURNTYPE : ANTWORT AUF EUREN ZUG
 SUCCESS : DIE ANFRAGE WAR ERFOLGREICH
 PLAYER : DER SPIELER VON DEM DER ZUG GEMACHT WURDE
 ERRORCODE : EIN FEHLER-CODE (NUR WENN SUCCESS = FALSE)

Typ		Felder der Response
Verbindungsaufbau	Join	playerName clientType
Wagenkarten „offen“ ziehen	DrawPassengerCars	drawnCard faceUpPassengerCarDeck // Array hiddenDeck
Zielkarten ziehen	DrawDestinationTickets	drawnCards // Array
Zielkarten behalten	ClaimDestinationTickets	drawnCards // Array
Eine Strecke beanspruchen	ClaimRoute	d1 // Destination 1 d2 // Destination 2 routeColor // Color of claimed routes passengerCarColors // Array
Aktueller Stand des Spielbretts	BoardState	// board state faceUpPassengerCarDeck topdownPassengerCarDeckCount destinationTicketsCount finalTurn // players state leftPassengerCars drawnPassengerCars drawnDestinationTickets toBeClaimedDestinationTickets ownRoutes
Den kompletten Graphen ausgeben (und wer welche Strecke bebaut hat)	ListAllRoutes	routes // Array
Nächster Zug wird erwartet von player	Turn	turnType // expected turn player // active player success // last request was ... errorCode // errorCode if !successfull finalTurn
Finaler Punktestand	FinalScore	totalScore scorePassengerCars longestRouteLength winner longestRoute claimedTickets nonClaimedTickets

Beispiele:

```

{
  "TYPE": "INFO",
  "ERRORCODE": "RULEVIOLATION",
  "PLAYER": "BLUE",
  "SUCCESS": FALSE,
  "TURNTYPE": "TURN"
}
  
```

```
{  "TYPE":      "INFO",
  "PLAYER":    "RED",
  "SUCCESS":   TRUE,
  "TURNTYPE":  "DRAWPASSENGERCARS",
  "DRAWNCARD": "PURPLE",
  "FACEUPPASSENGERCARDECK":["RAINBOW","RED","RAINBOW","RAINBOW","GREEN"]
}
```

Wichtige (Daten-)Typen

MessageType

Request, Info

ClientType

Player, // Join as player

Observer // Join as observer

TurnType // clients are asked for these kind of turns

Join,

DrawPassengerCars,

DrawDestinationTickets,

ClaimDestinationTickets,

ClaimRoute,

BoardState,

ListAllRoutes,

Turn,

FinalScore,

Unknown

ErrorCode

DoubleRouteClaimed,

NoCardLeft,

RuleViolation,

NotPlayersTurn,

InternalError,

WrongTurnFormat

PassengerCarColor

Purple, White, Blue, Yellow, Orange, Black, Red, Green, Rainbow

PlayerColor

Blue, Red, Green, Yellow, Black

Destination

Chicago, LosAngeles, Montreal, Atlanta, Calgary, Denver, ElPaso,

Houston, LasVegas, NewOrleans, OklahomaCity, Phoenix, Portland,

SaintLouis, SaultStMarie, Washington, SanFrancisco, Toronto, Boston,

Charleston, Dallas, Duluth, Helena, KansasCity, LittleRock, Miami,

Nashville, NewYork, Omaha, Pittsburgh, Raleigh, SaltLakeCity, SantaFe,

Seattle, Vancouver, Winnipeg