



Quality Assurance

Inhaltsverzeichnis

ZUVERLÄSSIGKEIT.....	- 2 -
SERVERGAMETEST.....	- 2 -
LOBBYTEST.....	- 2 -
EFFIZIENZ.....	- 3 -
KOMMENTARE (NORMALISIERT ZU LINES OF CODE).....	- 4 -
ANZAHL LOGGING-STATEMENTS (NORMALISIERT ZU LINES OF CODE)	- 5 -

Zuverlässigkeit

Mit der Zuverlässigkeit als Qualitätsmerkmal wollen wir sicherstellen, dass das Spiel nicht abstürzt. Es liegt uns am Herzen, dass die Spieler das Spiel in voller Länge und möglichst ohne Unterbrechungen erleben können. Bei diesem Qualitätsmerkmal sind viele Klassen in mehreren Packages betroffen. Wir erhoffen uns eine Coverage von 80% und setzen dafür Unit-Tests ein.

Wir haben uns für die Spielerverwaltung in den Lobbies und die Spiellogik entschieden als die zu testende Komponente unseres Games. Denn hier spielen sich die wirklich zentralen Vorgänge ab, und es ist wichtig, dass dies korrekt vonstatten geht.

ServerGameTest

Die ServerGame Klasse verwaltet auf der Serverseite die Spiellogik. Sie ist zuständig für die Organisation der eingeloggtten Clients und die Administration ihrer Punkte und Karten. Ebenfalls sorgt sie dafür, dass die Karten der Spieler nach jeder Runde zurückgesetzt werden. Ebenfalls weiss die Klasse Bescheid darüber, ob ein Spieler noch mitspielt oder er aus der Runde ausgestiegen ist.

Um den Test durchzuführen, erstellen wir eine Lobby mit zwei Playern drin, und dann ein ServerGame. In den Tests wird dann geprüft, ob die Deckgrösse korrekt berechnet wird, ob die Deckgrösse korrekt angepasst wird nach dem Austeilen einer Karte, und ob die Coins aller Spieler korrekt ausgegeben werden.

LobbyTest

Die Lobby Klasse ist auf der Server Seite implementiert und speichert den Namen der aktuellen Lobby, die Administration der Spieler und das dazugehörige Spiel. Ausserdem nimmt die Klasse die Clients an, die reinmöchten, falls die maximale Teilnehmerzahl nicht überschritten ist, und startet das Spiel, wenn alle Spieler in der Lobby sind.

Um den Test durchzuführen, erstellen wir eine Lobby mit zwei Playern drin, und dann ein ServerGame. In den Tests wird dann geprüft, ob ein Spieler korrekt abgewiesen wird, wenn er einer Lobby beitreten will, in der er sich bereits befindet. Ein weiterer Test ist, ob ein Spieler abgewiesen wird, wenn bereits ein Spiel aktiv ist. Schlussendlich wird noch geprüft, ob ein fremder Spieler aus einer anderen Lobby fälschlicherweise als bereit gemeldet werden kann für ein neues Spiel oder eine neue Lobby.

Diese vier Tests testen somit seltene Ausnahmeerscheinungen, die beim Programmieren gerne vergessen gehen.

Effizienz

Mit der Effizienz als Qualitätsmerkmal wollen wir sicherstellen, dass des Spielers so optimiert ist, dass er das gewünschte Ziel auch möglichst schnell und mit wenig Aufwand erreicht wird. Wir möchten das unser Spiel möglichst sparsam bezüglich der Ressourcen, Rechenzeit, Speicherplatz, beim Lösen eines festgelegten Problems ist. Etwa die Hälfte der Klassen müssen dafür getestet werden. Dafür spielen bei unserer Überprüfung nur der Server und Client eine Rolle.

Wir testen den Arbeitsspeichergebrauch in verschiedenen Zuständen und zu verschiedenen Zeiten.

Datum	Server	Client	Notiz
01.04.2020	532.7 MB	-	Verbindung von Server ohne Client
01.04.2020	718.6 MB	1.44 MB	Verbindung von Server und Client - ohne Spiel
04.04.2020	11.459 MB	9.121 MB	Verbindung von Server und Client- mit Spiel
25.04.2020	-	0.789 MB	Verbindung von Server ohne Client
25.04.2020	18.169 MB	16.504 MB	Verbundener Server und Client aber ohne Spiel
25.04.2020	20.515 MB	17.652 MB	Verbindung von Server und Client- mit Spiel

Kommentare (normalisiert zu Lines of Code)

Mit den Kommentaren wollen wir die Lesbarkeit des Codes sicherstellen.

Für aussenstehende oder auch für die anderen Members der Gruppe ist es nicht immer sofort klar, was in der Klasse genau generiert, ausgeführt oder weitergegeben wird. Allerdings ist es sehr wichtig, dass man schneller herauslesen kann welche Argumente wo definiert sind.

Stand: 01.04.20, 18:03 Uhr

Package/Klasse	Codezeilen	Kommentarzeilen	norm.
Chat	146	35	4,17
ChatClient	84	20	
ChatServer	61	15	
Client	553	43	12,86
Client	199	18	
ClientGame	156	7	
ClientHandler	198	18	
Game	421	73	5,76
Card	24	3	
Player	70	3	
ServerGame	297	67	
ServerMatch	30		
GUI	307	18	17,11
GameWindowController	31	4	
LobbyController	95	4	
Login	43	3	
LoginController	110	4	
Main	28	3	
Server	466	86	5,41
Server	129	20	
Lobby	101	25	
ServerHandler	236	41	
Gesamt	1.893	255	7,42

Aus dem jetzigen Standpunkt der Kommentare kann man entnehmen, dass es bisher noch relativ wenig Kommentare gibt. Dies kann man darauf zurückführen, dass die Variablen gut benannt sind und dass bei so vielen Zeilen Code die Gewohnheit alles zu kommentieren noch nicht automatisiert ist.

Anzahl Logging-Statements (normalisiert zu Lines of Code)

Mit den Logging-Statements wollen wir die Fehlersuche vereinfachen.

Dafür verwenden wir das Tool Log4j, dass es ermöglicht auf einfache und komfortable Art, Meldungen auf verschiedener Art auszugeben.

Stand: 04.04.20, 19:31 Uhr

Package/Klasse	Codezeilen	Logging Statements	norm.
Chat	146	10	14,6
ChatClient	84	6	
ChatServer	61	4	
Client	553	25	46,08
Client	199	10	
ClientGame	156	5	
ClientHandler	198	10	
Game	421	7	60,14
Card	24	-	
Player	70	-	
ServerGame	297	7	
ServerMatch	30	-	
GUI	307	21	14,61
GameWindowController	31	4	
LobbyController	95	7	
Login	43	2	
LoginController	110	4	
Main	28	4	
Server	466	17	27,41
Server	129	5	
Lobby	101	3	
ServerHandler	236	9	
Gesamt	1.893	80	23,66