Quality Assurance

Inhaltsverzeichnis

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 Organisierung der eingeloggten 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 Spielt, 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.

Bemerkung: Die Tests haben aufgrund der hohen Kopplung unserer Module nicht funktioniert. Aufgrund unserer Architektur war es nicht möglich, die einzelnen Methoden der ausgewählten Klassen zu testen, da alle Methoden Netzwerkaktionen beinhalten, die man im Test nicht simulieren kann. Man kann sie aber auch nicht umgehen, deshalb war es unmöglich, die relevanten Methoden zu testen.

Effizienz

Mit der Effizienz als Qualitätsmerkmal wollen wir sicherstellen, dass die Aktionen des Spielers so optimiert sind, dass er das gewünschte Ziel auch möglichst schnell und mit wenig Aufwand erreichen kann.
Wir möchten, dass unser Spiel beim Lösen eines festgelegten Problems möglichst sparsam bezüglich der Ressourcen, Rechenzeit und Speicherplatz ist. Dafür spielen bei unserer Überprüfung der Server und Client eine Rolle, das heisst wir nehmen die Werte des Servers und Clients von der Aktivitätsanzeige und analysieren diese.

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

Datum	Server	Client	Notiz
01.04.2020	532.7 MB	-	Starten des Servers 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	-	Starten des Servers ohne Client
25.04.2020	18.089 MB	16.321 MB	Verbundener Server und Client ohne Spiel
25.04.2020	20.515 MB	17.652 MB	Verbindung von Server und Client- mit Spiel
30.04.2020	0.789 MB	-	Starten des Servers ohne Client
30.04.2020	18.345 MB	16.521 MB	Verbundener Server und Client ohne Spiel
30.04.2020	20.467 MB	17.789 MB	Verbindung von Server und Client- mit Spiel
13.05.20	274 MB	-	Starten des Servers ohne Client

13.05.20	275,2 MB	213 MB	Verbundener Server und Client ohne Spiel
13.05.20	443 MB	215 MB	Verbindung von Server und Client- mit Spiel

Der Diskrepanz zwischen den verschiedenen Zuständen ist nachvollziehbar. Im Mai stieg der Gebrauch an Memory, jedoch ist dies durchaus normal und nicht übertrieben. Außerdem steigt unsere Speichernutzung nicht über ein zehn Minuten YouTube Video in schlechter Qualität, was beeindruckend ist.

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. Deshalb sind die Kommentare wichtig, dass man schneller herauslesen kann, wo welche Variablen definiert sind.

Stand: 13.05.20, 10:09 Uhr

Package/Klasse	Codezeilen	Kommentarzeilen	norm.
Chat	146	35	4.17
ChatClient	84	20	
ChatServer	61	15	
Client	553	93	5.95
Client	199	18	
ClientGame	156	57	
ClientHandler	198	18	
Game	421	79	5.33
Card	24	3	
Player	70	3	
ServerGame	297	67	
CoinsCompare	30	6	
GUI	1254	256	4.90
GameWindowController	31	110	
LobbyController	95	30	
Login	43	5	
LoginController	110	31	
CardWindowController	26	3	
ChatInfoWindow	58	3	
ConfirmBox	48	3	
DroppedOutWindow	51	3	

GameInfoWindow	74	3	
HighScoreWindow	72	3	
LobbyList	69	4	
Playerlist	69	4	
PlayMusic	48	12	
PlayVideo	100	6	
RejectJoiningLobbyWindow	61	7	
SelectLobby	93	4	
SelectOptions	116	5	
VariousWindowController	90	20	
Server	574	107	5.36
Server	129	20	
Lobby	101	25	
ServerHandler	236	41	
WriteHighScore	81	18	
Main	27	3	
Gesamt	2948	570	5.17

Stand: 01.04.20 (nicht in der Tabelle ersichtlich)
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.
Stand:13.05.20 (in der Tabelle ersichtlich)
Jetzt zum Ende des Projektes sind einige Klassen dazugekommen, die schon um einiges besser kommentiert sind. Mit der Zeit ist das Kommentieren zur Gewohnheit geworden, aber trotzdem könnten es in manchen Klassen etwas mehr sein. Die Gesamtkennzahl war anfangs April ungefähr 4 und jetzt ist es 5.17. Dies ist ein guter Fortschritt.

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: 13.05.20, 10:48 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	1254	21	14.61
GameWindowController	31	4	
LobbyController	95	7	
Login	43	2	
LoginController	110	4	
CardWindowController	26	1	
ChatInfoWindow	58	1	
ConfirmBox	48	-	
DroppedOutWindow	51	1	
GameInfoWindow	74	1	
HighScoreWindow	72	1	
LobbyList	69	1	
Playerlist	69	1	
PlayMusic	48	1	
PlayVideo	100	2	
RejectJoiningLobbyWindow	61	1	
SelectLobby	93	2	
SelectOptions	116	-	
VariousWindowController	90	4	
Server	574	24	23.92
Server	129	5	
Lobby	101	3	
ServerHandler	236	9	
WriteHighScore	81	3	
Main	27	4	
Gesamt	2948	87	33.88

Die Logging Statements waren sehr hilfreich für das Projekt und wurden von uns in fast allen Klassen verwendet. Sie halfen uns sehr schnell auf die Fehlerquellen zu kommen. Die Gesamtkennzahl war anfangs April 23,66 und jetzt ist es 33,88. Dies ist ein grosser Fortschritt.

Fazit

Schlussendlich ist es uns nicht gelungen die UnitTests funktionsfähig durchzuführen. Deshalb haben wir für die Zuverlässigkeit keine durchschnittliche Code Coverage von 60 % erreichen können. Die Klassen wiesen eine sehr hohe Kopplung auf und waren deshalb sehr schwer zu testen. Man sollte daher die Tests und das Projekt zusammen entwickeln, um die Klassen und die Tests gegenseitig anpassen zu können (eine Architektur vorplanen). Dasselbe gilt auch für die GUI und die Logik, denn durch eine gemeinsame Entwicklung hätte sich die Kopplung nicht weiterhin so stark durch das gesamte Projekt gezogen.
Für die Effizienz wäre es für ein nächstes Mal interessant, die Geschwindigkeit zu messen.
Die Kommentare und Logging-Statements sind erst ein bisschen mehr geworden, nachdem wir mehr darauf geachtet haben.