Bildungszentrum Zürichsee

Projektdokumentation Modul 326

Auftraggeber:	R. Probst
Autor*in:	Yannis Lee, Jonas Rhbary, Thanabalasingam Maruthan
Klasse:	IM19A
Projekttitel:	M326 Memory

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Projektbeschreibung

Wir entwickeln mit der Hilfe der Programmiersprache Java (OOP) ein Memory-Spiel, welches von zwei Spieler gespielt werden kann. Das Spiel läuft über ein GUI, welches mithilfe von Java-Swing implementiert wird. Wir haben dafür 7 Wochen Zeit, in welcher wir auch diese Dokumentation erstellen müssen. Das Memory-Spiel hat zusätzliche Fähigkeiten wie eine Joker-Karte, bei deren Aufdeckung der Spieler eine höhere Anzahl an Punkte bekommt. Ebenso sollen die Spieler mit einem Timer spielen können, der das Spiel erschweren soll. Zusätzlich dazu sollen die Spieler auch die Option haben mehrere Spiele am Stück zu spielen, und den Punktestand dabei zu übernehmen.

Ziele

MUSS-Ziele:

Funktionale Ziele:

- Das Memory-Spiel soll fehlerfrei laufen können.
- Das Memory-Spiel besitzt alle in der Aufgabenstellung erwähnten Funktionalitäten, und diese sind ebenfalls fehlerfrei auszuführen.

Organisatorische Ziele:

- Jede Abgabe, auch die Reflexionen, soll fristgerecht und vollständig eingereicht werden.
- Wir möchten eine Modulendnote von 5.0 erreichen.
- Wir halten uns jede Woche an unseren bereits erstellten Zeitplan.

KANN-Ziele:

Funktionale Ziele:

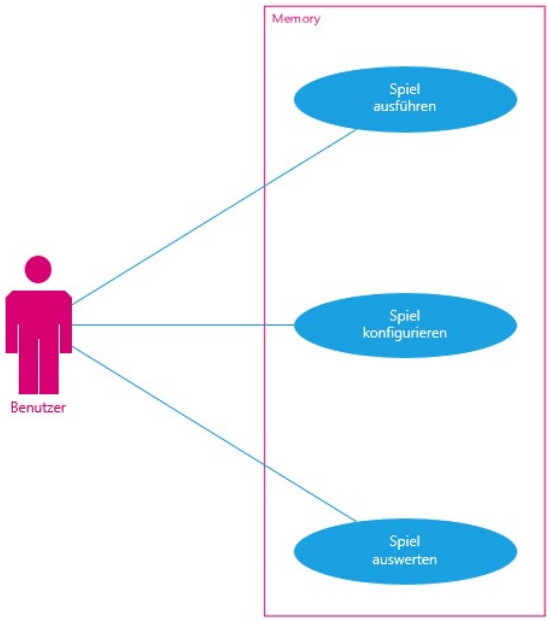
- Das Programm soll drei fakultative Funktionalitäten wie Joker-Karte, Timer und die Speicherung auf einem Datenträger beinhalten.

Organisatorische Ziele:

- Wir möchten dass die Arbeit fair auf alle drei mitarbeitenden Personen aufgeteilt wird.

Anwendungsentwurf

Anwendungsfalldiagramm



Der Akteur hier ist der Benutzer, der das Programm verwendet. Die drei Anwendungsfälle sind einmal das Spiel ausführen, das ist wenn der Programmbenutzer das Memory spielt, das Spiel konfigurieren, dies passiert vor dem Spielen des Memorys, der Benutzer kann dann die verschiedenen Spieleinstellungen auswählen, und der dritte Anwendungsfall ist das Spiel auswerten, das ist wenn sich am Schluss eines Spieles der Benutzer die Punktzahl ausgeben lassen will.

Anwendungsfallspezifikationen

Tabelle 1: Spiel ausführen

ID	1
Name	Spiel ausführen
Kurzbeschreibung	Es werden alle Sequenzen ausgeführt, die bei einem Memory-Spiel gebraucht werden.
Akteur(e)	Benutzer

Auslöser	Das Programm wurde gestartet und die Spieleinstellungen konfiguriert.
Vorbedingung(en)	1. Spiel wurde konfiguriert. 2. Programm läuft fehlerfrei.
Ergebnis	Ein Spieldurchgang wurde absolviert und es sind Ergebnisse, die ausgewertet werden können, vorhanden.
Nachbedingung(en)	Der nächste Anwendungsfall «Spiel auswerten» kann gestartet werden.
Ablauf	1. Das Spiel wird gestartet. 2. Zwei Karten werden vom Benutzer gewählt. 3. Die Karten stimmen nicht überein, der andere Spiel ist am Zug mit Schritt Nr. 2. 4. Die Karten stimmen überein, siehe Alternativen. 5. Das Spielfeld hat keine Karten mehr übrig, das Spiel ist beendet und man kann zum Anwendungsfall «Spiel auswerten» gehen.
Alternativen	4. Die Karten stimmen überein, die Karten werden vom Spielfeld entfernt und der gleiche Spieler ist nochmal am Zug mit Schritt Nr. 2. Der Spieler bekommt einen Punkt.
Fehlerfälle	1. Das Programm stürzt ab. 2. Das Programm reagiert nicht mehr.
Testfälle	1. Bekommt der Spieler einen Punkt bei zwei identischen Karten. 2. Ist der Spieler nochmal an der Reihe bei zwei identischen Karten. 3. Ist der andere Spieler an der Reihe bei zwei nicht identischen Karten.

Tabelle 2: Spiel konfigurieren

ID	2
Name	Spiel konfigurieren
Kurzbeschreibung	Es können alle Spieleinstellungen und Modifikationen für das Memory eingestellt werden.
Akteur(e)	Benutzer
Auslöser	Durch einen Button im Hauptmenü.
Vorbedingung(en)	Das Programm muss gestartet sein.
Ergebnis	Es sind die Einstellungen für das Memory getroffen worden, und das Memory funktioniert jetzt nach diesen Einstellungen.
Nachbedingung(en)	Das Spiel muss fehlerfrei laufen.
Ablauf	1. Man befindet sich im Hauptmenü. 2. Durch einen Button kommt der Benutzer zu den Spielkonfigurationen. 3. Es werden die gewünschten Einstellungen und Änderungen gewählt.

	4. Die Spieleinstellungen wurden gespeichert. 5. Der Benutzer kann jetzt zum 1. Anwendungsfall «Spiel ausführen» gehen.
Alternativen	1. Der Benutzer befindet sich im Hauptmenü. 2. Der Benutzer hat keine Änderungen an den Spieleinstellungen vorzunehmen und geht direkt zum 1. Anwendungsfall «Spiel ausführen».
Fehlerfälle	1. Die Spieleinstellungen werden nicht übernommen. 2. Das Programm stürzt ab. 3. Das Programm reagiert nicht mehr.
Testfälle	1. Die Spieleinstellungen werden übernommen. 2. Durch den Button «Spielkonfiguration» wird der Benutzer auf das Einstellungsmenü geleitet.

Tabelle 3: Spiel auswerten

ID	3
Name	Spiel auswerten
Kurzbeschreibung	Nach einem Spieldurchgang wird das Spiel ausgewertet. Die Ergebnisse werden dem Benutzer ausgegeben.
Akteur(e)	Benutzer
Auslöser	Der Benutzer ist mit seinem Memory-Spiel fertig, das heisst es liegen keine Memorykarten mehr auf dem Spielfeld.
Vorbedingung(en)	Es wurde ein kompletter Spieldurchgang gespielt.
Ergebnis	Die Punkte können dem Benutzer ausgegeben werden.
Nachbedingung(en)	Der Benutzer bekommt die Option das Spiel zu beenden oder nochmals ein Spiel zu spielen und die Punkte vom vorherigen Spiel zu übernehmen.
Ablauf	1. Die Punkte von beiden Spielern werden angezeigt. 2. Es wird ein Gewinner anhand der Punktzahl ausgemacht und ausgegeben. 3. Der Benutzer kann das Spiel beenden.
Alternativen	4. Der Benutzer will noch ein Spiel spielen und kann dies auswählen und die Punkte vom vorherigen Spiel ins nächste Spiel übernehmen.
Fehlerfälle	1. Das Programm stürzt ab. 2. Das Programm reagiert nicht mehr. 3. Die Punkte werden nicht übernommen. 4. Das Programm wird nicht beendet.
Testfälle	1. Das Programm wird beendet wenn der Benutzer das so will. 2. Es wird die richtige Punktzahl für beide Spieler angegeben. 3. Der Benutzer kann ein weiteres Spiel starten.

Anforderungen

Funktionale Anforderungen:

- Das Spiel Memory ist für 2 Spieler anwendbar
- Das Spiel ist zwischen 5x5 und 10x10 Felder gross, wobei 5x5, 7x7 und 9x9 keine Optionen sind, da dies eine ungerade Anzahl an Feldern ergibt.
- Mehrere Spiele können hintereinander gespielt werden, dabei wird die Punktzahl der Spiele vorher übernommen.
- Beim Start des Spieles sind die Kartenrückseiten zu sehen
- Bei einem Zug zieht ein Spieler 2 Karten, diese beiden Rückseiten werden dann angezeigt, sind diese identisch, kriegt der Spieler ein Punkt und bekommt einen neuen Zug.
- Sind die Karten nach einem Zug nicht identisch, werden wieder die Rückseiten der Karten angezeigt und der andere Spieler bekommt einen Zug.
- Wenn alle Kartenpaare aufgedeckt sind, wird das Spiel beendet und der Benutzer bekommt die Option ein weiteres Spiel zu starten oder das Programm zu beenden.
- Der Spieler der mehr Punkte erzielt hat, ist der Gewinner.

Fakultative Anforderungen:

- Es kann Joker-Karten geben, diese geben eine höhere Punktzahl als normale Karten.
- Alle Spiele können einzeln gespeichert werden und als Ausgabe können die Spieler mit Anzahl der Spiele, der erreichten Punkte und dem Wert Punkte pro Spiel ausgegeben werden.
- Es kann einen Timer geben, wenn dieser abgelaufen ist, ist der andere Spieler am Zug.

Nichtfunktionale Anforderungen:

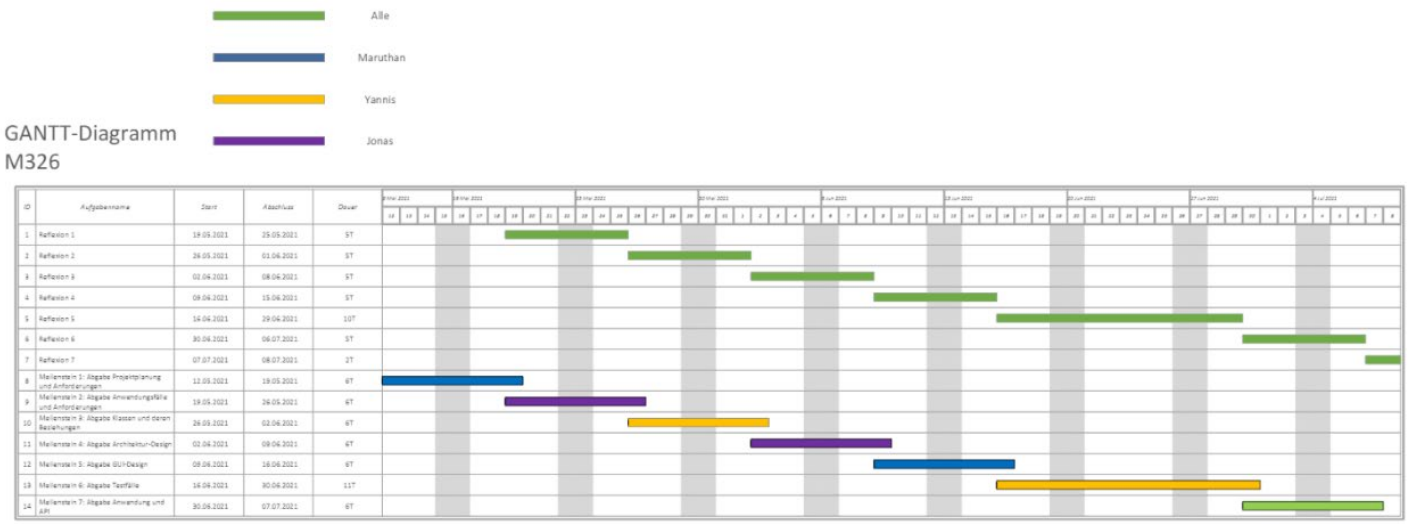
- Das Programm ist in Java mit der OO-Technik geschrieben
- Das GUI wird mit Java-Swing realisiert und es ist das MVC-Pattern zu benutzen.
- Die Umsetzung basiert auf selbst erstellten Klassen-, Sequenz-, Zustands- und Aktivitätsdiagrammen, welche die Anwendung in groben Zügen beschreiben.

Projektplanung

Aufteilung unter den Gruppenmitgliedern:

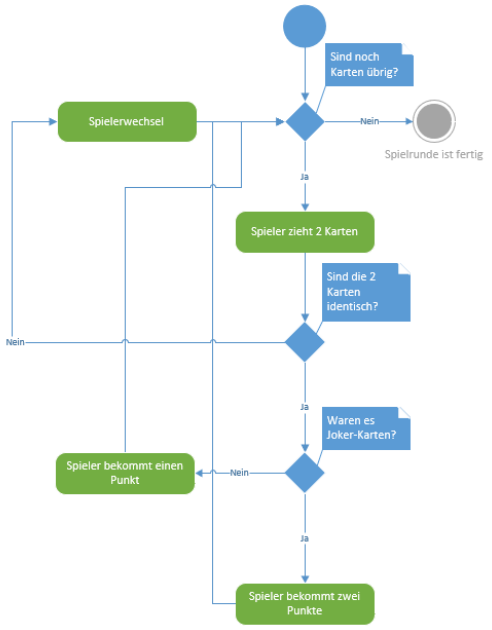
- Wir haben jeweils für jede Woche einen Gruppenchef bestimmt, der die Verantwortung für diese Woche und auch für die jeweilige Abgabe in dieser Woche.
- Wir haben 7 Abgaben, die letzte Abgabe ist diese, in der wir das ganze Programm abgeben müssen, deshalb haben wir diese Woche alle zusammen übernommen.
- Die restlichen 6 Abgaben haben wir durch 3 geteilt, was jeweils 2 Wochen als Gruppenchef für jedes Gruppenmitglied macht.

- Der Gruppenchef übernimmt die Verantwortung für die jeweilige Gruppe und deren Abgabe, das heisst aber nicht dass er der Einzige ist, der diese Woche den Auftrag macht, sondern wir bearbeiten die Aufträge jeweils gemeinsam.

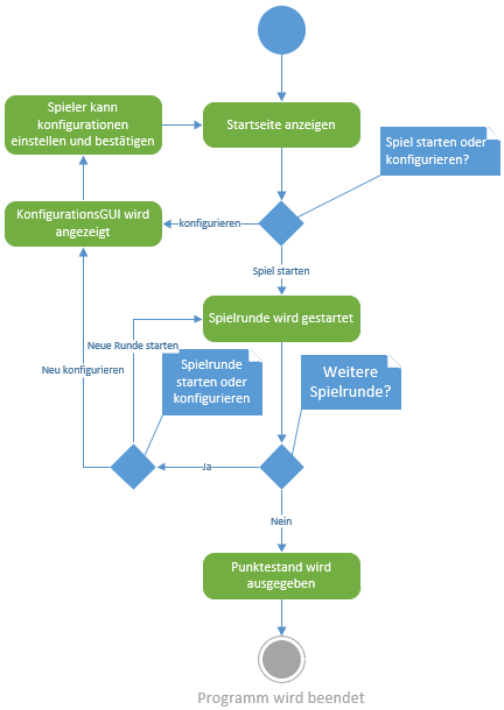


- Zeitliche Aufteilung:
- Wie Sie hier sehen können, haben wir zeitlich eingeplant, dass jede Woche jeder von uns 3 Gruppenmitgliedern eine Reflexion schreiben muss.
- Der Gruppenchef für die erste Woche war Maruthan, danach wird Jonas den 2. Meilenstein übernehmen und Yannis wird sich um die Abgabe der Klassen und deren Beziehungen als Gruppenchef kümmern.
- In der 2. Hälfte des Quartals wird Jonas den 4. Meilenstein «Architektur-Design» leiten und danach wird Maruthan die Abgabe des Gui-Designs übernehmen.
- In der letzten Woche, in der wir einen Gruppenchef zugewiesen haben, wird Yannis die Testfälle machen, und diese natürlich auch testen, bevor er die Verantwortung für deren Abgabe übernimmt.
- In der letzten Woche haben wir uns dazu entschieden dass wir alle zusammen die Verantwortung übernehmen und dann spätestens am 07.07.2021 das ganze Programm «Memory» abgeben.
- Wie Sie ebenfalls sehen können, haben wir geplant dass wir jede Woche einen Meilenstein bearbeiten und diesen auch abgeben werden, da ohne den heutigen Tag vom 12.05.2021 uns 7 Schultage am BZZ übrig bleiben, für 7 Abgaben.

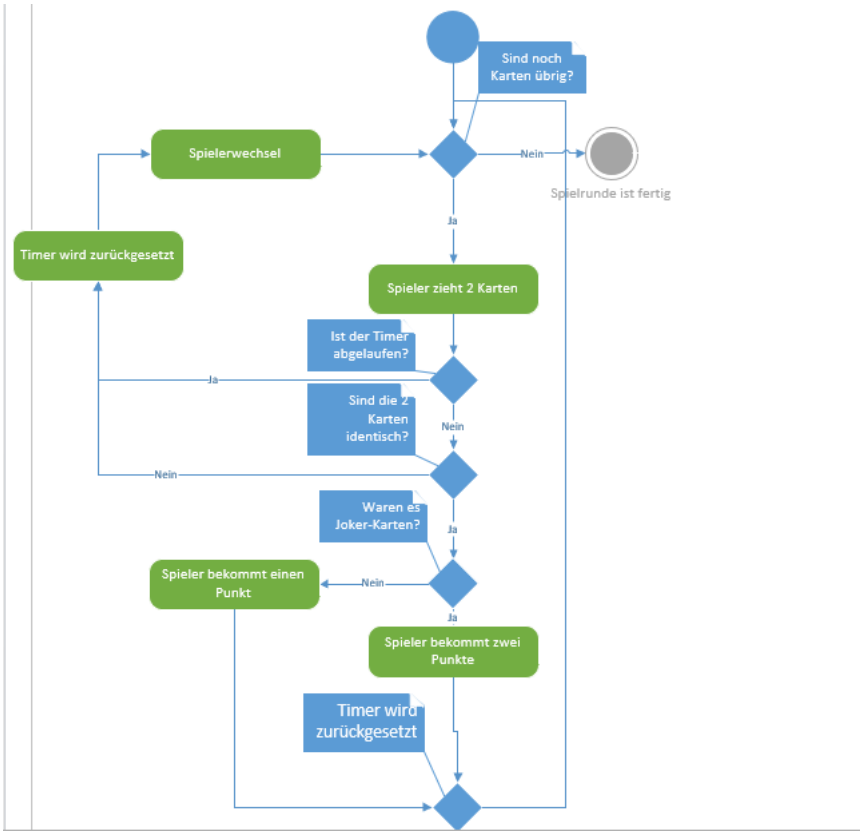
Aktivitätsdiagramme



Dies ist das Aktivitätsdiagramm für eine Spielrunde. Bevor ein Spieler Karten ziehen kann, muss überprüft werden ob es überhaupt noch Karten übrig hat, wenn dies nicht der Fall ist, ist die Spielrunde fertig. Wenn es noch Karten hat, kann der Spieler 2 Karten ziehen, falls diese identisch sind wird überprüft ob es ein Joker-Kartenpaar war, wenn die Karten nicht identisch waren, wird der Spieler gewechselt und es wird wieder überprüft ob es noch Karten hat. Wenn es Joker-Karten waren bekommt der Spieler zwei Punkte, wenn es keine waren bekommt er keinen Punkt und es wird in beiden Fällen wieder überprüft ob noch Karten auf dem Spielfeld sind.



Dies ist das Aktivitätsdiagramm für das gesamte Programm. Beim Starten des Programmes kommt man auf die Startseite wo man zwischen Konfiguration und Spiel starten wählen kann. Wenn man konfigurieren wählt kommt man auf ein KonfigurationsGUI, wo man seine Einstellungen treffen kann, und diese bestätigen kann, danach kommt man zurück auf die Startseite. Wenn man das Spiel startet, läuft eine Spielrunde, wenn diese fertig ist, kann man wählen ob man ein weiteres Spiel spielen will oder nicht. Wenn nicht wird der Punktestand ausgegeben und das Programm beendet. Wenn man weiter Spiele will kann man eine neue Runde starten oder nochmal Konfigurationen treffen.



Spielrundenablauf für den Fall dass ein Timer benutzt wird. Der Unterschied zum normalen Spielrundenablauf ist dass nachdem der Spieler die Karten gezogen hat wird überprüft ob der Timer noch nicht abgelaufen ist. Falls der abgelaufen ist, wird der Timer zurückgesetzt und der Spieler wird gewechselt. Falls er nicht abgelaufen ist geht es normal weiter. Nachdem die Punkte vergeben wurden wird der Timer ebenfalls zurückgesetzt.

CRC-Karten

Tabelle 4: CRC-Karte StartGUI

StartGUI	
Responsibilities: - Fehlerfreie Darstellung der Applikation - Button für die Einstellungen, um zu den Konfigurationen zu kommen und Button für das Spiel, um zum SpielGUI zu kommen - Muss Programm beenden wenn das GUI geschlossen wird	Collaborators: - App - Konfiguration

Tabelle 5: CRC-Karte SpielGUI

SpielGUI	
Responsibilities: - Fehlerfreie Darstellung der Applikation - Hat einen «Abbrechen» Button, welcher die Applikation zurück auf das StartGUI bringt, und alle Spieleinstellungen werden zurückgesetzt	Collaborators: - App - KonfigurationGUI - Kartendeck

- Hat ein Grid für die Memorykarten - Hat eine Punkteanzeige für die beiden Spieler - Nach Ende einer Spielrunde gibt es die Möglichkeit eine weitere Runde zu spielen durch einen Button, oder die Spielstände anzuzeigen - Muss Programm beenden wenn das GUI geschlossen wird	

Tabelle 6: CRC-Karte App

App	
Responsibilities: - Kann ein funktionierendes Memory ausführen - Übernimmt die Einstellungen von der Klasse «Konfiguration» - Übergibt Punkte und Anzahl Spiele an den Spieler - Gibt die Chance nach Ende des Spieles weiterzuspielen - Falls ein Timer ausgewählt wurde, muss der Timer nach jedem Spielzug zurückgesetzt werden	Collaborators: - KonfigurationGUI - Kartendeck - Spieler - StartGUI - SpielGUI

Tabelle 7: CRC-Karte KonfigurationGUI

Konfiguration	
Responsibilities: - Muss ein Dialog öffnen - Muss die Anzahl der Spielkarten einstellen können - Muss gegebenenfalls einen Timer hinzufügen - Muss gegebenenfalls ein Jokerpaar hinzufügen - Muss Default Einstellungen haben	Collaborators: - App - StartGUI

Tabelle 8: CRC-Karte Memorykarte

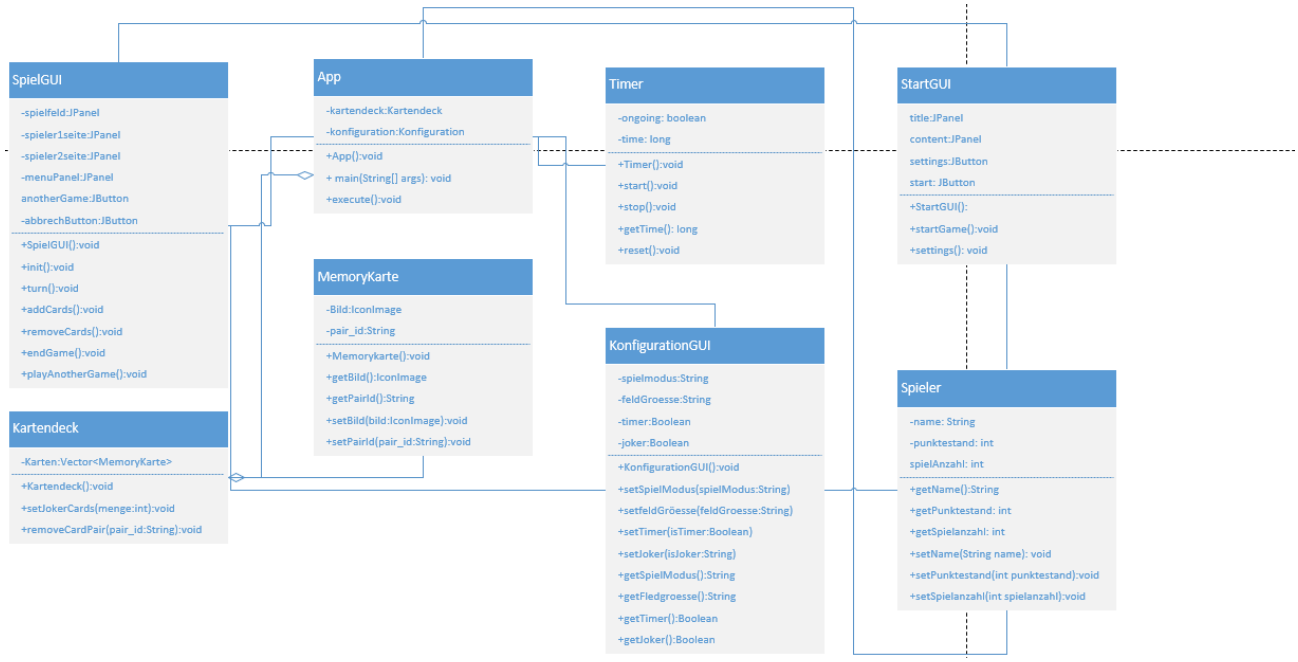
Memorykarte	
Responsibilities: - Müssen vergleichbar sein - Müssen ein Bild speichern können	Collaborators: - Kartendeck

Tabelle 9: CRC-Karte Spieler

Spieler	
Responsibilities:	Collaborators:

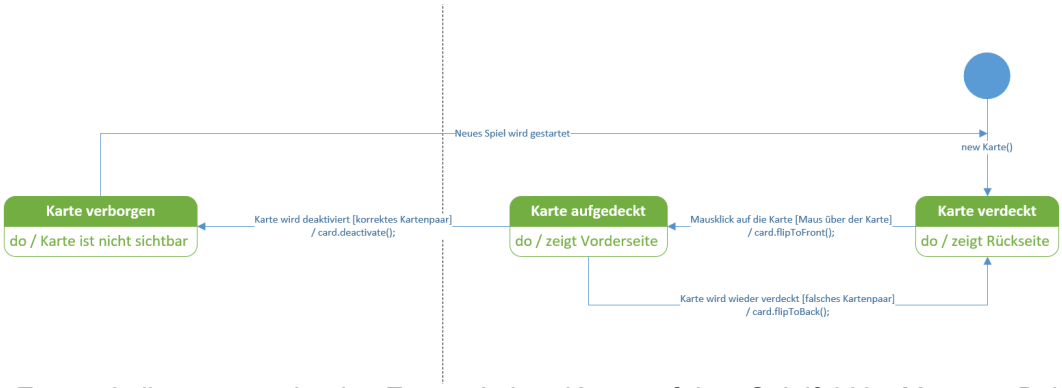
- Spieler soll Spiel spielen können - Soll Punkte speichern können - Hat einen Namen - Anzahl Spieler die der Spieler gespielt hat	- App

Klassendiagramm



Das Klassendiagramm besitzt 3 GUI's eines für den Start und das Menü, eines für die Konfigurationen und eines für die Durchführung des Spieles. Die Memorykarten werden via Kartendeck in die App und dann ins SpielGUI eingefügt, auch der Timer wird via App im SpielGUI angezeigt. Der Spieler beinhaltet die Punktzahl, die Anzahl Spiele die gespielt worden sind, und er hat einen Namen, welche alle ebenfalls über die App im SpielGUI angezeigt werden können. Das KonfigurationsGUI funktioniert als Modalfenster, das aus dem StartGUI heraus angezeigt werden kann.

Zustandsdiagramm



Das Zustandsdiagramm zeigt den Zustand einer Karte auf dem Spielfeld im Memory. Beim starten des Spieles ist die Karte immer verdeckt, wenn dann durch einen Mausklick auf diese Karte die Karte ausgewählt wird, wird Sie aufgedeckt und zeigt die Vorderseite. Wenn die zweite Karte die aufgedeckt wird eine nicht identische Karte ist, wird die Karte wieder umgedreht und zeigt dann wieder dir Rückseite an bis Sie wieder ausgewählt wird. Wenn als zweite Karte jedoch eine identische Karte ausgewählt wird, wird die Karte verborgen und ist nicht mehr sichtbar bis zum Ende der Spielrunde. Eine verborgene Spielkarte kann nur wieder ihre Rückseite anzeigen wenn eine neue Spielrunde gestartet wird.

Sequenzdiagramm(e)

Erstellen Sie mindestens ein Sequenzdiagramm, welches den Programmstart bis hin zum ersten Zug darstellt und beschreiben Sie dieses.

Mockup(s)

Erstellen Sie Mockups (Entwürfe der grafischen Oberfläche) zu Ihrem Spiel und beschreiben Sie diese nachvollziehbar. Gehen sie dabei auf die verwendeten Elemente und Struktur (LayoutManager) ein.

Testfälle

Beschreiben Sie in den Unterkapiteln «JUnit-Testfälle» sowie «Manuelle Testfälle» die Testfälle, welche Sie verwenden, um die fehlerfreie Arbeitsweise Ihrer Anwendung nachzuweisen.

Wichtige Elemente sind:

- *Eindeutige Testfallnummer und Kurzbezeichnung*
- *Beschreibung des Testfalls*
- *Testdaten mit erwartetem Testresultat*

Am Ende müssen Sie die Durchführung der Testfälle angemessen und nachvollziehbar protokollieren. Die Protokolle der Testfälle sind dem Anhang hinzufügen.

JUnit-Testfälle

Beschreiben Sie hier JUnit-Testfälle (mind. 10) gemäss obigem Schema, die Sie umsetzen werden.

Manuelle Testfälle

Beschreiben Sie hier manuelle Testfälle (mind. 5) gemäss obigem Schema, die nicht automatisiert werden können und die Sie manuell ausführen werden.

Lösungsbeschreibung

Beschreiben Sie hier, wie Ihre Lösung aussieht und was sie kann respektive was noch nicht umgesetzt wurde (z.B. wichtige Hinweise zum Programmcode). Bewerten Sie dazu die von Ihnen gesetzten Ziele und Anforderungen und kommentieren Sie den Unterschied zwischen Ihrem IST-Ergebnis und dem ursprünglich definierten SOLL-Zustand. Machen Sie schliesslich einen kurzen Ausblick, wie Ihre Anwendung weiterentwickelt werden könnte.

Erstellen Sie zusätzlich eine API-Beschreibung für Entwickler sowie ein Benutzerhandbuch (inkl. Installationsanleitung) für Anwender Ihres Spiels.

Anhang

Hinweis: Anhänge müssen mindestens in einem Format vorhanden sein, welches ohne Zusatzprogramme dargestellt werden kann. Zu den akzeptierten Formaten gehören Bilddateien (JPEG, PNG, GIF), gängige Microsoft Office Dateien und PDFs.

Folgende Anhänge werden erwartet:

- *Testprotokolle*
- *API-Dokumentation*
- *Benutzerhandbuch*
- *Programm inkl. Source Code*
- *Deklaration Eigenleistung*
- *Optional: Weitere Dateien*