



Praktikum aus Programmierung  
Dr. Michael Hahsler  
WS 2006/2007

Dokumentation des Projektes

Memory

Marco Ruzicka (9851027)

## Inhaltsverzeichnis

1. Problembeschreibung .....	3
2. Analyse.....	3
2.1. Beschreibung der Klassen.....	3
2.1.1. Klasse MemoryApp:.....	3
2.1.2. Klasse Card: .....	3
2.1.3. Klasse CardEvent: .....	4
2.1.4. Klasse CardListener:.....	4
2.1.5. Klasse CardBeanInfo: .....	4
2.1.6. Klasse GameBoard: .....	4
2.1.7. Klasse AboutDlg: .....	5
2.2. UseCase Diagram .....	5
2.3. Klassendiagramm .....	6
2.4. Sequenzdiagramm .....	7
3. Installation und Wartung .....	8
3.1. Installation.....	8
3.2. Wartung.....	8

## Abbildungsverzeichnis

Abbildung 1: Use Case Diagram

Abbildung 2: Klassendiagramm

Abbildung 3: Sequenzdiagramm

# 1. Problembeschreibung

Beim Memory geht es darum das Gedächtnis zu trainieren. Das Original wird mit einem Kartensatz gespielt, der eine bestimmte Anzahl von Bildpaaren enthält. Die Karten werden mit der Bildseite nach unten gelegt.

Es spielen zwei oder mehr Spieler gegeneinander. Wer an der Reihe ist darf zwei beliebige Karten aufdecken. Sind zwei Karten identisch, darf sich der Spieler die Karten behalten und nochmals zwei Karten aufdecken. Dies wiederholt sich so lange bis der Spieler zwei Karten aufdeckt die nicht ident miteinander sind.

Für das Java Projekt soll nur eine einfache Variante mit einem Spieler realisiert werden. Der Spieler spielt in diesem Fall gegen sich selbst und dreht dabei solange die Karten um, bis er alle Kartenpaare gefunden hat.

Als Oberfläche dient ein Hauptfenster, das aus einer einfachen Menüleiste und einem Spielbrett aufgebaut ist. Das Spielbrett soll aus 24 Karten (12 Kartenpaaren) bestehen. Jede dieser Karten soll über eine Vorder- und eine Rückseite verfügen. Das Spiel soll nur aus zwei Dialogen bestehen: Einem der erscheint wenn das Spiel vorbei ist und einen der den Namen des Autors, den Namen des Spiels und die Version anzeigt.

## 2. Analyse

### 2.1. Beschreibung der Klassen

Das Spiel Memory besteht aus folgenden Klassen

#### 2.1.1. Klasse MemoryApp:

Main Klasse des Programmes.

#### 2.1.2. Klasse Card:

Um die Funktion des Dialoges mit möglichst wenig eigener Programmierung umzusetzen ist eine Reihe von Klassenimporten notwendig. Die importierten Klassen stehen als Vorlage für Objekte oder als Basisklasse zur Verfügung. Die Klasse Image aus der AWT Bibliothek wird benötigt um die Bilder der Spielkarten darzustellen. Die Klasse MouseEvent reagiert auf Mausereignisse und die Klasse JComponent wird durch die Klasse Card erweitert.

Die Karte benötigt zwei Objekte des Typs Image um die Bilddaten für die Vorder und Rückseite der Karte darzustellen. Sie verfügt über zwei Zustände turned und playable. Mit turned merkt sich die Karte ob sie auf der Vorder oder Rückseite liegt. Playable gibt an ob die Karte noch im Spiel ist.

Die Methode `paint` hat die Aufgabe die Spielkarte zu zeichnen. Der Kartenzustand kann `turned = true` (Karte ist aufgedeckt) oder `turned = false` (Karte ist zugedeckt) sein.

Die Methode `mousePressed` wird durch die Klasse `MouseAdapter` zur Verfügung gestellt. Die Klasse `Card` überschreibt diese. Ist eine Karte angeklickt kann sie über zwei Zustände verfügen (fixiert oder umgedreht). Ist sie fixiert ist sie nicht mehr spielbar (Die Karte reagiert nicht mehr, solange bis sie wieder spielbar ist.) Im anderen Fall findet die Methode heraus welche Seite oben liegt und dreht den Zustand um.

Die Klasse `MouseAdapter`: Java bietet Verfahren um Ereignisse abzufangen. Eine Art Mausereignisse abzufangen ist der `MouseListener`. Dieser wird zu Beginn der Klasse `Card` über `addMouseListener ()` eingebunden.

### **2.1.3. Klasse CardEvent:**

Diese Klasse hat die Aufgabe als Bote für die Kennung der Karte zu dienen, wenn diese angeklickt wurde.

### **2.1.4. Klasse CardListener:**

Das Interface `CardListener` verfügt nur über eine Methode `turned`. Die Methode `turned` wird immer dann stets aufgerufen, wenn die Karte umgedreht wurde.

### **2.1.5. Klasse CardBeanInfo:**

Diese Klasse sorgt dafür, dass die `JavaBean` von einem GUI-Builder erkannt wird. Ist dies der Fall kann der Gui-Builder an seiner Oberfläche sinnvolle Werte anzeigen

### **2.1.6. Klasse GameBoard:**

Um die entsprechende Funktionalität des Spielbretts zu gewährleisten werden hier aus den Hilfsklassen der AWT- Bibliothek und der Swing Bibliothek Hilfsklassen geladen.

Die Klasse `JOptionPane` erlaubt es ein Nachrichtenfenster anzuzeigen, z.B. für die Anzeige wenn das Spiel beendet ist. Die Klasse `JPanel` bildet die Basisklasse des Spielbretts.

Die Variablen der Klasse sind nicht alle variabel. Die meisten wurden als konstant deklariert. Sie dienen nur dazu, das Spiel z.B. an mehr Karten anzupassen. Die zwei wichtigsten Variablen sind `currentPair` und `pairCounter`. Die erste dient dazu in der Methode `turned` sich die Kennung der umgedrehten Karten zu merken. Die zweite Variable übernimmt die Aufgabe das Spielende festzustellen. Das Spielende ist dann erreicht, wenn `pairCounter` gleich der Anzahl der Bilder ist.

Die Methode `turned` ist der Kern des Spiels. Die Methode startet mit der Abfrage des letzten Paares, das aufgedeckt wurde. Wenn das Array mit Werten größer null belegt ist, handelt es sich um ein Paar ungleicher Karten. D.h. Die Karten müssen wieder verdeckt werden. Anschließend setzt die Methode die Variablen auf `currentPair - 1`. Die Methode wertet das Ereignis `CardEvent` aus, das die Kennung der Spielkarte übermittelt.

### 2.1.7. Klasse AboutDlg:

Zunächst werden eine Reihe von AWT Klassen importiert, die der Dialog als Layout Manager benötigt.

Als Variablen benötigt der Dialog verschiedenen Komponenten die alle aus der Klasse JPanel erzeugt werden.

Der Konstruktor erzeugt ein Objekt des Typs AboutDlg. Hier soll nicht die gesamte Funktionalität der Basisklasse überschrieben werden, deshalb erfolgt der Aufruf des Vorgängers mit super.

## 2.2. UseCase

### 2.2.1 UseCase Beschreibung

*Use-Case Name:* Memory

*Akteur:* Spieler

*Vorbedingungen:* Aufgedrehter PC und Installation mindestens von Java j2re1.4.2\_11

*Auslöser:* Lust auf ein Memory Spiel

*Ablaufbeschreibung:* \*)Memory Spieler hat Lust auf das Spiel.  
\*)Spieler vollzieht seine Spielzüge

*Instanzen:* Marco hat Langeweile und möchte mal wieder Memory spielen, da er leider keine Karten zu Hause hat dreht er seinen Computer auf und startet das Memory Spiel. Er spielt solange er Lust hat und beendet danach das Spiel.

*Ergebnis:* Langeweile wurde beendet

*Nicht funktionale Anforderung:* Zufriedenheit am Spiel

Das Spiel muss auf zwei Hauptfälle reagieren. Im ersten Fall ist noch keine Karte aufgedeckt. Wenn der Spieler auf die erste Karte klickt muss sich diese umdrehen und man muss die Vorderseite sehen. Diese Karte wird fixiert. In diesem Fall darf die Karte auf keine Mausklicks mehr reagieren.

Im zweiten Fall hat der Spieler schon eine Karte aufgedeckt. Dieser Fall verzweigt zu einer Fallunterscheidung. Diese findet heraus ob es sich um die gleiche Karte oder um verschiedene handelt. Sind die Karten gleich werden diese fixiert und reagieren nicht mehr auf Mausklicks und bleiben bis zum Spielende mit der Vorderseite auf dem Spielfeld liegen.

Sind die Karten unterschiedlich bleiben sie einstweilig liegen, werden aber vom Spiel beim nächsten Mausklick auf eine andere Karte umgedreht.

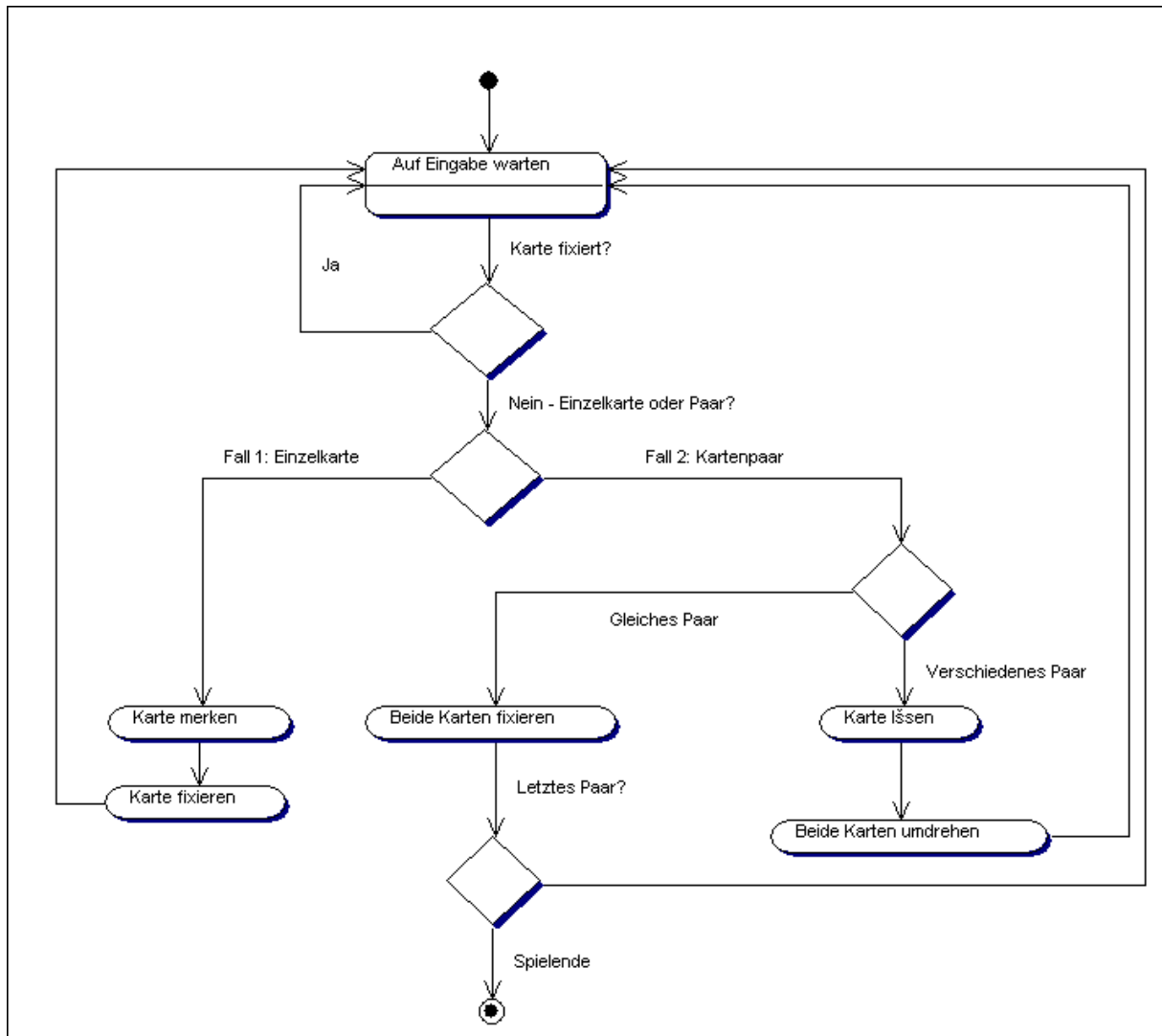


Abbildung 1 Use Case Diagramm

### 2.3. Klassendiagramm

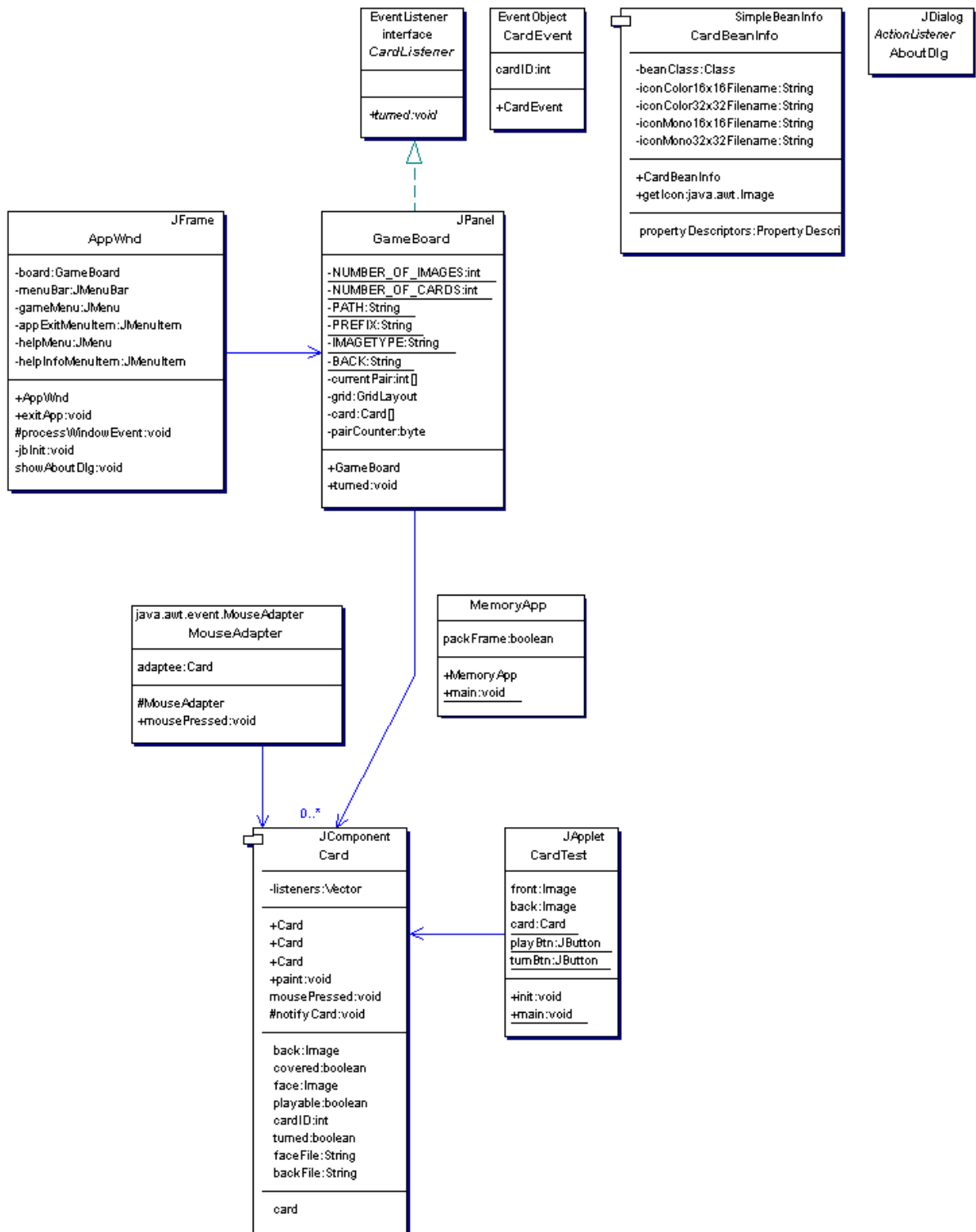


Abbildung 2 Klassendiagramm

## 2.4. Sequenzdiagramm

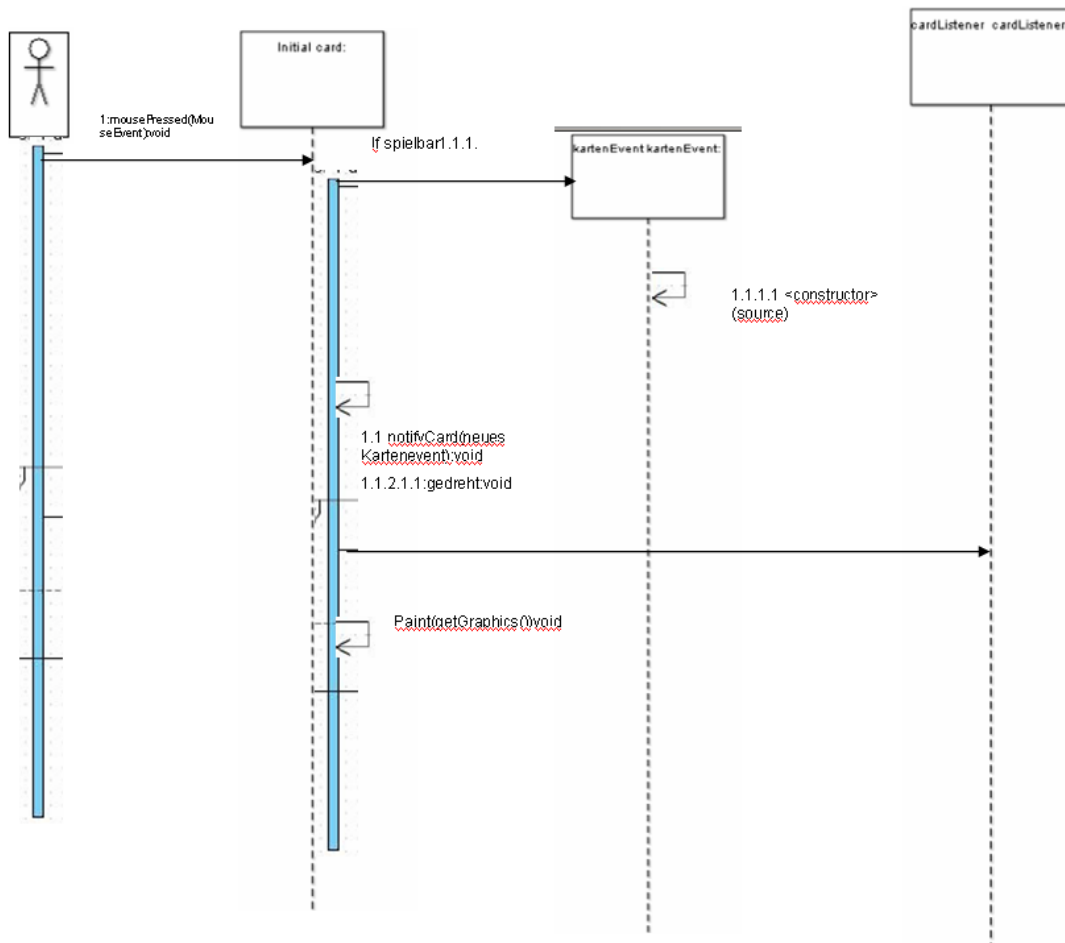


Abbildung 3 Sequenzdiagramm

### 3. Installation und Wartung

#### 3.1. Installation

Um das Spiel auf Ihren Rechner auszuführen ist es notwendig den Ordner Memory lokal auf den PC zu kopieren. Damit das Spiel dann auch über das Startup gestartet werden kann, muss vorher noch der Klassenpfad im Startup Script angepasst werden und der Ort an dem die Applikation abgespeichert wurde

z.B.:

```
SET JAVA_HOME=C:\Data \jdk1.5.0_08\bin
```

```
%JAVA_HOME%\java.exe -cp C:\TEMP\Memory\src MemoryApp
```

#### 3.2. Wartung

Für das Programm ist keine Wartung vorgesehen. Sollten dennoch Problem auftreten, so können Sie mir gerne ein Mail an folgende Adresse schicken

h9851027@wu-wien.ac.at