

DOKUMENTATION

**PROGRAMMIERPRAKTIKUM
SBWL VERTIEFUNGSKURS I**

Priv.Doz. Dr. Michael Hahsler
Abteilung für Informationswirtschaft
Wirtschaftsuniversität WIEN
Augasse 2-6
A-1090 Wien, AUSTRIA



Andrea LEINER

Matrikelnummer: h0150124

e-mail: h0150124@wu-wien.ac.at

Abgabedatum: 13. 01. 2007

INHALTSVERZEICHNIS

Abbildungsverzeichnis	Seite 2
1. Problemdefinitor: MEMORY	Seite 3
2. Analyse	Seite 3
2.1. Spielverlauf	Seite 3
2.2. Use-Case Diagramm	Seite 4
2.2.1. Use-Case Beschreibung	Seite 4
3. Design	Seite 5
3.1. Sequenzdiagramm	Seite 5
3.1.1. Beschreibung des Sequenzdiagramm	Seite 6
3.2. Klassendiagramm	Seite 6
4. Beschreibung der Klassen anhand einzelner Attribute und Methoden	Seite 7
5. Benützung des Programms	Seite 8
6. Installation und Wartung	Seite 9

ABBILDUNGSVERZEICHNIS

Abbildung 1: Use-Case Diagramm	Seite 4
Abbildung 2: Sequenzdiagramm	Seite 5
Abbildung 3: Klassendiagramm	Seite 6

1 PROBLEMDEFINITION:

Spiel „**MEMORY**“

Datum:

13. 01. 2007

Autor:

Andrea LEINER, 0150124, h0150124@wu-wien.ac.at

Ziel:

Ziel dieses Projektes ist, das Spiel „MEMORY“ zu implementieren.

Problembeschreibung/-definition:

Es handelt sich um ein „klassisches“ Memory-Spiel, bei dem 2 Spieler gegeneinander antreten. Es gilt alle Kartenpaare mit möglichst wenigen Versuchen aufzudecken.

Konkret gibt es

- 6 Kartenpaare, die aufgedeckt werden können.

Beim Starten des Spiels bekommen die 2 Spieler automatisch den Status „*SPIELER 1*“ bzw. „*SPIELER. 2*“ zugewiesen.

Zu Beginn jeder Spielrunde werden zunächst die Karten gemischt und darauf folgend beginnt tatsächlich das „Memory-Spiel“. Bei jedem richtig geratenen Pärchen bekommt der Spieler einen Punkt. Sobald der aktuelle Spieler keine 2 zusammengehörenden Karten (= ein Kartenpaar) mehr aufdeckt, verfällt der Zug und der nächste Spieler (2. Spieler) ist an der Reihe.

Dieser Vorgang wiederholt sich solange, bis das gesamte Spielfeld (die 6 Kartenpaare) aufgedeckt ist.

Wurde das gesamte Spielfeld aufgedeckt, gilt die Spielrunde als beendet. Am Ende des Spiel wird die Gesamtpunktezahl und der Name des Siegers automatisch angezeigt. Bevor das Spiel endgültig beendet wird, besteht die Option ein neues Spiel zu starten.

2 ANALYSE:

2.1) Spielverlauf:

Anzahl der Spieler: 2

Beginner: „*SPIELER 1*“

Gewinner: ist derjenige Spieler, der die meisten zusammengehörenden Kartenpaare richtig aufgedeckt hat.

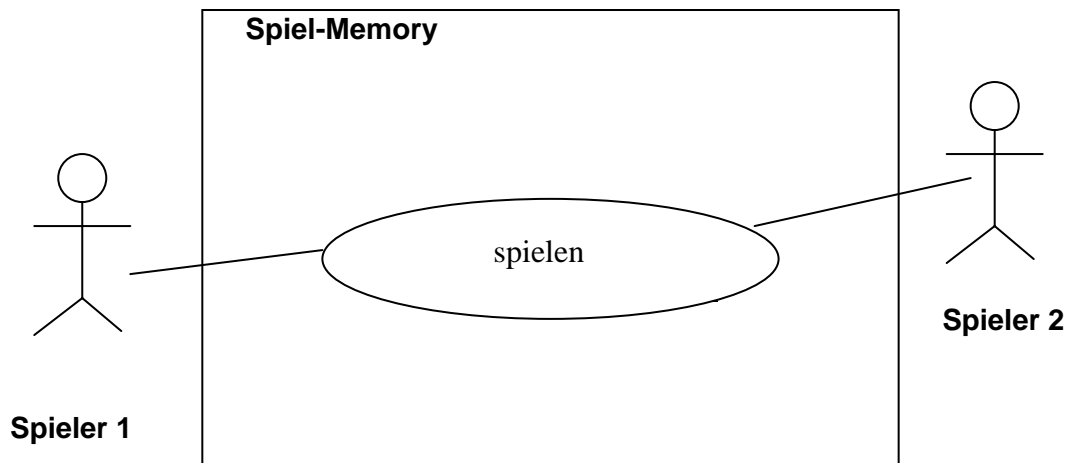
2.2) Use-Case-Diagramm:

Abbildung 1: Use-Case Diagramm

2.2.1) Use-Case Beschreibung:

<i>Use-Case Name:</i>	Spiel – Memory
<i>Akteure:</i>	Spieler 1 und Spieler 2
<i>Vorbedingungen:</i>	eingeschalteter PC; Software JOE (java oriented editing) installiert
<i>Auslöser:</i>	Spaß am Spiel Memory
<i>Ablaufbeschreibung:</i>	<ul style="list-style-type: none"> - Memory -Spieler haben Lust auf das Spiel; - die Spieler vollziehen ihre (Spiel-)Züge (dh sie versuchen abwechselnd so viele richtige Kartenpaare zu erraten, als es, innerhalb des Spiels (siehe Problembeschreibung) möglich ist) und spielen solange sie Lust haben oder alle Kartenpaare richtig erraten wurden.
<i>Fehlsituation:</i>	Einem der beiden Spieler vergeht die „Lust/Laune“ am Spielen und möchte nicht mehr weiterspielen, somit wird der Spielverlauf gestört.
<i>Instanzen:</i>	<ul style="list-style-type: none"> - Andrea und Daniela ist es fad und möchten aus diesem Grund wieder einmal Memory spielen. Da alle anderen Freundinnen, der beiden, wegen einer bevorstehenden Schularbeit an einem Spiel im Freien verhindert sind, setzten sich Andrea und Daniela zum Computer, schalten ihn ein und starten bzw. spielen anschließend das Memoryprogramm mittels JOE. - Felix ist „Denksportmeister“ der 3. Klasse der Volksschule Podersdorf am See. Da es ihm wahnsinnig großen Spaß bereitet, auch in der Freizeit sein Gehirn zu „trainieren“ und ihm gerade sehr langweilig ist, beschließt Felix spontan, an einem Nachmittag den PC einzuschalten und das Spiel Memory mittels JOE zu spielen.
<i>Ergebnisse:</i>	Langeweile wurde beendet bzw. zumindest für eine gewisse Zeit unterbrochen.
<i>Nicht-funktionale Anforderungen:</i>	Zufriedenheit im Spiel
<i>Autor:</i>	Andrea LEINER

3 DESIGN:

3.1) Sequenzdiagramm:

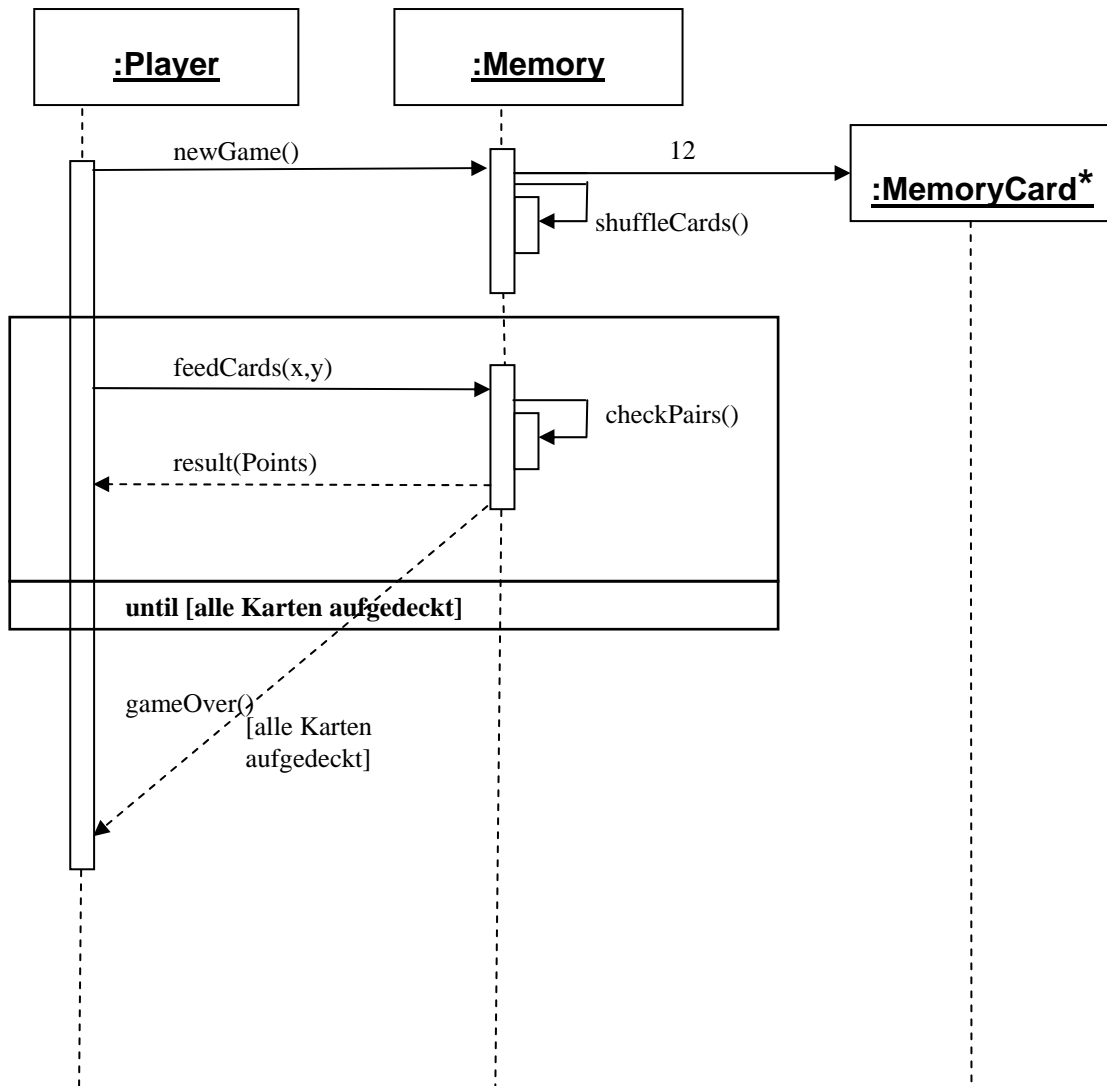


Abbildung 2: Sequenzdiagramm

3.1.1) Beschreibung des Sequenzdiagramm:

In diesem Sequenzdiagramm soll der grundlegende Ablauf dieses Memory-Spiels dargestellt werden.

Das Sequenzdiagramm besteht aus den Objekten „**Player**“, „**Memory**“ und „**MemoryCard**“. Der Ablauf gestaltet sich nun folgendermaßen:

- vom Objekt **Player** wird die Nachricht (*newGame()*, dh ein neues Spiel soll gestartet werden) an das Objekt **Memory** gesendet;
- das Objekt **Memory** „greift“ daraufhin auf das Objekt **MemoryCard** zu, um letztendlich auf die 12 Memory-Karten (bzw. 6 Kartenpaare) Zugriff zu haben und sie für die nun folgenden Abläufe benutzen zu können;
- weiters sendet nun das Objekt **Memory** die Nachricht (*shuffleCards()*) an sich selbst, mit der „Aufforderung“, die Karten zu mischen;
- die nächste (zeitliche) Abfolge ist, dass vom Objekt **Player** die Nachricht (*feedCards(x,y)*) an das Objekt **Memory** gesendet wird, dh mit dieser Nachricht wird nun bezweckt, dass Karten(nummern) vom Player eingegeben werden und das Objekt **Memory** auf die gewünschten Karten zugreift;
- darauf folgend verschickt das Objekt **Memory** wieder eine Nachricht (nämlich *checkPairs()* – „überprüfe, ob ein Kartenpaar vorliegt“) an sich selbst und gibt eine Nachricht bzw. Antwort (*result(Points)*) mit dem Ergebnis (Punkteanzahl) an das Objekt **Player** zurück;
- dieser Ablauf wird solange wiederholt, bis die Bedingung *until[alle Karten aufgedeckt]* erfüllt ist;
- ist diese Bedingung [*alle Karten aufgedeckt*] erfüllt, wird die Nachricht bzw. Antwort (*gameOver()*) an das Objekt **Player** gesendet.

3.2) Klassendiagramm:

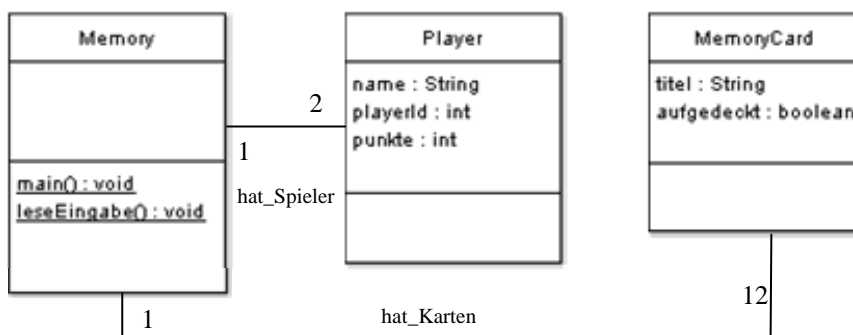


Abbildung 3: Klassendiagramm

4 BESCHREIBUNG DER KLASSEN ANHAND EINZELNER ATTRIBUTE UND METHODEN:

Das Spiel MEMORY besteht aus den Klassen

- class **Memory**
- class **Player** und
- class **MemoryCard**.

Die Beziehungen zwischen den Klassen gestalten sich folgendermaßen:

das Spiel MEMORY hat

- 12 Memory-Karten (dh 6 Kartenpaare) und
- 2 Spieler.

Nun folgt eine genauere Beschreibung der einzelnen Klassen:

Memory:

Dies ist die umfangreichste Klasse des Programms. Einstiegspunkt des Programms ist die *main* Methode, sie ist jene Methode, die zum Programmstart ausgeführt wird. Weiters werden in dieser Klasse

- die beiden Spieler (Spieler 1 und 2) erzeugt, in dem die Variablen *spieler 1* und *spieler 2* vom Typ *Player* angelegt, der Konstruktor zum Erzeugen der neuen Objekte aufgerufen und Werte für die Instanzvariablen gesetzt;
- die Anzahl der Kartenpaare (nämlich 6 Kartenpaare) und
- die Variable *karten* vom Typ *vector* angelegt (hält alle Karten, die im Spiel sind).
 - Mit der Methode *add()* werden die Memory-Karten angelegt und dem *Vector* hinzugefügt (insgesamt gibt es 12 Karten, entspricht 6 Pärchen);
 - die Methode *shuffle()* sorgt dafür, dass die Memory-Karten, die sich im *Vektor* befinden, gemischt werden;
 - die Methode *parseInt()* löst in dieser Klasse folgendes Problem: die Methode *leseEingabe()* liest die Eingabe des Users aus und gibt eine Variable vom Typ *String* zurück; mit Hilfe der Methode *parseInt()* kann aus einem *String* (also *Text*) ein *int* (eine Zahl) gemacht werden.
 - die Methode *get()* wird hier konkret verwendet, um eine Memory-Karte an einer bestimmten Stelle im *Vektor* zu erhalten;
 - mit der Methode *equals()* ist es möglich, (2 *Strings* zu vergleichen und liefert *true*, wenn beide gleich sind, sonst *false*), diese Methode wird u. a. angewandt, um die Titel zweier Memory-Karten zu vergleichen und herauszufinden, ob ein Kartenpaar vorliegt.
 - die Methode *leseEingabe()* wird in der Klasse *Memory* verwendet, um Eingaben von der Kommandozeile zu lesen. Konkret wird mit einem *BufferedReader* in Kombination mit *InputStreamReader* die Eingabe des Users (Spielers) gelesen.
zB werden unter anderem die Spieler innerhalb des *Memory-Spiels* aufgefordert, Ziffern für die erste und zweite Memory-Karte, die aufgedeckt werden soll, in die Kommandozeile einzugeben, mit Hilfe der Methode *leseEingabe()* ist es nun möglich, diese Eingaben zu lesen.

Player:

Die Klasse *Player* (mit den Instanzvariablen *name*, *playerId* und *punkte*) bildet einen Spieler ab, der bestimmte Eigenschaften besitzt und zwar folgende:

<i>name:</i>	der Name des Spielers
<i>playerId:</i>	eindeutige Kennung des Spielers
<i>punkte:</i>	die Punkte, die der Spieler gesammelt hat (Anmerkung: pro richtig aufgedecktem Kartenpaar erhält der Spieler einen Punkt)

MemoryCard:

Die Klasse *MemoryCard* stellt eine (Memory-)Karte dar, die nun jene Eigenschaften besitzt:

<i>titel:</i>	bezieht sich auf den Text, der auf der Karte angezeigt wird, damit der Spieler unterscheiden kann, welche konkrete Karte er nun aufgedeckt hat
<i>aufgedeckt:</i>	ist der Status der Karte, wenn sie aus dem Spiel ist (dh wenn sie als Pärchen bereits aufgedeckt wurde, dann kann die Karte nicht noch einmal aufgedeckt werden)

Die Methode *reset()* setzt den Status der Karte zurück, wenn ein neues Spiel begonnen wurde.

4 BENÜTZUNG DES PROGRAMMS:

Nachdem das Spiel mittels JOE (java oriented editing) gestartet wurde, erscheint folgende *Textausgabe auf der MS-DOS-Maske:*

"MEMORY"

Bitte einfach eine Zahl von 1-12 eingeben, jede Zahl ist eine Karte.

Spieler 1 du bist dran.

Erste Karte?"

durch Eingabe einer Zahl mittels der Tastatur und drücken der Taste „ENTER“ erscheint darauf folgend, um welche Memorykarte (es wurden Tierpaare für das Spiel Memory gewählt) es sich handelt.

Anschließend wird der User aufgefordert, eine zweite Karte aufzudecken, indem er eine weitere Zahl mittels Tastatur eingibt und auf „ENTER“ drückt; auch jetzt erscheint wieder auf der MS-DOS-Maske, um welche Karte bzw. Tier es sich handelt und des weiteren wird überprüft, ob der aktuelle Spieler ein Pärchen aufgedeckt hat.

- Wenn er ein Kartenpaar aufgedeckt hat, dann darf er weiterspielen (gleiche Vorgangsweise, wie bereits oben erklärt) und der Spieler bekommt zusätzlich einen Punkt für jedes richtig erratene Kartenpaar.
- Wenn der aktuelle Spieler kein Kartenpaar erraten hat, wird der andere Spieler aufgefordert, Karten aufzudecken.

Textausgabe auf der MS-DOS-Maske:

„Leider kein Pärchen, der andere Spieler ist dran.“

Das Spiel wird solange fortgesetzt (gleicher Ablauf, wie bereits oben erklärt) bis alle Kartenpaare richtig erraten wurden.

Letztendlich erscheint folgender *Text auf der MS-DOS-Maske:*

"Spiel zu Ende
Spieler ... hat gewonnen
Punktstand:
Spieler 1 hat ... Punkte
Spieler 2 hat ... Punkte
Noch einmal spielen (j/n)?"

Wird die Taste „j“ auf der Tastatur gedrückt, beginnt ein neues Spiel, bei drücken der Taste „n“ wird das Spiel beendet.

5 INSTALLATION UND WARTUNG:

Installation:

Um das Spiel auf Ihrem PC ausführen zu können, ist es zunächst notwendig den Code auf dem jeweiligen PC (in einem von Ihnen selbst gewählten Ordner) abzuspeichern. Anschließend ist es möglich, das Spiel zB über **JOE** (= *java oriented editing*) zu starten.

Damit das Spiel nun tatsächlich „gespielt“ bzw. gestartet werden kann, müssen noch folgende Schritte, nach aufrufen des JOE, beachtet werden:

1. **Kompilieren** Sie die **Klassen Memory, MemoryCard und Player**
(dh sie öffnen die 3 Klassen mit JOE und klicken darauffolgend auf das Symbol „Kompilieren“ in der Symbolleiste bzw. auf die Taste F9)
2. *Nachdem die Kompilierung abgeschlossen ist, klicken Sie anschließend auf das Symbol „Ausführen“ in der Symbolleiste bzw auf die Taste Strg+F9)*

(Anmerkung: Das Kompilieren und Starten des Spiels ist selbstverständlich auch im Programm **eclipse** möglich.)

Wartung:

Für das Programm ist grundsätzlich keine Wartung vorgesehen. Treten dennoch Fehler auf, können Sie gerne eine E-mail mit einer Problembeschreibung an die folgende Adresse senden: h0150124@wu-wien.ac.at.