# Dokumentation der Praktischen Arbeit zur Prüfung zum Mathematisch-technischen Softwareentwickler

16. April 2015

**Jonas Hamers**

Prüfungs-Nummer:

Programmiersprache: Java

# Inhaltsverzeichnis

## H In- und Output der Testdokumentation

# 1. Aufgabenanalyse

## 1.1. Analyse

Es soll das Spiel Nim implementiert werden. Nim ist ein Spiel, welches zu zweit mit Streichhölzern gespielt wird, welche zu Beginn auf eine feste Anzahl sogenannter Reihen verteilt sind. Bei diesem Spiel gibt es einige Regeln zu beachten. Zwei Spieler ziehen jeweils abwechselnd und nehmen jeweils in ihrem Zug mindestens ein oder mehrere Streichhölzer aus einer der Reihen weg. Die Maximalanzahl der Streichhölzer spielt keine Rolle, jedoch müssen sie ein und derselben Reihe angehören. Der Spieler, welcher das letzte Streichholz entnimmt, gewinnt das Spiel.
Bei dieser Implementierung beträgt die maximale Anzahl an Reihen neun, die maximale Anzahl an Streichhölzern in einer Reihe 10. Demzufolge können maximal 90 Streichhölzer auf dem Feld liegen. Weiterhin sollen zwei Spielstrategien entwickelt werden. Die Strategie des Spieler 1 soll wenn möglich immer gewinnen. Spieler 2 hingegen zieht alle Hölzer einer Reihe weg, sofern nur noch in einer Reihe Hölzer liegen. Ist die Anzahl an belegter Reihen größer als eins, zieht Spieler 2 aus einer zufällig gewählten Reihe eine zufällig ausgewählte Anzahl an Hölzern weg.
Zu einer gegebenen Startverteilung sollen zehn voneinander unabhängige Spiele durchgeführt werden und dabei die Anzahl der gewonnenen Spiele für jeden Spieler gezählt werden. Wenn vorhanden sollen die Züge eines gewonnenen Spiels eines Spielers notiert werden. Ziel ist es das Spiel lauffähig zu implementieren und eine möglichst gewinnorientierte Spielstrategie für Spieler 1 zu entwickeln.

## 1.2. Eingabeformat

Die Eingabedateien enden auf ".in". Am Anfang einer Eingabedatei stehen beliebig viele Kommentarzeilen welche mit einem "#" beginnen. Es muss jedoch mindestens eine Kommentarzeile mit einer kurzen Beschreibung existieren. Nach der letzten Kommentarzeile folgt eine Zeile mit der Startverteilung des Spiels. Gültig sind durch Leerzeichen getrennte ganze Zahlen größer 0 und kleiner 10. Mindestens eine Zahl und maximal neun Zahlen müssen gegeben sein. Beispiel einer gültigen Eingabedatei:

```
#IHK Beispiel 1
3 4 5
```

## 1.3. Ausgabeformat

Die Ausgabedateien enden auf ".out" und sind gleichnamig zu den jeweiligen Eingabedateien. Jede Kommentarzeile aus der Eingabedatei soll eins zu eins übernommen werden. Darauf folgt die Startverteilung, die prozentuale Verteilung der gewonnenen Spiele sowie die Züge der von den Spielern gewonnenen Spiele, wenn vorhanden. Beispiel einer Ausgabedatei (korrespondierend zum Beispiel einer Eingabedatei s.o.):

#IHK Beispiel 1
Startverteilung: (3,4,5)
Gewonnene Spiele Spieler 1: 100%
Gewonnene Spiele Spieler 2: 0%
Beispiel eines von Spieler 1 gewonnenen Spiels:
Zug 1, Spieler 1 : (3,4,5) > (1,4,5)
Zug 2, Spieler 2 : (1,4,5) > (1,4,0)
Zug 3. Spieler 1 : (1,4,0) > (1,1,0)
Zug 4, Spieler 2 : (1,1,0) > (0,1,0)
Zug 5, Spieler 1 : (0,1,0) > (0,0,0)
Beispiel eines von Spieler 2 gewonnenen Spiels:
Nicht vorhanden.

## 1.4. Anforderungen an das Programm

Das Programm arbeitet nach dem MVC-Entwurfsmuster. Die Main-Funktion delegiert die Verarbeitung an den Controller. Der Controller liest die Daten ein, und speichert sie in das Model, bearbeitet diese und gibt das Ergebnis über die View aus. Es ist dabei notwendig auf Fehler angemessen zu reagieren. Das Programm soll mit einer aussagekräftigen Meldung beendet werden und nicht abstürzen. Die Robustheit des Programms wird anhand von Testfällen überprüft.

## 1.5. Sonderfälle

Durch die Analyse der Aufgabenstellung und des Eingabeformates ergeben sich einige Startverteilungen die äußerst günstig und ungünstig für Spieler 1 sind: Günstige Startverteilungen:

- Es existiert nur eine Reihe
- Es existieren zwei Reihen mit unterschiedlich vielen Hölzern
- Es existieren drei Reihen, bei denen zwei Reihen gleich viele Hölzer beinhalten
- Es existieren drei Reihen, bei denen mit ein Zug eine Verteilung (1,2,3) erzeugt werden kann
- Es existiere vier Reihen mit der Verteilung (1,2,3,x) , mit beliebigen x

Ungünstige Startverteilungen:

- Es existieren zwei Reihen mit einer geringen, gleichen Anzahl an Hölzern

## 1.6. Fehlerfälle

Fehler können auftreten wenn die Eingabedatei ein unerwünschtes Format beinhaltet:

- Keine Kommentarzeile vorhanden
- Erste Zeile nach der Kommentarzeile enthält nicht nur Ziffern
- Ziffern sind nicht durch ein einfaches Leerzeichen getrennt
- Eine Zahl ist gleich 0
- Es werden mehr als 9 Ziffern aufgelistet
- Es wird keine Ziffer aufgelistet

# 2. Verfahrensbeschreibung

## 2.1. Vorgehensweise

Nach dem Einlesen der Eingabedatei und dem Überprüfen auf Korrektheit wird, sofern das Eingabeformat nicht eingehalten worden ist, eine Ausgabedatei mit der Benachrichtigung einer falschen Eingabe erstellt. Ist hingegen das Eingabeformat eingehalten worden, startet das Spiel Nim und ein Feld mit der vorgegebenen Startverteilung wird erstellt. Solange das Spielfeld Streichhölzer enthält fordert die Kontrolle-Klasse die Spieler abwechseln auf einen Zug zu tätigen und gibt diesen anschließend aus.
Zur Problemlösung eine geeignete, möglichst gute, Spielstrategie für Spieler 1 zu entwickeln wurden einige Sonderfälle betrachtet: Es gibt einige Spielsituationen die auf jeden Fall zu einem Sieg führen. Um eine gute Spielstrategie zu entwickeln ist es daher nötig eine solche Situation zu erschaffen. Ziel es dabei immer die Situation zu erhalten, immer den Gegner dazu zu zwingen ein letztes Streichholz aus der vorletzten bestehenden Reihe zu entnehmen, um selber die letzte Reihe leeren zu können.
Sofern nur noch zwei Reihen existieren ist es optimal, wenn man immer so zieht, sodass die Anzahl der Streichhölzer in beiden Reihen gleich ist.
Wenn die aktuelle Anzahl der belegten Reihen gleich drei ist, sollte man grundsätzlich versuchen die Spielsituation (3,2,1) zu kreieren. Gibt es die Spielsituation jedoch schon her, dass die Anzahl zweier Reihen gleich ist, so sollte die dritte Reihe komplett leer gezogen werden, da man so zu einer gewünschten Spielsituation mit zwei Reihen gelangt. Kann beides nicht erreicht werden, so gilt, dass man stets versucht die Anzahl der insgesamt liegenden Streichhölzern gerade zu halten. Diese Strategie greift auch wenn die Anzahl der Reihen größer oder gleich vier ist. Die einzige Einschränkung wäre eine Spielsituation mit vier Reihen, wenn eine Reihe ein, eine Reihe zwei und eine Reihe drei Streichhölzer enthält, da man mit dem Wegnehmen der vierten Reihe die gewünschte Spielsituation mit drei Reihen erhält.
Gelangt man selbst durch das Wegnehmen einer geraden Anzahl an Streichhölzern in eine der gewünschten Situationen sollte man immer nur ein Streichholz entfernen um die Anzahl der verbliebenen Spielzüge möglichst groß zu halten, damit die Zufallsstrategie einen Fehler begehen kann.

Es folgt ein Beispiel, welches die Strategie veranschaulicht:
# Eigenes Beispiel
Startverteilung: (1,2,3,4,5,6)

Gewonnene Spiele Spieler 1: 100%
Gewonnene Spiele Spieler 2: 0%
Zug 1, Spieler 1: (1,2,3,4,5,6) -> (0,2,3,4,5,6)
Zug 2, Spieler 2: (0,2,3,4,5,6) -> (0,2,1,4,5,6)
Zug 3, Spieler 1: (0,2,1,4,5,6) -> (0,2,1,4,5,4)
Zug 4, Spieler 2: (0,2,1,4,5,4) -> (0,2,1,3,5,4)
Zug 5, Spieler 1: (0,2,1,3,5,4) -> (0,2,1,3,2,4)
Zug 6, Spieler 2: (0,2,1,3,2,4) -> (0,2,1,3,2,1)
Zug 7, Spieler 1: (0,2,1,3,2,1) -> (0,2,0,3,2,1)
Zug 8, Spieler 2: (0,2,0,3,2,1) -> (0,1,0,3,2,1)
Zug 9, Spieler 1: (0,1,0,3,2,1) -> (0,0,0,3,2,1)
Zug 10, Spieler 2:(0,0,0,3,2,1) -> (0,0,0,2,2,1)
Zug 11, Spieler 1:(0,0,0,2,2,1) -> (0,0,0,2,2,0)
Zug 12, Spieler 2:(0,0,0,2,2,0) -> (0,0,0,2,1,0)
Zug 13, Spieler 1:(0,0,0,2,1,0) -> (0,0,0,1,1,0)
Zug 14, Spieler 2:(0,0,0,1,1,0) -> (0,0,0,1,0,0)
Zug 15, Spieler 1:(0,0,0,1,0,0) -> (0,0,0,0,0,0)

In den ersten vier gespielten Zügen von Spieler 1 (Zug Nr. 1,3,5,7) zieht Spieler 1 eine beliebige Anzahl Streichhölzer weg, sodass eine gerade Anzahl an Streichhölzern liegen bleibt. Im Zug 9 erkennt Spieler 1, dass mit dem Wegnehmen einer Reihe die gewünschte Spielsituation mit drei Reihen erschaffen werden kann (3,2,1). Egal wie Spieler 2 reagieren würde wird Spieler 1 nun das Spiel gewinnen, denn nun besteht immer die Chance eine gewünschte Spielsituation mit zwei Reihen zu erhalten für Spieler 1. Im 11. Zug kann er eine (2,2) Situation und im 13. Zug eine (1,1) Situation erzwingen.

## 2.2. Gesamtsystem

### 2.2.1. Main-Funktion

Die Main-Funktion erzeugt den Controller IControl. An diesen wird der Dateipfad weitergegeben. Das Spielgeschehen erfolgt nun in IControl.

### 2.2.2. Model

Hier sind die wesentlichen Klassen, die die Spieldaten und Algorithmen enthalten und auf ihnen operieren. Es ist kein expliziter Zugriff auf die Daten möglich, lediglich über die Schnittstelle IControl kann auf diesen zugegriffen werden. Zum Model gehören die zwei Interfaces IModel, welches die Spielinfos enthält, und IAlgo, welches die beiden Spielstrategien für Spieler 1 und Spieler 2 enthält.

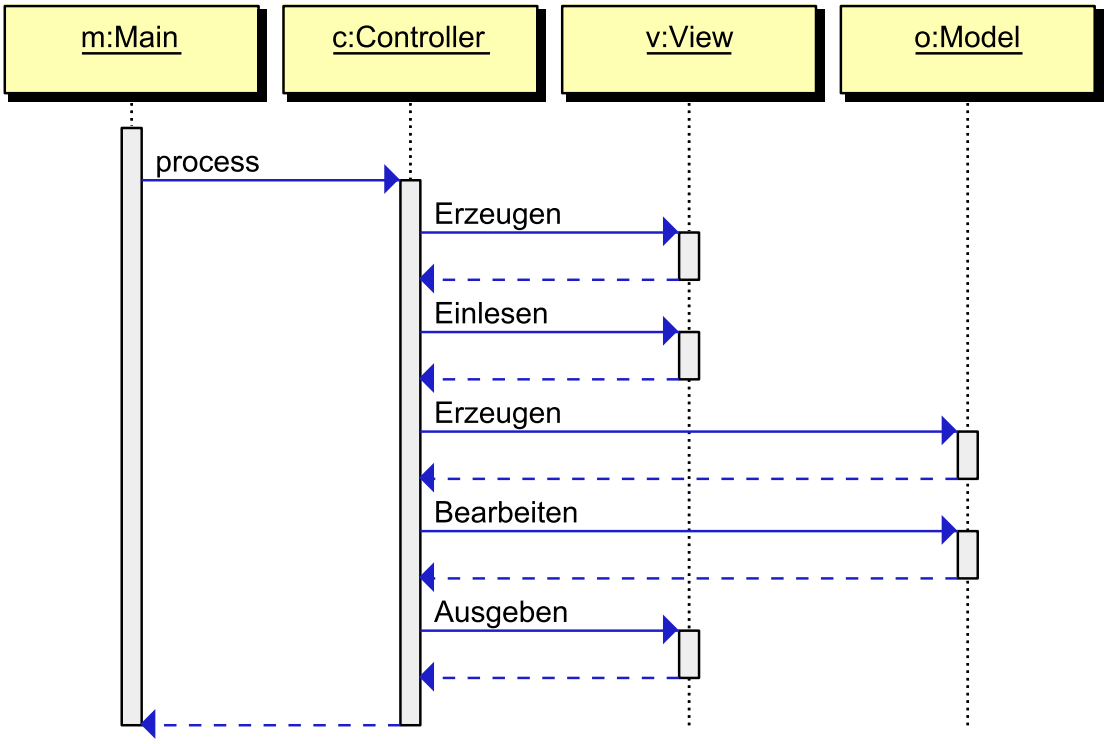
### 2.2.3. View

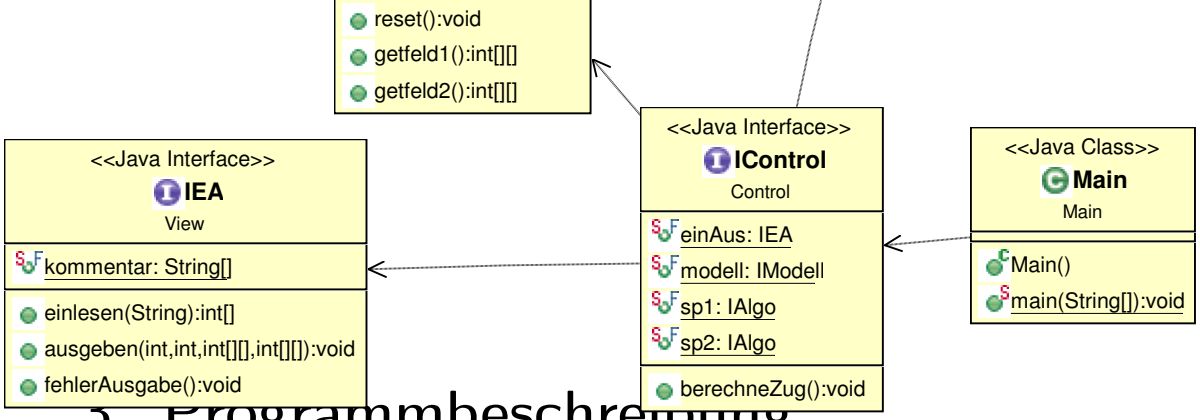
Die Ein-/Ausgabe wird in der View vorgenommen. Dazu besteht für Input und Output jeweils ein vorgegebenes Eingabe- und Ausgabeformat. Beim Einlesen der Daten soll dabei schon auf Korrektheit überprüft werden.

### 2.2.4. Controller

Im Controller werden die Daten über die View in das Model eingelesen. Mithilfe der Spielinformationen vom Model kann der Controller das Spiel leiten.

## 2.3. Datenfluss






# 3. Programmbeschreibung

## 3.1. Pakete

Die Hauptmodule des Programms sind nach dem Drei-Schichten-Modell aufgeteilt. Zudem ist für jedes Aufgabenfeld zunächst ein Interface deklariert, auf diese Weise bleibt das Programm erweiterbar. Gemäß .NET-Konvention beginnen die Namen der Interfaces mit einem „I“.

# 4. Testdokumentation

# 5. Zusammenfassung und Ausblick

# A. Abweichungen und Ergänzungen zum Vorentwurf

# B. Benutzeranleitung

# C. Entwicklungsumgebung

| | |
|---|---|
| Programmiersprache | : |
| Compiler | : |
| Rechner | : |
| Betriebssystem | : |

# D. Verwendete Hilfsmittel

-

# E. Erklärung

Erklärung des Prüfungsteilnehmers / der Prüfungsteilnehmerin:

Ich versichere durch meine Unterschrift, dass ich das Prüfungsprodukt selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Das auf den beiden identischen CDs abgelegte Prüfungsprodukt entspricht der gedruckten Version.

Jülich, den 16. April 2015

Ort und Datum

Unterschrift des Prüfungsteilnehmers

# Anhang F

# Aufgabenstellung

# Anhang G

# Quellcode

1. **Sourcedateien**
   XX Seiten
2. **Skripte**
   XX Seiten
3. **Inline-Dokumentation**
   XX Seiten

# Anhang H

# In- und Output der Testdokumentation