

# **JAVA – Projekt**

## **„Mensch ärgere dich nicht“**

Rechnerpraktikum aus Programmierung, WS06/07

Unter der Leitung von Dr. Michael Hahsler

**Autor:**

Name: Patrick Siebert

Matrikelnummer: 0252978

## **Inhalt:**

<b>PROBLEMBESCHREIBUNG:</b> .....	<b>4</b>
Projektziel:.....	4
Problemstellung: .....	4
<b>ANALYSE UND MODELLIERUNG:</b> .....	<b>4</b>
<b>Use Case 1</b> .....	<b>4</b>
Use Case Name: .....	4
Akteure:.....	4
Vorbedingungen:.....	4
Nachbedingungen:.....	4
Auslöser: .....	5
Ablaufbeschreibung: .....	5
Fehlsituationen: .....	5
Variationen:.....	5
Instanz: .....	5
Ergebnisse: .....	5
Nicht funktionale Anforderungen: .....	5
Autor: .....	5
Use Case Diagramm:.....	5
<b>Use Case 2</b> .....	<b>6</b>
Use Case Name: .....	6
Akteure:.....	6
Vorbedingungen:.....	6
Nachbedingungen:.....	6
Auslöser: .....	6
Ablaufbeschreibung: .....	6
Fehlsituationen: .....	6
Variationen:.....	6
Instanz: .....	6
Ergebnisse: .....	6
Nicht funktionale Anforderungen: .....	6
Autor: .....	6
Use Case Diagramm:.....	6
<b>Class Diagramm:</b> .....	<b>6</b>
<b>Sequenz-Diagramm:</b> .....	<b>8</b>
<b>WEITERES DESIGN:</b> .....	<b>9</b>
Class Diagramm überarbeitet:.....	9
Sequenz-Diagramm überarbeitet: .....	9
<b>IMPLEMENTIERUNG:</b> .....	<b>11</b>
<b>Beschreibung der Klassen:</b> .....	<b>11</b>
Klasse Figur: .....	11
Klasse Mensch: .....	11
Klasse Spiel:.....	11
Klasse Spielbrett:.....	11

Klasse Spieler:.....	12
Klasse wuerfel:.....	12

**INSTALLATION UND WARTUNG:..... 12**

## **Problembeschreibung:**

### ***Projektziel:***

Ziel des Projekts ist es, das beliebte Gesellschaftsspiel „Mensch ärgere dich nicht“ in einer Software zu realisieren.

### ***Problemstellung:***

Es soll das Gesellschaftsspiel „Mensch ärgere dich nicht“ entwickelt werden. Man kann es gegen einen, zwei oder drei Computer-Gegner spielen. Das „Spielbrett“ besteht aus insgesamt 40 Feldern, von denen vier als Startfelder (0, 10, 20, 30) und vier als Zielfelder (39, 9, 19, 29) dienen (jeder Spieler besitzt also ein Start- und ein Zielfeld).

Jeder Spieler erhält vier Spielfiguren. Jede dieser Spielfiguren muss alle 40 Felder ablaufen, damit sie ins Ziel kommt. Mittels eines Würfels wird ermittelt, wie viele Felder die Figur vorrückt. Wenn alle vier Spielfiguren im Ziel sind, ist das Spiel gewonnen. Kommt eine Figur auf ein besetztes Feld, wird die bereits dort stehende Spielfigur zurück an den Start geworfen.

Um eine Figur ins Spiel bringen zu können, muss am Anfang gewürfelt werden. Wenn innerhalb der ersten drei Würfe ein Sechser fällt, muss die Figur auf das Startfeld gestellt werden. In diesem Fall kann derselbe Spieler nochmals würfeln und einen Stein bewegen. Gelingt es nicht, dass in den ersten drei Würfeln ein Sechser gewürfelt wird, kommt der nächste Spieler an die Reihe.

Wird ein Sechser während des Spiels gewürfelt, muss eine neue Figur auf das Startfeld gestellt werden, außer alle Figuren sind schon im Spiel.

Eventuell soll es eine Hilfefunktion geben, wo man die Regeln nachsehen kann.

(Der Autor hofft dieses Projekt umsetzen zu können, ohne sich zu oft an den Ratschlag des Titels halten zu müssen)

## **Analyse und Modellierung:**

### ***Use Case 1***

**Use Case Name:** Mensch-ärgere-dich-nicht Spiel

**Akteure:** Spieler, Computergegner

**Vorbedingungen:** Keine

**Nachbedingungen:** ein Spieler hat gewonnen

**Auslöser:** Spielstart

**Ablaufbeschreibung:**

1. Spiel starten
2. Gegneranzahl wählen
3. Würfeln
4. Spielfigur wählen und auf Startbereich stellen.
5. Würfeln
6. Figur ziehen.
7. Nächster Spieler
8. Wiederholung Punkte 3-7
9. Spielende

**Fehlsituationen:**

- Mehr als 3 Mitspieler gewählt
- Falsche Figur gewählt

**Variationen:**

- Spiel wird zwischendurch abgebrochen
- Nach dem Spiel wird sofort ein neues Spiel gestartet
- Spieler sieht in den Regeln nach

**Instanz:**

Spieler 1 startet ein neues Spiel. Er sucht sich 2 Gegner aus. Er würfelt die Zahl 6. Er wählt Spielfigur 1 und stellt sie auf den Startbereich. Er würfelt noch mal. Es ist die Zahl 4. Er fährt mit seiner Figur 4 Felder vor. Die beiden Computergegner kommen dran. Diese Punkte wiederholen sich, bis der Spieler alle Figuren im Ziel hat.

**Ergebnisse:**

Spieler 1 hat gewonnen. Seine 4 Figuren sind alle im Ziel.

**Nicht funktionale Anforderungen:**

Spieler 1 ärgert sich nicht.

**Autor:**

Patrick Siebert am 10.11.2006

**Use Case Diagramm:**



## **Use Case 2**

**Use Case Name:** Regeln lesen

**Akteure:** Spieler

**Vorbedingungen:** Keine

**Nachbedingungen:** Regeln wurden angezeigt

**Auslöser:** Spieler versteht die Regeln nicht

**Ablaufbeschreibung:**

1. Spiel starten
2. Regeln lesen

**Fehlsituationen:**

Regeln werden nicht verstanden

**Variationen:**

keine

**Instanz:**

Spieler 1 startet das Spiel. Er liest die Regeln.

**Ergebnisse:**

Spieler 1 kennt die Regeln.

**Nicht funktionale Anforderungen:**

Spieler 1 kennt die Regeln gut.

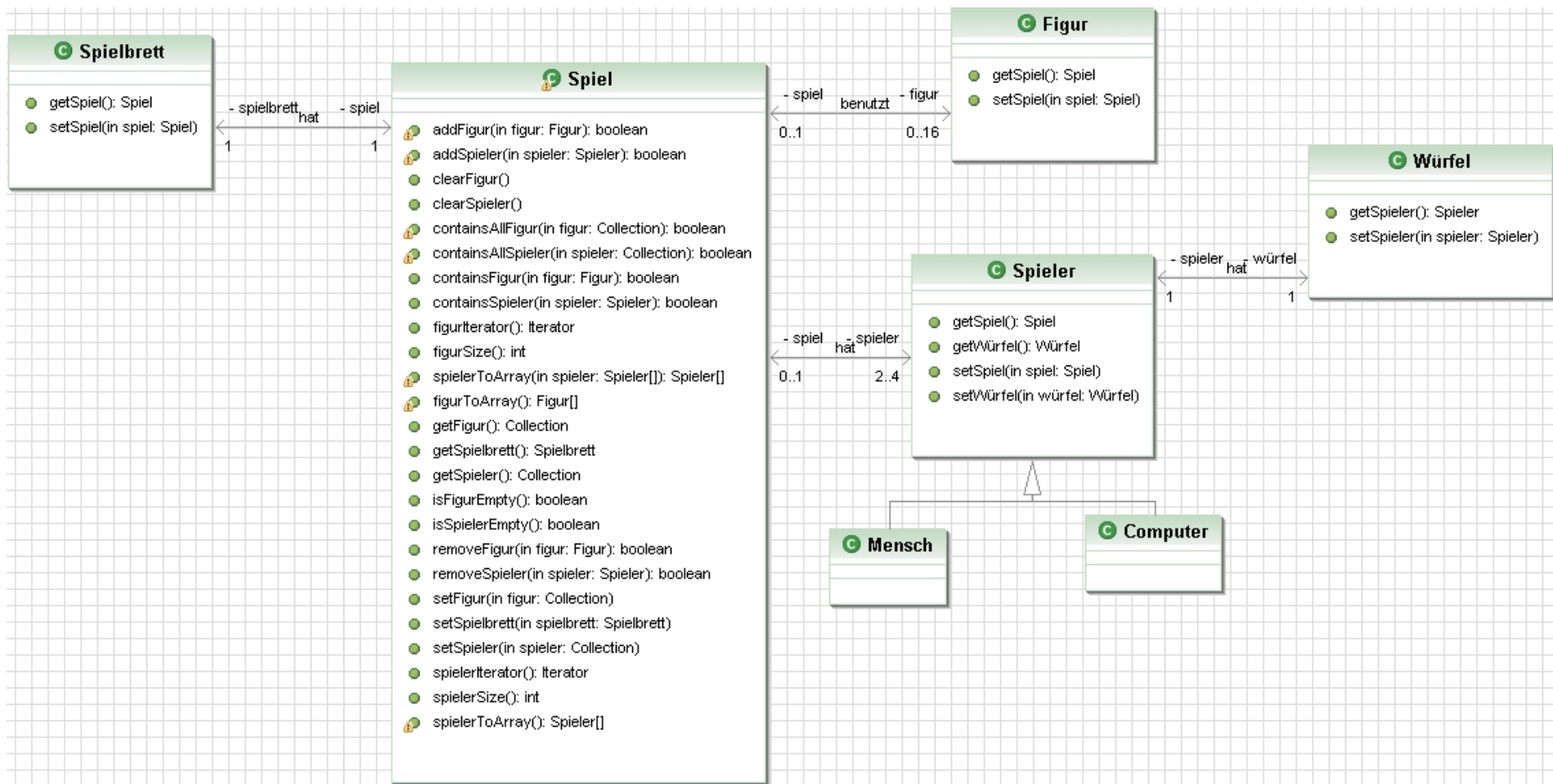
**Autor:**

Patrick Siebert am 22.11.2006

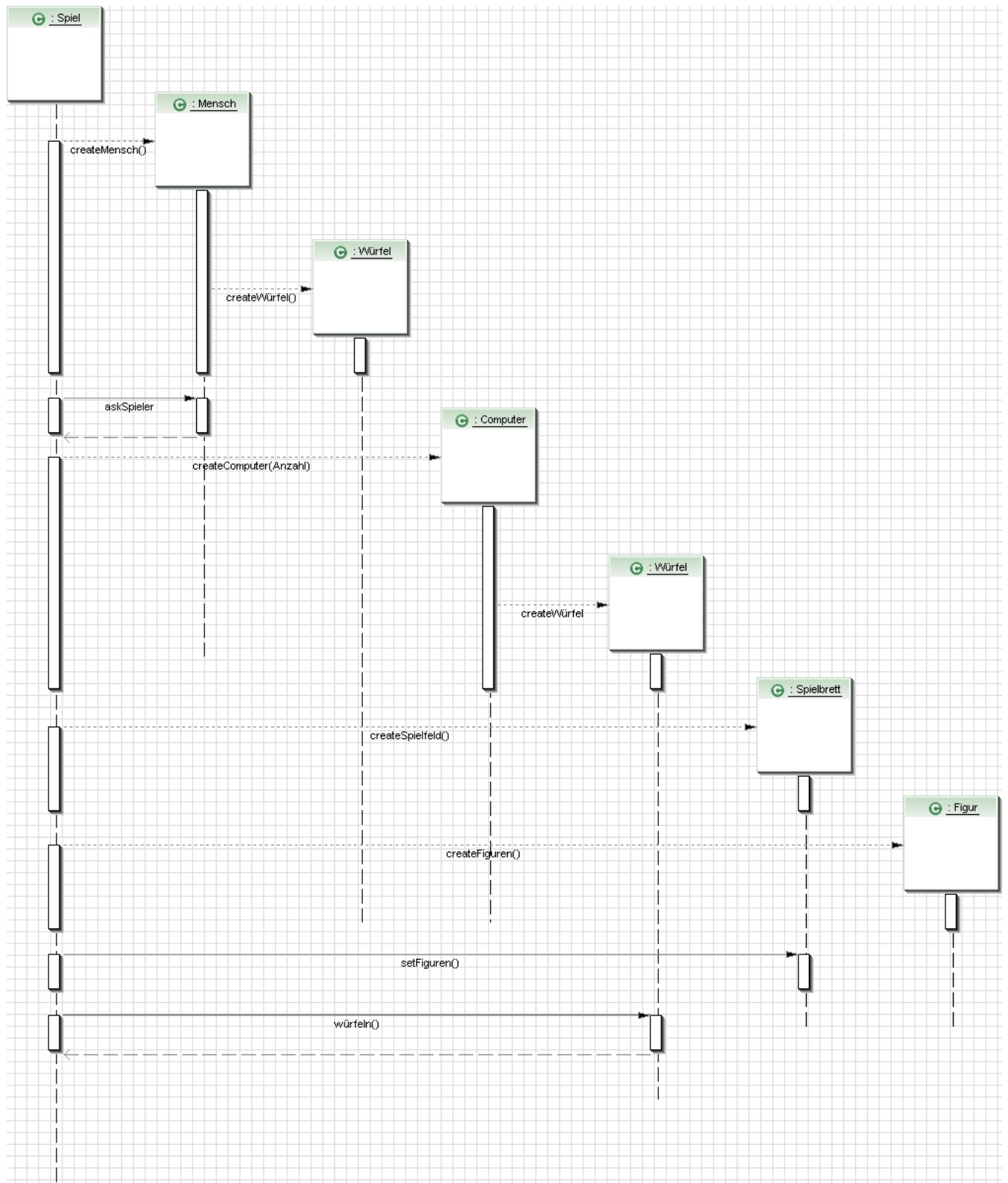
**Use Case Diagramm:**



**Class Diagramm:**



## Sequenz-Diagramm:



## Weiteres Design:

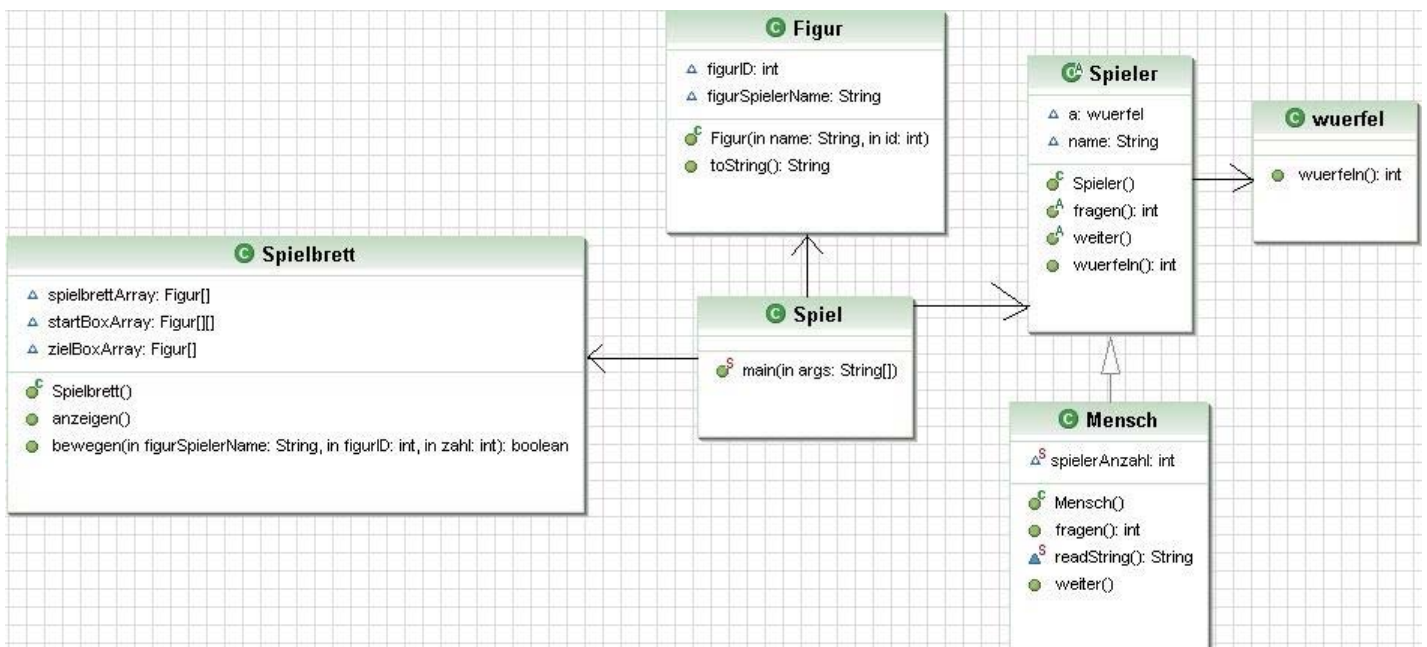
Aufgrund der Knappheit der zeitlichen Ressourcen musste das Projekt drastisch vereinfacht werden. Es stellte sich heraus, dass die oben beschriebene Darstellung den Rahmen des Semesters sprengen würde.

Folgende Vereinfachungen wurden unter anderem vorgenommen:

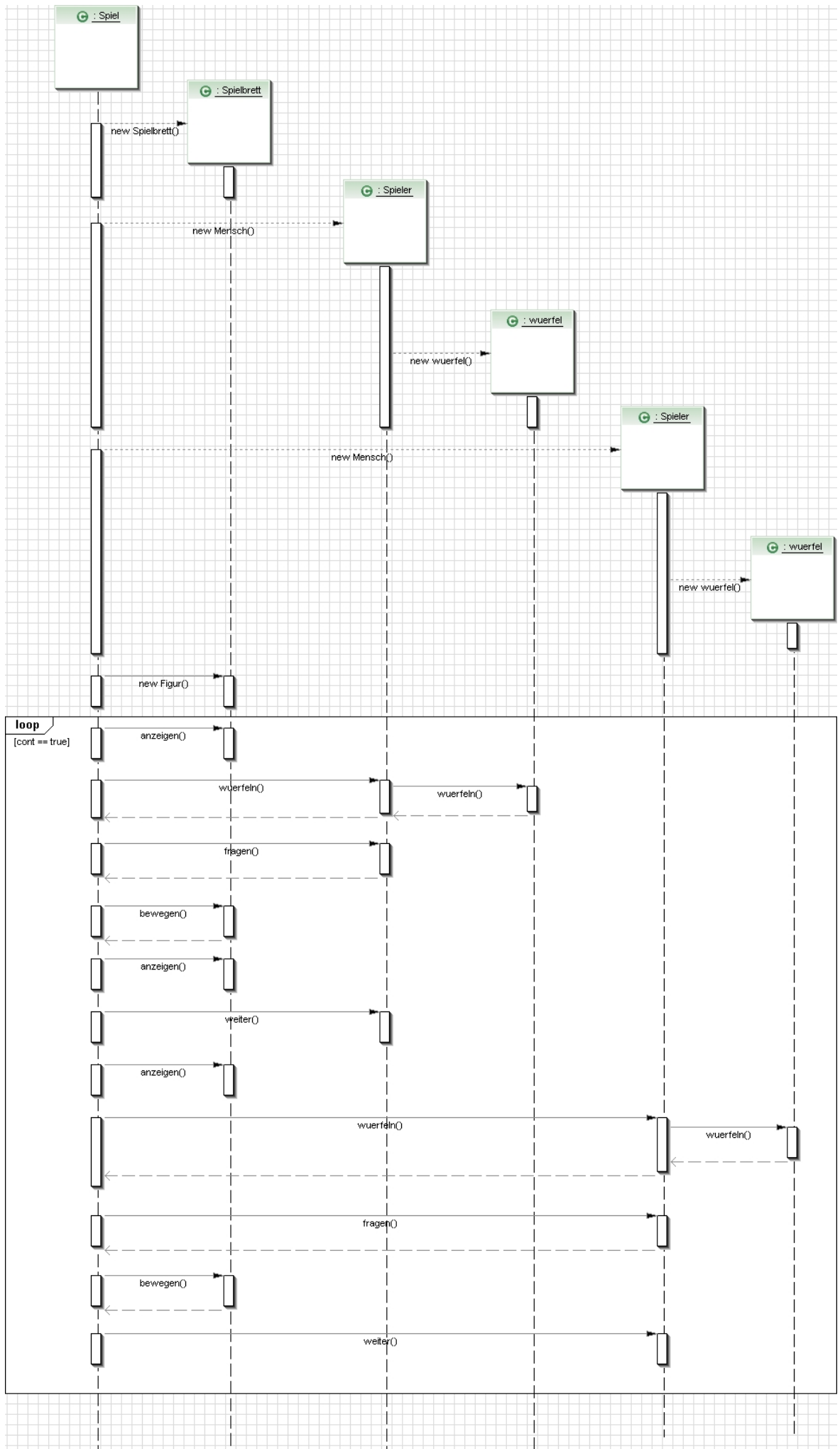
Das Spiel ist nur von 2 menschlichen Spielern spielbar. Das Spielfeld ist ein lineares Feld bestehend aus 40 Feldern, wobei beide Spieler beim gleichen Feld anfangen. Auf eine Hilfedatei oder eine Regelbeschreibung wurde verzichtet. Am Anfang darf nur einmal gewürfelt werden. Kommt kein Sechser, ist der nächste Spieler dran.

Das gesamte Spiel wurde sehr vereinfacht, jedoch wurde darauf Wert gelegt, dass es ausbaufähig bleibt. Zum Beispiel wurde eine abstrakte Klasse „Spieler“ verwendet, um eventuell später Computergegner hinzuzufügen.

## Class Diagramm überarbeitet:



## Sequenz-Diagramm überarbeitet:



# Implementierung:

## **Beschreibung der Klassen:**

Das Spiel besteht aus folgenden Klassen:

1. Figur
2. Mensch
3. Spiel
4. Spielbrett
5. Spieler
6. wuerfel

### **Klasse Figur:**

Besteht aus den Variablen `figurID (int)` und `figurSpielerName(String)`. Somit wird jede Figur eindeutig einem Spieler gegeben. Außerdem hat sie eine `toString` Methode, damit sie richtig angezeigt werden kann.

### **Klasse Mensch:**

Ist eine Kind-Klasse der Klasse `Spieler`. Besitzt eine `static Variable int`, damit das Spiel weiß, wie viele Spieler es gibt. Spezifiziert die Methoden `weiter()` und `fragen()` der Superklasse. Besitzt eine `readString()` Methode, damit der Name eingelesen werden kann.

### **Klasse Spiel:**

Besitzt die `main`-Methode. In dieser erzeugt sie das `Spielbrett` und die `Spieler`. Dann leitet sie das Spiel, indem sie Anfragen an die `Spieler` verschickt. Mittels einer `do`-Schleife wird läuft das Spiel. Wenn die Klasse 4 Figuren von einem Spieler im Zielfeld findet, wird die `do`-Schleife beendet.

### **Klasse Spielbrett:**

Besitzt drei Arrays: `spielbrettArray: Figur[]`, `startBoxArray: Figur[][]`, `zielBoxArray: Figur[]`. Der Konstruktor erstellt auf dem `spielbrettArray` Platz für 40 Figuren, auf dem `startBoxArray` 2 mal 4 Figuren für die Spieler und im `zielBoxArray` Platz für 8 Figuren.

Die Methode `anzeigen()` zeigt dort, wo keine Figur steht, einen Punkt an. Dort, wo eine Figur steht, ruft sie die `toString`-Methode dieser Figur auf.

Der Methode `bewegen()` werden der `figurSpielerName(String)`, die `figurID(int)` und die gewürfelte `zahl(int)` übergeben.

Sie sucht zunächst die Figur im `spielbrettArray`. Ist die `zahl == 6`, wird die Figur der Variable `inDerHand` übergeben.

Dann wird das Spielfeld gesucht. Auch hier wird die gefundene Figur an `inDerHand` übergeben.

Von `inDerHand` wird die Figur dann jeweils auf dem `spielbrettArray` verschoben.

### **Klasse Spieler:**

Die Klasse Spieler wurde als abstract definiert, damit eventuell später noch Computergegner hinzugefügt werden könnten.

Sie enthält die Variable name(String) und der Konstruktor erstellt einen Würfel. Außerdem enthält sie die abstrakten Methoden weiter() und fragen().

### **Klasse wuerfel:**

Enthält die Methode wuerfeln(), die eine Zufallszahl von 1-6 retourniert (int).

### **Installation und Wartung:**

Da es sich vorerst nur um einen Prototypen handelt, muss mit Fehlern gerechnet werden.

Eventuell wird das Projekt noch zum vollständigen Spiel ausgebaut.

Zum Ausführen wird JRE 1.5 oder höher benötigt. Leider musste auf eine GUI verzichtet werden, das Spiel kann jedoch textbasiert gespielt werden.