BERUFLICHE SCHULE WIRTSCHAFT . KIEL

LUDWIG ERHARD SCHULE

DER RAVENSBERG und Abendgymnasium

Software Challenge 2010

Hase und Igel

Unter der Leitung von Herrn Dombergen und Raphael Randschau

Abgabedatum: 19.01.2010

Gruppe 2: Frederik N., Frederic T., Sean P.,
Jessica B., Sönke E. und Tristan R.

Schule: Regionales Bildungszentrum Kiel
Standort der Ravensberg

Vorwort

Die Ausarbeitung der Gruppe 2 besteht aus:

- einem Wochenprotokoll
- einem Lastenheft (bei der Ausarbeitung wurde folgende Internetseite als Grundstruktur verwendet: http://www.stefan-baur.de)
- einem Pflichtenheft
- einem Quellcode des Spielclient

Protokoll Hase und Igel Spielclient

Das Projekt beginnt am 08.01.2010 und endet mit einer Präsentation am 19.01.2010, die die Ergebnisse der Projektphase zeigen und die das Programmieren den Schülern aus der Klasse 11a näher bringen soll, die unsere Arbeit als Zuschauer begutachten werden.

Zunächst fanden wir uns in den Raum 520 zusammen, wo wir auch den größten Teil der Projektzeit verbringen werden. Unser Projektunterstützer, Rafael, und Herr Dombergen, unser Klassenlehrer und Betreuer der Woche, klärten uns Schüler über den Verlauf der Projektwoche auf.

Mit Rafael, der Informatik an der Universität in Kiel studiert, hatten wir zuvor ungefähr ein halbes Jahr das Programmieren gelernt, sodass wir schon in dem Besitz der Grundkenntnisse waren. Zu Beginn arbeiteten wir mir Java Kara, um dann über Eclipse das Spiel Hase und Igel zu programmieren.

Das Projekt sollte möglichst aus 4 Gruppen bestehen, damit es zu einer gerechten Arbeitsverteilung kommen kann. Das Minimalziel jeder Gruppe sollte es sein am Ende der Woche einen lauffähigen Client vorweisen zu können. Zusätzlich bekamen wir Informationen über die 13. Klasse des letzten Schuljahres, um das Wissen zum kommenden Verlauf erweitern zu können, da sie auch an der Software Challenge teilgenommen hatten.

Es folgten Informationen über die Raumverfügbarkeit in der Projektphase. Der EDV-Raum wird nicht immer für uns zur Verfügung stehen und daher müssen wir auch flexible Aufgaben für die übrig bleibende Zeit finden, die wir auch ohne einen Computer bewältigen können.

Im ganzen Projekt fand kein Unterricht statt, ausschließlich vom Sport- und Religionsunterricht.

Ein allgemeines Treffen fand täglich immer von 11.15 – 12.00 Uhr im Sammelraum und EDV-Raum mit unserem Klassenlehrer statt, um die Fortschritte zu erkennen und bei Fragen Hilfestellungen zu geben

Protokoll Hase und Igel Spielclient

Rafael stand nicht nur täglich bei Problemfragen in der Unterrichtszeit für uns zur Verfügung, sondern auch privat. Aus diesem Grund erhielten wir Rafaels Kontaktdaten.

E-Mail Adresse: RRA@INFORMATIK.uni-kiel.de

Anschrift der Firma: Schauenburgerstr 219 bei raddatz/bielefeld.

Nach einigen Diskussionsphasen am ersten Tag, weil sich in jeder Gruppe qualifizierte Computerfachlaute befinden sollten, stellten die Gruppen sich wie folgt zusammen:

Gruppe 1	**Gruppe 2**	Gruppe 3	Gruppe 4
Johannes	Frederic T.	Hans Jochen	Tjark
Adrian	Frederik N.	Andre	Gianni
Ender	Sönke	Thore	Svenja
David	Jessica	Todor	Stella
Andreas	Tristan	Yassa	Alex
Phillip	Sean	Christoph	Nils

Der fertig gestellte und immer wieder verbesserte Client muss spätestens Ende Januar bei der Software Challenge abgegeben werden.

Die erste Tätigkeit in unserer Projektphase war es, dass wir gruppenintern die Aufgabenverteilung vorgenommen haben, sodass jede Person der jeweiligen Gruppe eine Grundaufgabe hat, mit der er sich beschäftigt und die jeder auch zu Hause erweitern kann.

Protokoll Hase und Igel Spielclient

- Mit der Programmierung beschäftigen sich hauptsächlich Sean Parker und Tristan Reimer.
- Das Lasten und Pflichtheft übernimmt Frederic Trautmann.
- Frederik Nitsch beschäftigt sich überwiegend mit der schriftlichen Erklärung unserer Strategie.
- Sönke Ehlert sowie Jessica Behrens schreiben einen Wochenbericht über das Projekt und planen die Präsentation.

Jede Person unserer Gruppe wird aber auch eine prüfende Funktion, für die Aufgaben der Anderen, übernehmen.
Am Anfang und am Ende jeden Tages erfolgte eine Gruppenbesprechung und die gegenseitige Hilfestellung bei Problemen. Ebenfalls versuchten wir uns auch in den verschiedenen Unterpunkten zu ergänzen und dieses führte dann auch zu dem Austausch unserer privaten Kontaktdaten. Der Austausch der Daten war in Hinsicht auf das Projekt wichtig, da wir immer erreichbar sein müssen und damit wir gegenseitig Informationen austauschen können.

In der Mitte der Woche wählten wir aus jeder der 4 Gruppen eine Person aus, die die gruppeninternen Vorschläge zur Präsentation in einem Stuhlkreis nennt. Dieses sollte dazu beitragen, dass mehrere Ideen zusammengefasst und aufgelistet werden können, sodass jede Gruppe, durch 2-3 ausgewählte Personen, unser Projekt am 19.01 präsentieren können.

Protokoll Hase und Igel Spielclient

Die Präsentation besteht aus folgenden Unterpunkten mit ihren dazugehörigen Gruppen und Themen:

Präsentationspunkte	**Gruppe**	**Details**
Einleitung	3	-Aufgabenstellung erläutern -Vorbereitungsmöglichkeiten -Was ist ein Client
Spielregeln	4	-Regeln des Spiels -Erklärung der Spielfelder
Spielstrategie	alle	-Jede Gruppe stellt ihre eigene Spielstrategie vor
Umsetzung	1	-Erklärung der Programmierung mit Eclipse
Spiel	**2**	-Die 4 fertigen Client spielen gegeneinander -Gewinnerclient gegen einen Schüler -Gewinnerclient gegen Dombergen

Die Präsentation wurde in unserem Klassenraum präsentiert und von einer Power Point Präsentation begleitet.

Protokoll Hase und Igel Spielclient

In der restlichen Zeit arbeitete jeder aus unserer Gruppe intensiv an seinen Aufgaben weiter oder wir erarbeiteten ein Teil der noch nicht behandelten Aufgaben der Anderen aus der Gruppe.
Gegen Ende des Projekts wurde die Mappe von Jessica Behrens auf Rechtschreibfehler kontrolliert und die verschiedenen Texte werden formgleich von Frederic Trautmann anpasst, damit die Mappe eine Struktur erhält. Die Präsentation vor der Klasse 11a werden Sean Parker, Tristan Reimer und Frederik Nitsch übernehmen, da sie sich mit der Strategie und dem Spiel am meisten auseinander gesetzt haben.

Die Bewertungskriterien für die Endnote des Projekts setzten sich auch folgenden Punkten zusammen:

- ½ Client inklusive Lasten- und Pflichtenheft
- ¼ Wochenbericht
- ¼ Präsentation

Das Lasten und Pflichtenheft wird von dem Wochenbericht ergänzt. Sie glichen sich in einigen Punkten, wurden jedoch getrennt von einander behandelt. Das Lasten- und Pflichtenheft bezieht sich auf den Softwareeinsatz, die Funktion des Clients, den Spielverlauf, die Anforderungen an das Programm und die vorliegenden Hilfsmittel. Im Gegensatz dazu spiegelt der Wochenbericht die Einteilung und die Funktionalität der Aufgabenbearbeitung sowie den Ablauf der Woche wider. Die Präsentation wird dieses Mal im Hintergrund gestellt, da die Hauptaufgabe ist einen Client zu programmieren.

Protokoll Hase und Igel Spielclient

Reflektion

Schwierigkeiten traten in Hinsicht auf die Überlieferung der Texte von zu Hause auf, da diese oft in einer neueren Version von Word hergestellt wurden und somit konnte nicht jeder aus unserer Gruppe ohne Probleme auf die Dateien zugreifen.
Ebenfalls gingen in den einzelnen Gruppen die Klassengespräche unter, weil wir uns hauptsächlich auf die Aufgaben spezialisiert haben den Client fertig zu stellen und eine Mappe anzufertigen, anstatt die Präsentation vorzubereiten, wo diese Klassengespräche notwendig gewesen wären. Dieses führte dazu, dass die Ausarbeitung der Präsentation sehr spät begann und schleppend voranging.

Jedoch gab es von Anfang an große Fortschritte bei der Bearbeitung vom Lasten- und Pflichtenheft, sowie beim Wochenbericht und bei der Programmierung des Clients.
In der ganzen Projektphase herrschte eine gute Gruppenfunktionalität. Jeder bewältigte seine Aufgaben vollständig und erhielt sogar auch außerhalb der eigenen Gruppe Hilfe von den Mitschülern und Lehrern. Oft setzten wir uns auch selbstständig mit Mitschülern aus den anderen Gruppen zusammen, um zum Beispiel zu besprechen, was ein Lasten- und Pflichtenheft enthalten soll.
Die Zusammenarbeit im Allgemeinen verlief oft reibungslos. Es gab nur wenige Meinungsverschiedenheiten und wenn doch welche auftraten wurden sie mit der ganzen Gruppe besprochen, sodass jeder seine Meinung vertreten konnte und wir letztendlich zu einer gemeinsamen Lösung kamen.

Tipps: -Kontrolliertes Vorgehen und jede Aufgabenstellung rechtzeitig beachten.
-Klassengespräche nicht vernachlässigen, da es auch ein Klassenprojekt sein soll.
-Einen Wochenplan zu erstellen, indem Zwischenziele festgelegt werden.

Lastenheft

Hase und Igel Spielclient

Projekt:	Client für Brettspiel Hase & Igel
Autor:	Gruppe 2
Schule:	Regionales Bildungszentrum Kiel
	Standort der Ravensberg
Letzte Änderung:	17. Februar 2009

Inhaltsverzeichnis

1 Zielbestimmungen

Durch die Zielbestimmungen soll klargestellt werden unter welchen Bedingungen der Client erstellt werden soll.

Ein Spielclient soll für das Brettspiel Hase und Igel erstellt werden. Er soll gegen andere Spielclients über eine Programmoberfläche spielen. Die Programmfläche wird von der Christian-Albrechts-Universität zu Kiel gestellt. Die erstellten Spielclients müssen am 28. Februar abgegeben werden und treten am 15. März bis zum 15. Mai 2010 gegen andere Spielclients von unterschiedlichen Schulen an. Die zweite Phase findet am 4. Juni 2010 in Kiel im Sophienhof statt. Die ersten acht Mannschaften sind dort vertreten.
Das Ziel der Klasse ist es am Ende der Projektphase zwei lauffähige Clients abzugeben. Dazu werden in der Klasse vier Clients erstellt, die jeweils über eine andere Taktik bzw. Strategie verfügen sollen. Der Spielclient soll unter den modifizierten Regeln von dem Spielfeld der Version 2008 vom Ravensburger Spieleverlag erstellt werden. Es wurde wie folgt modifiziert:

- Die Igel- und Salatfelder werden so angeordnet wie im Original. Die Felder zwischen Igelfelder mit Ausnahme der Salatfelder werden zufällig angeordnet.
- Die Zahlenfelder 3 + 4 werden durch Möhrenfelder ersetzt
- Das Zahlenfeld 1/5/6 wird durch das Zahlenfeld 1 ersetzt.
- Das Ziel muss in spätestens 30 Züge erreicht werden

Unter diesen Bedingungen sieht die Benutzeroberfläche wie folgt aus:

[1]

Wie auch bei dem Original ist der Sieger, der Client, der als erstes die gesamte Wegstrecke von 64 Feldern zurücklegt, keinen Kohl besitzt und unter 10 Karotten ins Ziel kommt. Das Spiel wird jedoch vorzeitig nach Ablauf von 30 Runden abgebrochen und es gewinnt dann der Spieler, der am weitesten gekommen ist.

Beim Erstellen des Spielclients müssen erweiterte Module integrierbar sein, sodass der Client in der Wettkampfphase ergänzt und eingereicht werden kann. Die Clients können zwischen den Spieltagen erneut aktualisiert werden. Aus diesem Grund ist das Klassenziel insgesamt vier lauffähige Clients zu erstellen.

[1] http://www.informatik.uni-kiel.de/fileadmin/zentrale_bereiche/software_challenge/2010/download/spielregeln.pdf, Zugriff am 12.1.2010 / 10:42

2 Produkteinsatz (Softwareeinsatz)

Mit dem Spielclient soll im Rahmen eines Wettkampfes der Umgang mit der Java oder Delphi Programmierung spielend erlernt werden. Der Spielclient präsentiert die Projektarbeit und vertritt jeweils die Schule, die den Client erstellt hat.
Weiterhin kann der Spielclient als Computergegner eingesetzt werden. Besonders für Einzelpersonen die in ihrem Umfeld kaum Gelegenheit haben mit Gleichgesinnten dieses Brettspiel zu spielen, kann dieser Client als Spielgegner dienen.

Der Client soll im Laufe eines Schulprojektes erstellt werden. Der Spielclient wird nur innerhalb von Deutschland eingesetzt und erfordert daher nur Deutsch als Anmerkungen in den Programmierungen. Dieses soll nur die eigenen Arbeiten erleichtern.

3 Produktfunktionen

Produktfunktionen sollen angeben welche Fähigkeiten der Client besitzen soll und wie der Benutzer den Client anwenden bzw. starten kann.

3.1 Benutzerfunktionen

Der Benutzer kann den Client erst benutzen, wenn er sich die Software für Benutzeroberfläche bzw. den Offline-Server (mit grafischer Oberfläche) auf der Internetseite: http://www.informatik.uni-kiel.de kostenlos heruntergeladen hat. Der Benutzer kann den Spielclient durch die Installierung eines Offline-Servers (mit grafischer Oberfläche) ohne Internet-Verbindung oder Anmeldung ausführen.
Er kann jederzeit den Client und die Benutzeroberfläche von dem Computer entfernen.

3.2 Spiele Funktion des Client

Der Client soll über eine künstliche Intelligenz verfügen. Dabei muss er eine Spielstrategie ausführen bei der die modifizierten (siehe Seite 11) Spielregeln berücksichtigt werden. Die Zugzeit ist für einen Zug auf 2 Sekunden begrenzt. Dafür gilt für die Rechenzeit die im Institut verwendete Hardware als Referenz.

Der Wert von typ ist entweder

- MOVE
- EAT
- TAKE_OR_DROP_CARROTZ
- FALL-BACK
- PLAY-CARD
- SKIP

Der von card ist entweder

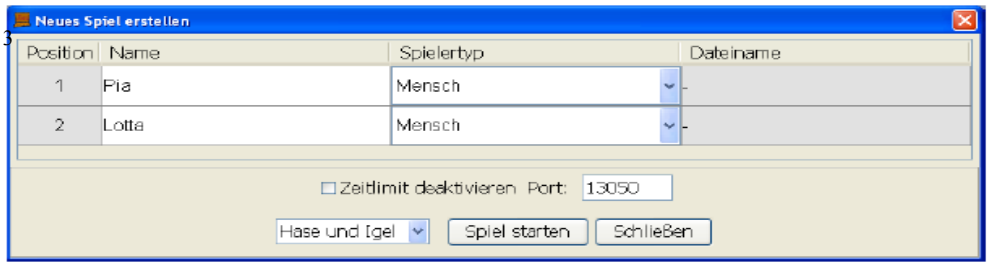
- TAKE_OR_DROP_CARROTS
- EAT_SALAD
- HURRY_AHEAD
- TIE

Die Parameter sollen sinnvoll im Client kombiniert werden.

3.3 Spieleverlauf[2]

Beim Erstellen eines neuen Spieles müssen in diesem Fenster die Spieler / Clients ausgewählt werden, die an dem Spiel teilnehmen sollen.

3



Um den Client auszuführen zu können muss als Spielertyp Computerspieler (manuell) ausgewählt werden und beim Dateiname der dazugehörige Client ausgewählt werden.

Um das Spiel starten zu können muss das Programm mit dem Server verbunden werden. Dabei wird der Port 13050 standardmäßig eingestellt und verwendet. Durch die Änderungen vom 3.November wurde eine automatische Portsuche mit eingefügt, die sich automatisch mit einem freien Port verbindet (falls z.B. der vorhandene belegt ist). Sobald das Spielfeld geöffnet wird muss zum Start das Startsymbol auf dem Steuerelement am unteren Rand gedrückt werden.

4

[2] Die Benutzung bzw. der Spielablauf ist auf folgender Internetseite ausführlich nachgelesen werden:

http://www.informatik.uni-kiel.de/fileadmin/zentrale_bereiche/software_challenge/2010/download/server-dokumentation.pdf, Zugriff am 11.1.2010 um 13:00 Uhr

[3] http://www.informatik.uni-kiel.de/fileadmin/zentrale_bereiche/software_challenge/2010/download/spielregeln.pdf, Zugriff am 12.11.2010 um 10:46 Uhr

Der Client soll für den Wettkampf auf 3 Kommando Zeilen Parameter reagieren. Diese sind:

Port, horse und ein Reservierungszeichen für das Spiel

[4] http://www.informatik.uni-kiel.de/fileadmin/zentrale_bereiche/software_challenge/2010/download/spielregeln.pdf Zugriff am 12.11.2010 um 10:47 Uhr

4 Programmgebundene Anforderungen

Die programmgebundenen Anforderungen geben an, welche Voraussetzungen die Software benötigt, um den Client anwenden zu können.

Der Benutzer benötigt Java 1.6[5] um die Software auszuführen. Java ist eine plattformunabhängige Software. Es ist eine objektorientierte Programmiersprache Die Software ist für die folgenden Betriebssysteme verfügbar:

- Windows (Windows 7 / XP / Vista / 2000 / 2003 / 2008)
- Solaris (Solaris x86 / Solaris x64)
- Linux (z.B. Ubuntu / Novell Suse)

Andere Hersteller lassen Java für ihre Plattform zertifizieren, wie zum Beispiel:

- Apple (OS X Apple Computer stellt seine eigene Version von Java zur Verfügung)

Für die Java Programmierung wird Eclipse in den Gruppen als Software für die Programmierung verwendet, da es ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedener Arten ist. Eclipse ist plattformunabhängig da es auf Java basiert.

[5] Die Java Software ist unter folgendem Link runterzuladen:
http://java.sun.com/javase/downloads/index.jsp, Zugriff am 11.1.2010 um 10:50 Uhr

5 Vorliegende Hilfsmittel

Vorliegende Hilfsmittel sollen klarstellen welche vorhandene Software / Hilfsmittel den Gruppen zur Verfügung stehen.

Für die Programmierung steht den einzelnen Gruppen das kostenlose Programm Eclipse[6] zur Verfügung. Auf Grund der relativ hohen Systemanforderungen steht den einzelnen Gruppen der Raum 520 in der Schule Regionales Bildungszentrum Kiel Standort Ravensberg zur Verfügung.
Die Grundlagen für die Java Programmierung wurde in dem ersten Halbjahr der 12ten Klasse durch die Unterstützung einer wissenschaftlichen Hilfskraft erlernt. Die Hilfskraft (Raphael Randschau) wird in der Projektphase weiterhin als Unterstützung dienen. Zudem steht den einzelnen Gruppen ein vorgefertigter Simple Client[7] zur Verfügung. Dieser verfügt über keine nennenswerte, künstliche Intelligenz, aber gibt den Gruppen eine Grundlage für die Programmierung und Ausarbeitung für ihre Taktik des Clients.

[6] Das Programm kann unter folgendem Link heruntergeladen werden:
www.eclipse.org, Zugriff 12.1.2010

[7] Der simple Client kann unter folgendem Link heruntergeladen werde:
http://www.informatik.uni-kiel.de/software-challenge/2010/download/

6 Qualitätsanforderungen

Die Qualitätsanforderungen sollen verdeutlichen auf welche Punkte besonders geachtet werden muss.

Auf die Zuverlässigkeit, Lauffähigkeit und die Effizienz[8] wird größten Wert gelegt. An zweiter Stelle stehen die Robustheit sowie auch die Übersichtlichkeit des Quellentextes vom Client.
Die Kommunikation zwischen Server und des Clients soll möglichst schnell erfolgen. Der Client soll verschiedene Strategien während des Spieles anwenden können. Dabei ist wichtig, dass der Client möglichst weit kommt und möglichst wenige Karotten ins Ziel bringt, da dies im Wettkampf zusätzlich mit in die Wertung fließt.

[8] Wettkampfphase bzw. die Ermittlung des Sieger Clients wird auf folgender Seite noch einmal genauer beschrieben:

http://www.informatik.uni-kiel.de/software-challenge/2010/wettkampf/, Zugriff am 12.1.2010 um 11:00 Uhr

7 Ergänzungen

Ergänzungen geben an welche speziellen Anforderungen der Client im Rahmen des Projektes haben muss.

7.1 Realisierung

Der Client soll die Bibliothek XSTREAM einsetzen, um das Protokoll in Java verarbeiten zu können. XStream parst das XML voll automatisch und gibt die erhaltene Nachricht komfortabel als Objektinstanz zurück. Die Kommunikation zwischen Client und Server werden daher über ein XML Dokument gelöst.

7. 2 Ausblick auf nächste Projekte im Rahmen der Software-Challenge

Die Verwendung des Client ist nur für das Projekt Software-Challenge 2010 vorgesehen. Im Rückblick auf vorherige Projekte können keine spezifischen Befehlssätze übernommen werden. Daher gilt für die Erstellung des Clients:
Der allgemeine Befehlssatz ändert sich mit der nächsten Software-Challenge nicht. Die Spiel-spezifischen Befehlssätze hingegen werden für jedes weitere Projekt geändert.[9]

[9] Alle allgemeinen Befehlsätze sind unter folgendem Link abrufbar:

http://www.informatik.uni-kiel.de/software-challenge/2010/dokumentation/kommunikation/, Zugriff am 10.1.2010 um 13:00

BERUFLICHE SCHULE WIRTSCHAFT . KIEL

LUDWIG ERHARD SCHULE

DER RAVENSBERG
und Abendgymnasium

Pflichtenheft

Hase und Igel Spielclient

Projekt:	Client für Brettspiel Hase & Igel
Autor:	Gruppe 2
Schule:	Regionales Bildungszentrum Kiel
	Standort der Ravensberg
Letzte Änderung:	17. Februar 2009

Inhaltsverzeichnis

5 Zielbestimmungen

Zielbestimmungen definieren welche Kriterien für den Client erforderlich sind. Dabei wird unterschieden zwischen Musskriterien, Wunschkriterien und Abgrenzungskriterien.

Die Gruppe 2 stellt einen Spielclient für das Brettspiel Hase und Igel, der das Spielen gegen andere Clients und Personen ermöglicht.

1.1 Musskriterien

Das Programm der Client, welcher geschrieben wird hat primär das Ziel, unter Berücksichtigung der gestellten Spielregeln der CAU dem Auftraggeber, in dem Spiel „Hase und Igel“ das Spielziel und somit das Ende des Spiels zu erreichen.

1.2 Wunschkriterien

Sekundär soll das von uns geschriebene Programm möglichst den Gegenspieler im Rahmen der Regeln daran hindern sein Ziel zu erreichen, welches dasselbe wie unseres ist. Dabei sollte das Voranschreiten des Kontrahenten verhindert werden. Wenn aber Gefahr besteht, dass unser Client sein Primärziel nicht erreicht, soll er vorerst auch nur noch dieses verfolgen.

1.3 Abgrenzungskriterien

Das Programm kann wie folgt abgegrenzt werden:

Das Programm muss nur mit dem vorgegebenen Server des Auftraggebers kompatibel sein. Für das geschriebene Programm ist es ausreichend, wenn es nur mit dem vorgegebenem Spielfeld lauffähig ist, d.h. es ist davon auszugehen, dass einige Elemente im Spielfeld unbeweglich sind und nur eine bestimmte Anzahl von Elementen auf dem Spielfeld zufällig generiert werden.

6 Produkteinsatz

Produkteinsatz definiert zu welchem Zweck, mit welchen Qualifikationen der Client bedient werden kann und die Betriebszeit des Clients.

2.1 Anwendungsbereiche

Der Client soll für den Wettkampf bei der Software-Challenge eingesetzt werden. Außerdem können Einzelpersonen diesen Client als Spielgegner verwenden. Dieser Client kann die Fertigkeiten des Spielers verbessern, da der Client über verschiedene Taktiken verfügt.

2.2 Zielgruppen

Für Einzelpersonen, die kurz zur Ablenkung zum Beispiel in der Mittagspause gerne das Brettspiel spielen wollen, aber keine Gleichgesinnten haben um es zu spielen, kann der Client als Gegner eingestellt werden. Es werden für die Benutzung nur Basiskenntnisse im Umgang von Computerprogrammen vorausgesetzt. Außerdem benötigt der Spieler die modifizierten Spielregeln für das modifizierte Spiel Hase und Igel. Im Vorfeld steht jedoch die Software-Challange 2010, daher muss der Client auf 3 Zeilen Parameter reagieren damit er im Wettkampf gegen andere Clients antreten kann. Daher wurde in public class folgendes geschrieben:

- @param ip = hier wird die IP bestimmt mit welcher sich der Client verbinden soll
- @param port = hier wird der Port bestimmt mit welcher sich der Client verbinden soll
- Param spielreservierung = falls eine Spielreservierung beim Server für den client vorligt

2.3 Betriebsbedingungen

Der Client soll sich bezüglich der Betriebsbedingung nicht wesentlich von dem simplen Client unterscheiden.

- Betriebsdauer: täglich 24 Stunden
- Wartungsfrei
- Falls nötig, muss der Client bei auftretenden Fehlern vom Administrator behoben werden.

7 Produktumgebung

Die Produktumgebung definiert welche Software und Hardware für die Erstellung des Client benötigt wird und welche Hardware, Software und Orgware für die Entwicklung des Clients in der Gruppe 2 verwendet wird. Es wird darauf geachtet das möglichst viele Entwicklungstools kostenlos (Freeware) sind.

3.1 Software

Es wird ein Java fähiges Betriebssystem benötigt. In der Gruppe 2 wurde Windows Vista, Windows 7 und Ubuntu (Linux Kernel) verwendet. Für die Software Java wird bevorzugt sun java 1.6 benötigt. Für die Entwicklung des Clients wird eine Java fähige Entwicklungsumgebung benötigt. In der Projektphase wird daher eclipse für die Programmierung verwendet

3.2 Hardware

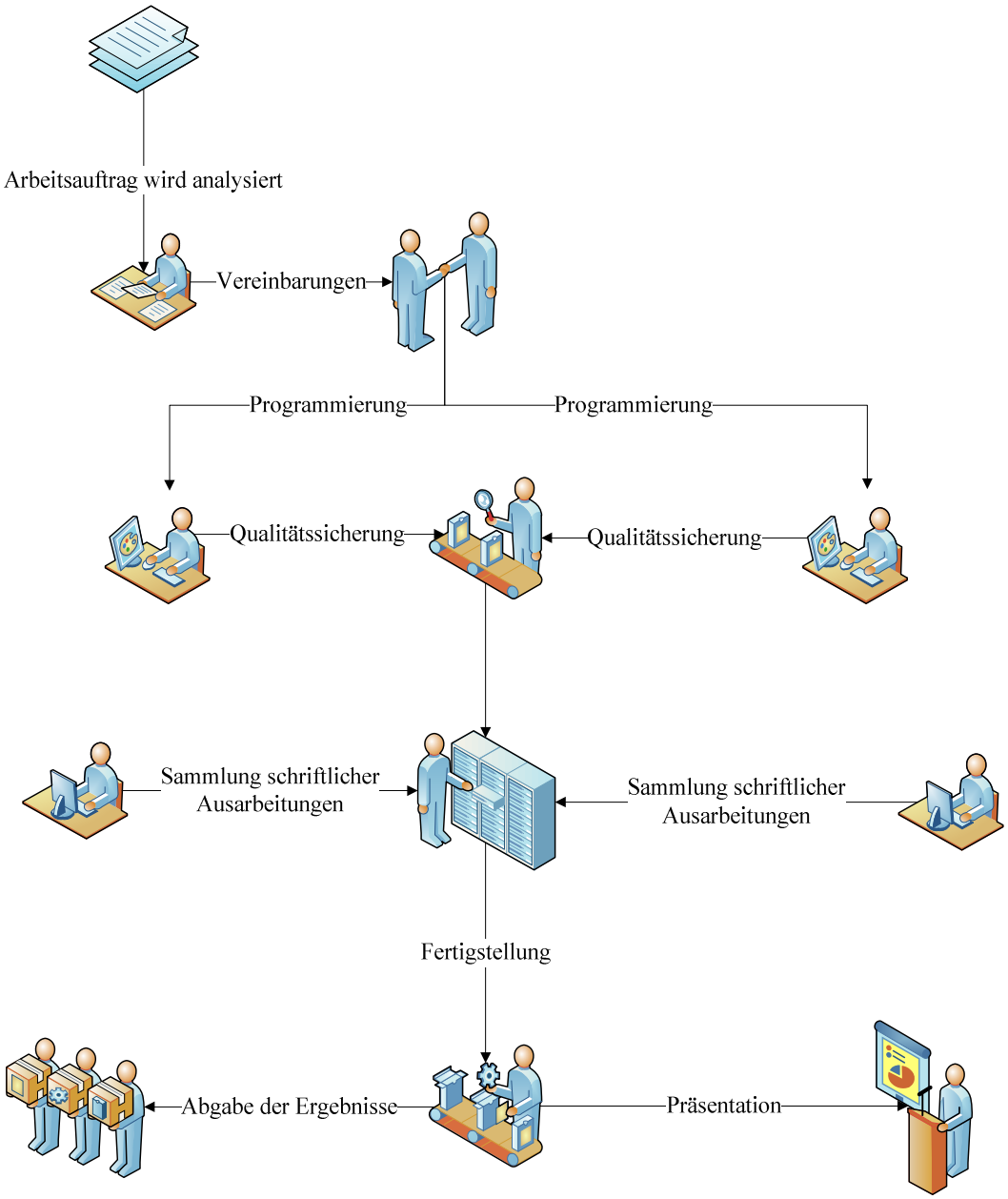
Hardware wird nicht benötigt, da der Client auf den Servern der Christian-Albrechts-Univerität zu Kiel im Rahmen des Wettkampfes laufen wird. Lediglich für die Entwicklung und die Ausführung des Clients wird ein Javafähiger Rechner benötigt und eine Internetverbindung um den Client auf die Server hoch zu laden.

3.3 Orgware

Zur Kommunikation für die Teilnehmer des Projektes wird ein Emailverteilersystem eingesetzt, sowie ICQ. Damit alle Teammitglieder auf ihrem Computer immer den aktuellsten Stand des Projektes haben.

3.4 Entwicklungsprozess

Die Graphik soll den Ablauf der Erstellung des Clients im Rahmen des Projektes ergänzend darstellen.

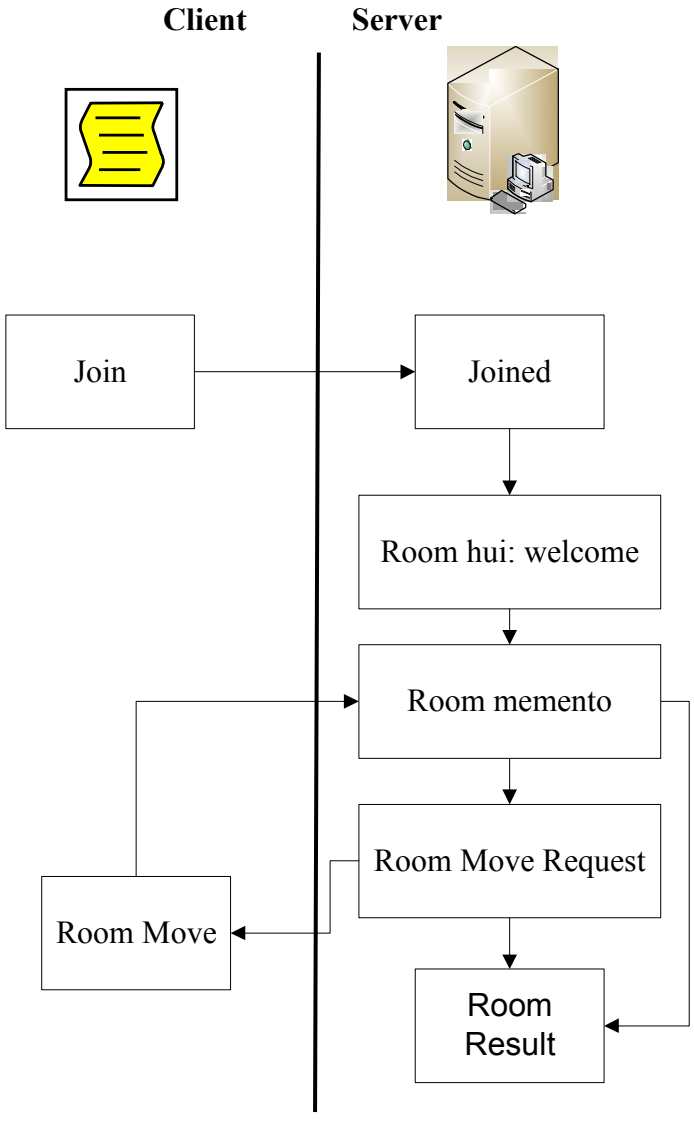


8 Produktfunktionen

Produktfunktionen geben den Benutzern eine Übersicht was der Client leistet.

4.1 Client und Server

Der Client hat die Fähigkeit zwischen den Servern zu kommunizieren. Ein Spielverlauf wird hier grafisch gezeigt. Als Grundlage diente die Internetseite: schule.reemo.de/pub/swchallenge2010/Protokolldokumentation.pdf



Die ersten Nachrichten des Clients an den Server werden für die Zuordnung eines Raumes benötigt. Dabei kann der Client zwei Nachrichten schicken. In den Wettkampfphasen muss ein JoinPrepared gesendet werden, da er mit einer Reservierungsnummer versehen wird. Ein Join muss dann gesendet werden wenn kein Reservierungscode vorhanden ist. Der Server antwortet dem Client auf beide Nachrichten mit einem joined. In Room hui-welcome wird dem Client seine Farbe im Spiel übermittelt. Die Farbe des Client ist dabei entweder rot oder blau. Room memento gibt den aktuellen Spielestand an. Diese Nachricht wird immer vor und nach einem Zug geschickt. Danach wird ein Move Request an den Client gesendet. Dieser antwortet mit einem hui:move d.h. er macht einen Spielzug. Zwischen diesen Schritten wird ein Result gesendet , welcher das Spielergebnis angibt.[10]

4.2 Strategien des Clients

Zuerst haben wir uns überlegt wie man die Taktik des Clients am besten aufbauen kann.

Wir haben uns entschlossen, dass Spielfeld in 3 Teile einzuteilen, um uns das Programmieren zu vereinfachen. Unsere Idee in der Taktik ist, dass wir bis Feld 22 möglichst viele Karotten sparen und somit größere Sprünge vermeiden. Wir haben uns außerdem überlegt, dass wir möglichst bis Feld 22 vier Salatköpfe gefressen haben.

Der erste Spielfeldabschnitt ist von Feld 0 bis Feld 10.

Unsere Taktik besagt für diesen Abschnitt, dass wenn der Gegner anfängt, wir nach Möglichkeit auf das Platzfeld 5 ziehen. Wir ziehen nur auf Platzfeld 5, wenn dieses ein „Zweites Platzfeld“ ist. Wenn es kein „Zweites Platzfeld“ ist, dann soll unser Spieler überprüfen ob zwischen Feld 0 und Feld 10 ein „Zweites Platzfeld“ liegt.

[10] http://schule.reemo.de/pub/swchallenge2010/Protokolldokumentation.pdf, Zugriff am 16.1.2010, Zugriff am 15.1.2010 um 14:49 Uhr

Wenn eins vorhanden ist soll unser Spieler auf dieses Feld ziehen, wenn keins vorhanden ist soll er auf ein Hasenjokerfeld gehen und den Joker „friss sofort einen Salat“ nehmen.
Wenn unser Spieler jedoch anfängt, ziehen wir direkt auf das Hasenjokerfeld und nehmen den Joker „friss sofort einen Salat“.
Nach diesem Szenario ziehen wir auf das Feld 10, aber nur wenn der gegnerische Spieler über die Feldnummer 10 ist. Wenn der Spieler jedoch noch nicht über diese Feldnummer ist, ziehen wir auf ein Hasenjokerfeld. Auf dieses Feld ziehen wir aber nur wenn der Hasenjoker „friss sofort einen Salat“ noch vorhanden ist. Wenn dieser nicht mehr vorhanden ist, dann ziehen wir auf das nächste Karottenfeld.
Wenn der Gegner jedoch die gleiche Taktik verfolgt uns vorlaufen zu lassen, ziehen wir, wenn wir noch nicht auf Feld 10 sind, auf ein Karottenfeld und setzen maximal 6 Züge aus und ziehen dann auf das Salatfeld und nach dem Fressen weiter.
Der zweite Spielfeldabschnitt ist von Feld 10 bis Feld 22.
Unser Ziel ist es erst einmal unter der Feldnummer 22, wenn noch vorhanden, den Hasenjoker „friss sofort einen Salat“ einzusetzen.
Wenn wir diesen vorher schon verwendet haben, ziehen wir automatisch auf Feld 17, außer dieses ist ein Hasenjokerfeld, dann ziehen wir auf Feld 18.
Nach diesem Zug ziehen wir auf das Salatfeld 22, wenn dieses jedoch besetzt ist ziehen wir auf Feld 20 oder 21 und bevorzugen dabei ein „Zweites Platzfeld“, aber nur wenn dieses auf Feld 20 liegt. Ist dort kein zweites Platzfeld gehen wir auf das Karottenfeld, auf Feld 20 oder 21 und setzen solange aus bis das Salatfeld frei ist.
Wenn der Salat noch nicht gefressen wurde, ziehen wir jetzt auf das Salatfeld.
Nach dem Fressen des Salates ziehen wir auf das Igelfeld zurück, wenn dieser von dem Gegner belegt ist, ziehen wir auf das nächste Hasenjokerfeld und nehmen den Joker „falle eine Position zurück“.
Danach ziehen wir wieder auf Feld 22 und fressen einen Salat, wenn dieses Feld besetzt ist, warten wir wieder mit unserer oben genannten Strategie.

Nach dem Fressen des vierten Salates kommt der dritte Spielfeldabschnitt, er ist von Feld 22 bis Feld 64.

Wenn unser Spieler erster ist, soll er möglichst auf die Felder 29,36,42,49,55. Wir ziehen aber nur auf diese Felder, wenn diese Felder keine Hasenjokerfelder sind. Wenn es Hasenjokerfelder sind, ziehen wir möglichst auf die Felder 29,36,49,55 (-1), wenn diese Karottenfelder sind, wenn nicht dann nehmen wir das Karottenfeld, welches am nächsten vor Feld 29,36,42,49,55 liegt. Das machen wir damit der Gegner den Hasenjoker „rücke eine Position vor“ nicht einsetzten kann, denn bei den oben genannten Feldern, ist ein Feld weiter also 29,36,42,48,55 +1 ein Igelfeld. Wenn wir zu wenige Karotten für die Sprünge auf die gewünschten Felder 29,36,42,48,55 haben, setzen wir so lange aus, bis wir genügend Karotten für die Sprünge haben. Auf Feld 42 gibt es eine Besonderheit, wir müssen auf dieses Feld ziehen, weil es sich hierbei um ein Salatfeld handelt und wir dort unseren letzten Salat fressen wollen.

Wenn wir nicht erster sind und zwischen dem Gegner und uns ein 2tes Platzfeld ist und wir genügend Karotten für den Sprung haben, ziehen wir auf das 2te Platzfeld.

Wenn keins vorhanden ist und wir zwischen 34 und 41 sind können wir auf Feld 42 ziehen und einen Salat fressen. Wenn wir zu wenige Karotten haben, ziehen wir erstmal auf ein Karottenfeld und setzen solange aus, bis das Feld 42 frei ist.

Wenn wir zwischen Feld 22 und 29 sind ziehen wir auf das weit möglichste Karottenfeld und es sollte möglichst vor 29 liegen.

Wenn wir zwischen Feld 30 und 34 sind, ziehen wir wieder auf das weit möglichste Karottenfeld möglichst vor Feld 37.

Wenn wir zwischen 42 und 48 sind ziehen wir auf ein Karottenfeld welches so weit wie möglich vor Feld 50 liegt.

Ab Feld 50 prüfen wir ob wir mit dem nächsten Zug unter 10 Karotten ins Ziel gelangen können.

Wenn wir ab Feld 50 zu viele Karotten fürs Ziel haben ziehen wir auf das weit möglichste Karottenfeld und geben solange Karotten ab bis wir unter 10 Karotten haben und ins Ziel ziehen können.
Wenn wir zu wenige Karotten haben ziehen wir wieder auf das weit möglichste Karottenfeld und nehmen solange Karotten auf, bis wir ins Ziel, mit weniger als 10 Karotten, ziehen können. Wenn kein Karottenfeld vor unserem Spieler ist ziehen wir zurück auf das Igelfeld. Wenn kein weiteres vor ihm ist, aber wir auf einem Karottenfeld stehen, nehmen wir solange Karotten auf bis wir mit unter 10 Karotten ins Ziel ziehen können.
Ab Feld 55 probieren wir direkt mit unter 10 Karotten ins Ziel, Feld 64, zu kommen. Wenn dies nicht möglich ist gehen wir immer auf das weit mögliche Karottenfeld und prüfen ob wir mit dem nächsten Zug unter 10 Karotten ins Ziel gelangen können.

9 Benutzungsoberfläche

Die Benutzungsoberfläche klärt wie der Client auf der Benutzeroberfläche arbeitet bzw. was sie anzeigt.

Die Benutzungsoberfläche GUI (Grafic user interface) wurde von der Christian-Albrechts-Universität zu Kiel gestellt. Die Entwickler sind Christian Wulf, Florian Fittkau, Marcel Jackwerth und Rapheal Randschau.

11

[11] Das Bild wurde durch eine Bildschirmkopie (Screenshot) erstellt.

Die Verwendung des Programms GUI wird nur zur Überprüfung des Clients benutzt und dient somit als Grundlage für die Fehlersuche in den einzelnen Gruppen.

10 Qualitätszielbestimmungen

In den Qualitätszielbestimmungen werden Qualitätsanforderungen an den Client erfasst.

	sehr wichtig	wichtig	unwichtig
Effizienz (Taktik)	X		
Lauffähigkeit	X		
Robustheit		X	
Änderbarkeit		X	
Übersichtlichkeit		X	

11 Ergänzungen

Ergänzungen definieren unter welchen Bedingungen die Ausarbeitung des Projektes stattfand.

7.1 Problemanalyse

Ein wesentlicher Bestandteil der Programmierung war es Probleme zu finden, diese zu analysieren und zu lösen. Dazu war das Testen des Programms unerlässlich, da nicht in allen Szenarien und Konstellationen des Spiels die gleichen oder überhaupt Fehler auftraten. Ein wichtiges Mittel um, wenn man nun die fehlerhafte Stelle im Spiel gefunden hatte, den Fehler auch im Quelltext zu finden ist der Befehl:

System.*out*.println("Test Text");

Durch diesen Befehl ist es später möglich, falls die betroffene Programmzeile mitsamt dieses Befehls den Fehler aufweist, in der Konsole von Eclipse den entsprechenden Text abzulesen, welcher durch den Befehl ausgegeben wird. In unserem Beispiel wäre dies der „Test Text“. Falls diese Ausgabe nun die letzte vor dem Absturz des Programms war, kann die betroffene Stelle im Quelltext gefunden werden und das Problem beseitigt werden.

12 Glossar

Client: (englisch client = Kunde) ist ein Computerprogramm, das Kontakt zu einem anderen Computerprogramm, dem Server aufnimmt, um dessen Dienstleistung zu nutzen.[12]

Orgware: beschreibt die Rahmenbedingung bei diesem IT-Projekt.[13]

ICQ: ist ein Instang-Messaging-Programm. Benutzer könne damit über das Internet miteinander chatten.[14]

Qualitätssicherung: hier soll auf evtl. Fehler in der Software gesucht werden.

Quelltext: ist der für den Menschen lesbare, in einer Programmiersprache (hier Java) geschriebene Text.

Server: (Software) ist ein Programm, das mit einem anderen Programm, dem Client kommuniziert, um ihm Zugang zu speziellen Dienstleistungen zu verschaffen.[15]

[12] http://de.wikipedia.org/wiki/Client, Zugriff am 17.1.2010, Zugriff am 17.1.2010 um 18:22 Uhr
[13] http://de.wikipedia.org/wiki/Orgware Zugriff am 17.1.2010, Zugriff am 17.1.2010 um 18:25 Uhr
[14] http://de.wikipedia.org/wiki/ICQ, Zugriff am 17.1.2010 am 17.1.2010 um 18:31 Uhr
[15] http://de.wikipedia.org/wiki/Server, Zugriff am 17.1.2010 am 17.1.2010 um 18:43

BERUFLICHE SCHULE WIRTSCHAFT . KIEL

LUDWIG ERHARD SCHULE

DER RAVENSBERG
und Abendgymnasium

Quellencode

Projekt:	Client für Brettspiel Hase & Igel
Autor:	Gruppe 2
Schule:	Regionales Bildungszentrum Kiel
	Standort der Ravensberg
Letzte Änderung:	17. Februar 2009

```
package gameClient;

import sc.plugin2010.framework.Hasenjoker;
import sc.plugin2010.framework.SpielClient;
import sc.plugin2010.framework.Spielbrett;
import sc.plugin2010.framework.Spieler;
import sc.plugin2010.framework.Spielerfarbe;
import sc.plugin2010.framework.Spielfeldtyp;

/**
 * Ein SimpleClient für das Spiel Hase und Igel. Zu beachten ist, dass sobald
 * eine Aktion, wie "frissSalat", auf einem Spieler aufgerufen wurde, sämtliche
 * Anweisungen, wie "setzeFigur", zum Beenden des Spiels führen.
 *
 * @author ffi
 *
 */
public class SimpleClient extends SpielClient
{

	// interne Zustaende
	private final Spielbrett	spielbrett;
	private final Spieler		eigenerSpieler;
	private final Spieler		gegner;
	public int                      doppel = 0;
	public int                      mali0  = 0;
	public int                      mali1  = 0;
	public int                      mali2  = 0;
	public int                      mali3  = 0;
	public int                      mali4  = 0;
	public int                      mali5  = 0;
	public boolean                  karo0  = false;
	public boolean                  karo1  = false;
	public boolean                  karo2  = false;
	public boolean                  karo3  = false;
	public boolean                  karo4  = false;
	public boolean                  karo5  = false;
	public int                      salat0 = 0;
	public int                      salat1 = 0;
	public int                      salat2 = 0;

	/**
	 * wird beim Spielstart aufgerufen
	 *
	 * @param ip
	 *          die IP mit welcher der Client verbinden soll
	 * @param port
	 *          der Port mit welchem der Client verbinden soll
	 * @param spielreservierung
	 *          falls eine Spielreservierung beim Server für den Client
	 *          vorliegt
```

```
 */
public boolean macheStandardAktion()
{
	// zeigt an, ob eine Standardaktion ausgeführt wurde
	boolean zugGemacht = false;

	// Wenn keine Aktion möglich ist, dann muss der Spieler aussetzen
	if (eigenerSpieler.kannSalatFressen())
	{
		eigenerSpieler.frissSalat();
		System.out.println("Salat gefressen");
		zugGemacht = true;
	}
	else if (eigenerSpieler.mussAussetzen())
	{
		eigenerSpieler.setzeAus();
		System.out.println("Spieler muss aussetzen");
		zugGemacht = true;
	}

	else if (eigenerSpieler.holeHasenjoker().size() > 0)
	{
		// spiele den ersten Hasenjoker auf der Hand
		if (eigenerSpieler.kannHasenjokerSpielen(eigenerSpieler
				.holeHasenjoker().get(0)))
		{


eigenerSpieler.spieleHasenjoker(eigenerSpieler.holeHasenjoker()
					.get(0));

			System.out.println("Hasenjoker gespielt");
			zugGemacht = true;
		}
	}

	return zugGemacht;

}

public SimpleClient(String ip, int port, String spielreservierung)
{
	// verbinde zum Spiel
	super(ip, port, spielreservierung);

	eigenerSpieler = new Spieler();
	gegner = new Spieler();
	spielbrett = new Spielbrett(eigenerSpieler, gegner);
	eigenerSpieler.setzteGegner(gegner);
	eigenerSpieler.setzteSpielbrett(spielbrett);
```

```
		// gib interne Referenzen der Logik
		super.setzeSpielbrett(spielbrett);
		super.setzeSpieler(eigenerSpieler);
		super.setzeGegner(gegner);

		super.starteSpiel();
	}

	void Neu1()
	{
		System.out.println("In Methode Neu1");
		int feldNummer = eigenerSpieler.holeFeldnummer();
		int letztePosition2 = spielbrett.holeVorherigesSpielfeldNachTyp(
				Spielfeldtyp.POSITION_2, 10);
		int Hasenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
				Spielfeldtyp.HASE, feldNummer);

		if (eigenerSpieler.holeSpielerfarbe() == Spielerfarbe.ROT)
		{

			eigenerSpieler.setzeFigur(Hasenfeld);
			eigenerSpieler.spieleHasenjoker(Hasenjoker.FRISS_SALAT);
		}
		else if ((spielbrett.holeSpielfeldtyp(letztePosition2 + 1).equals(
				Spielfeldtyp.KAROTTEN)
				&& (letztePosition2 + 1) < 10 && eigenerSpieler
				.kannAufFeldZiehen(letztePosition2 + 1))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 2).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 2) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 2))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 3).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 3) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 3))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 4).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 4) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 4))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 5).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 5) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 5))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 6).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 6) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 6))
				|| (spielbrett.holeSpielfeldtyp(letztePosition2 + 7).equals(
						Spielfeldtyp.KAROTTEN)
						&& (letztePosition2 + 7) < 10 && eigenerSpieler
						.kannAufFeldZiehen(letztePosition2 + 7)))
```

```
{
        if (spielbrett.holeSpielfeldtyp(5).equals(Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(5))
        {
                eigenerSpieler.setzeFigur(5);
        }
        else if (spielbrett.holeSpielfeldtyp(6).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(6))
        {
                eigenerSpieler.setzeFigur(6);
        }
        else if (spielbrett.holeSpielfeldtyp(4).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(4))
        {
                eigenerSpieler.setzeFigur(4);
        }
        else if (spielbrett.holeSpielfeldtyp(7).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(7))
        {
                eigenerSpieler.setzeFigur(7);
        }
        else if (spielbrett.holeSpielfeldtyp(3).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(3))
        {
                eigenerSpieler.setzeFigur(3);
        }
        else if (spielbrett.holeSpielfeldtyp(8).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(8))
        {
                eigenerSpieler.setzeFigur(8);
        }
        else if (spielbrett.holeSpielfeldtyp(2).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(2))
        {
                eigenerSpieler.setzeFigur(2);
        }
        else if (spielbrett.holeSpielfeldtyp(9).equals(
                        Spielfeldtyp.POSITION_2)
                        && eigenerSpieler.kannAufFeldZiehen(9))
        {
                eigenerSpieler.setzeFigur(9);
        }
}

else if (eigenerSpieler.kannAufFeldZiehen(Hasenfeld))
```

```
        {
                eigenerSpieler.setzeFigur(Hasenfeld);
                eigenerSpieler.spieleHasenjoker(Hasenjoker.FRISS_SALAT);

        }
        else
        {
                eigenerSpieler.setzeFigur(Hasenfeld + 1);
        }
}

void Neu2()
{
        System.out.println("In Methode Neu2");

        int Position = eigenerSpieler.holeFeldnummer();
        int gegnerposition = gegner.holeFeldnummer();
        int Hasenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                        Spielfeldtyp.HASE, Position);
        int Karottenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                        Spielfeldtyp.KAROTTEN, Position);
        if (gegnerposition > 10)
        {
                eigenerSpieler.setzeFigur(10);
                gewarteteRunden = gewarteteRunden - 2;
        }

        else if (eigenerSpieler.hatHasenjokerTyp(Hasenjoker.FRISS_SALAT)
                        && Hasenfeld < 10
                        && eigenerSpieler.kannAufFeldZiehen(Hasenfeld))
        {

                eigenerSpieler.setzeFigur(Hasenfeld);
                eigenerSpieler.spieleHasenjoker(Hasenjoker.FRISS_SALAT);

        }
        else if (eigenerSpieler.kannAufFeldZiehen(Karottenfeld))
        {

                eigenerSpieler.setzeFigur(Karottenfeld);
        }
        else
        {
                int zweiKarottenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                                Spielfeldtyp.KAROTTEN, Karottenfeld);
                eigenerSpieler.setzeFigur(zweiKarottenfeld);
        }
}

void Neu3()
{
```

```
		System.out.println("In Methode Neu3");
		int feldNummer = eigenerSpieler.holeFeldnummer();

		int gegnerposition = gegner.holeFeldnummer();
		if (gegnerposition <= 10)
		{

			int Karottenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
					Spielfeldtyp.KAROTTEN, feldNummer);
			if (spielbrett.holeSpielfeldtyp(feldNummer).equals(
					Spielfeldtyp.KAROTTEN))
			{
				eigenerSpieler.nimmKarotten();

			}
			else if (eigenerSpieler.kannAufFeldZiehen(Karottenfeld))
			{

				eigenerSpieler.setzeFigur(Karottenfeld);
			}
			else
			{
				int zweiKarottenfeld = spielbrett

.holeNaechstesSpielfeldNachTyp(Spielfeldtyp.KAROTTEN,
                                                    Karottenfeld);
				eigenerSpieler.setzeFigur(zweiKarottenfeld);
			}

		}

		else
		{
			eigenerSpieler.setzeFigur(10);
			gewarteteRunden = gewarteteRunden - 1;
		}
}

void Neu4()
{
		System.out.println("In Methode Neu4");
		if (spielbrett.holeRunde() >= 7)
			GenugGefuttert = true;
		if (gegner.holeFeldnummer() <= 10)
		{
			System.out.println("In Methode Neu4, in if");
			eigenerSpieler.nimmKarotten();
			gewarteteRunden = gewarteteRunden + 1;
		}
		else
		{
```

```
                if (gewesen == false)
                {
                        System.out.println("In Methode Neu4, in else");
                        eigenerSpieler.setzeFigur(10);
                        GenugGefuttert = true;
                        gewesen = true;
                }
        }
}

boolean         gewesen         = false;

void Neu4b()
{
        if (gewesen == false)
        {
                System.out.println("In Methode Neu4b");

                eigenerSpieler.setzeFigur(10);
                gewesen = true;
        }
        else
        {

        }
}

void Neu5()
{
        System.out.println("In Methode Neu5");

        int Position = eigenerSpieler.holeFeldnummer();
        int Hasenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                        Spielfeldtyp.HASE, Position);
        if (eigenerSpieler.hatHasenjokerTyp(Hasenjoker.FRISS_SALAT)
                        && Hasenfeld < 22
                        && eigenerSpieler.kannAufFeldZiehen(Hasenfeld))
        {
                eigenerSpieler.setzeFigur(Hasenfeld);
                eigenerSpieler.spieleHasenjoker(Hasenjoker.FRISS_SALAT);
        }
        else
        {
                if (eigenerSpieler.kannAufFeldZiehen(17)
                                && !spielbrett.holeSpielfeldtyp(17).equals(
                                                Spielfeldtyp.HASE))
                {
                        eigenerSpieler.setzeFigur(17);
                }
                else
                {
```

```
                        eigenerSpieler.setzeFigur(18);
                    }

                }
            }

            void Neu6()
            {
                System.out.println("In Methode Neu6");

                if (eigenerSpieler.kannAufFeldZiehen(22))
                {
                    eigenerSpieler.setzeFigur(22);
                }
                else
                {
                    int Karottenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                            Spielfeldtyp.KAROTTEN,
eigenerSpieler.holeFeldnummer());
                    eigenerSpieler.setzeFigur(Karottenfeld);
                }
            }

            void KohlBesetzt()
            {
                System.out.println("In Methode KohlBesetzt");

                if (gegner.holeFeldnummer() == 22)
                {
                    eigenerSpieler.nimmKarotten();
                    gewarteteRunden++;
                }
                else
                {

                    eigenerSpieler.setzeFigur(22);
                    GenugGefuttert = true;

                }
            }

            void KohlBesetztzwei()
            {
                System.out.println("In Methode KohlBesetztzwei");

                eigenerSpieler.setzeFigur(22);
                gewarteteRunden++;
            }

            void Neu7()
            {
```

```
                System.out.println("In Methode Neu7");

                if (eigenerSpieler.kannZurueckfallen())
                {
                        eigenerSpieler.zurueckAufLetztenIgel();
                        // zugGemacht = true;
                }
                else
                {
                        int Position = eigenerSpieler.holeFeldnummer();
                        int Hasenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                                        Spielfeldtyp.HASE, Position);
                        eigenerSpieler.setzeFigur(Hasenfeld);
                        eigenerSpieler.spieleHasenjoker(Hasenjoker.FALLE_ZURUECK);
                        // zugGemacht = true;
                }
        }

        void Neu8()
        {
                System.out.println("In Methode Neu8");

                if (eigenerSpieler.kannAufFeldZiehen(22))
                {
                        eigenerSpieler.setzeFigur(22);
                }
                else
                {

                        int Karottenfeld = spielbrett.holeNaechstesSpielfeldNachTyp(
                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler.holeFeldnummer());
                        eigenerSpieler.setzeFigur(Karottenfeld);
                        gewarteteRunden++;
                }
                doppel++;

        }

        void ab22()
        {
                System.out.println("Eintreten in Feldphase ab Feld 22");
                if (spielbrett.istErster(eigenerSpieler) && doppel == 2)
                {
                        System.out.println("Eintreten in Feldphase ab Feld 22 IF-Zweig");

                        // Feld
29++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        if (eigenerSpieler.holeFeldnummer() <= (28 - mali0))
                        {
                                System.out.println("Eintreten in Feldphase ab Feld 29");
```

```
                        if (spielbrett.holeSpielfeldtyp(29) == Spielfeldtyp.HASE)
                        {

                                mali0++;
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(29 - mali0)
                                        && karo0 == false)
                        {
                                eigenerSpieler.setzeFigur(spielbrett
                                                .holeNaechstesSpielfeldNachTyp(
                                                                Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                karo0 = true;
                        }
                        else
                        {
                                if (!eigenerSpieler.kannAufFeldZiehen(29 - mali0))
                                {
                                        eigenerSpieler.nimmKarotten();
                                }
                                else
                                {
                                        eigenerSpieler.setzeFigur(29 - mali0);
                                }
                        }
                        eigenerSpieler.setzeFigur(29 - mali0);
                }

                // Feld
36++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                else if (eigenerSpieler.holeFeldnummer() <= (35 - mali1))
                {
                        System.out.println("Eintreten in Feldphase ab Feld 36");
                        if (spielbrett.holeSpielfeldtyp(36) == Spielfeldtyp.HASE)
                        {

                                mali1++;
                        }
                        if (spielbrett.holeSpielfeldtyp(35) == Spielfeldtyp.HASE)
                        {

                                mali1++;
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(36 - mali1)
                                        && karo1 == false)
                        {
                                eigenerSpieler.setzeFigur(spielbrett
                                                .holeNaechstesSpielfeldNachTyp(
```

```
                                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                karo1 = true;
                        }
                        else
                        {
                                if (!eigenerSpieler.kannAufFeldZiehen(36 - mali1))
                                {
                                        eigenerSpieler.nimmKarotten();
                                }
                                else
                                {
                                        eigenerSpieler.setzeFigur(36 - mali1);
                                }
                        }
                        eigenerSpieler.setzeFigur(36 - mali1);
                }

                // Feld
42++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                else if (eigenerSpieler.holeFeldnummer() <= (41 - mali2))
                {
                        System.out.println("Eintreten in Feldphase ab Feld 42");
                        if (spielbrett.holeSpielfeldtyp(42) == Spielfeldtyp.HASE)
                        {

                                mali2++;
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(42 - mali2)
                                        && karo2 == false)
                        {
                                eigenerSpieler.setzeFigur(spielbrett
                                                .holeNaechstesSpielfeldNachTyp(
                                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                karo2 = true;
                        }
                        else
                        {
                                if (!eigenerSpieler.kannAufFeldZiehen(42 - mali2))
                                {
                                        eigenerSpieler.nimmKarotten();
                                }
                                else
                                {
                                        eigenerSpieler.setzeFigur(42 - mali2);
```

```
                        }
                    }
                    eigenerSpieler.setzeFigur(42 - mali2);
                }

                // Feld
49++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                else if (eigenerSpieler.holeFeldnummer() <= (48 - mali3))
                {
                    System.out.println("Eintreten in Feldphase ab Feld 49");
                    if (spielbrett.holeSpielfeldtyp(49) == Spielfeldtyp.HASE)
                    {

                        mali3++;
                    }
                    if (eigenerSpieler.kannAufFeldZiehen(49 - mali3)
                            && karo3 == false)
                    {
                        eigenerSpieler.setzeFigur(spielbrett
                                .holeNaechstesSpielfeldNachTyp(
                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                        karo3 = true;
                    }
                    else
                    {
                        if (!eigenerSpieler.kannAufFeldZiehen(49 - mali3))
                        {
                            eigenerSpieler.nimmKarotten();
                        }
                        else
                        {
                            eigenerSpieler.setzeFigur(49 - mali3);
                        }
                    }
                    eigenerSpieler.setzeFigur(49 - mali3);
                }

                // Feld
55++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                else if (eigenerSpieler.holeFeldnummer() <= (54 - mali4))
                {
                    System.out.println("Eintreten in Feldphase ab Feld 55");
                    if (spielbrett.holeSpielfeldtyp(55) == Spielfeldtyp.HASE)
                    {

                        mali4++;
```

```
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(55 - mali4)
                                && karo4 == false)
                        {
                                eigenerSpieler.setzeFigur(spielbrett
                                        .holeNaechstesSpielfeldNachTyp(
                                                Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                karo4 = true;
                        }
                        else
                        {
                                if (!eigenerSpieler.kannAufFeldZiehen(55 - mali4))
                                {
                                        if (spielbrett.holeSpielfeldtyp(eigenerSpieler
                                                .holeFeldnummer()) !=
Spielfeldtyp.KAROTTEN)
                                        {

        spielbrett.naechstesFreiesFeld(eigenerSpieler,

        eigenerSpieler.holeFeldnummer());
                                        }
                                        else
                                        {
                                                eigenerSpieler.nimmKarotten();
                                        }
                                }
                                else
                                {
                                        eigenerSpieler.setzeFigur(55 - mali4);
                                }
                        }
                        eigenerSpieler.setzeFigur(55 - mali4);
                }
                // Feld64 -
Ziel**********************************************

                else if (eigenerSpieler.holeFeldnummer() <= (54 - mali5))
                {
                        System.out.println("Eintreten in Feldphase des Ziel Feld 64");
                        if (spielbrett.holeSpielfeldtyp(64) == Spielfeldtyp.HASE)
                        {

                                mali5++;
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(64 - mali5)
                                && karo4 == false)
                        {
```

```
                                        eigenerSpieler.setzeFigur(spielbrett
                                                .holeNaechstesSpielfeldNachTyp(
                                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                        karo5 = true;
                                }
                                else
                                {
                                        if (!eigenerSpieler.kannAufFeldZiehen(64 - mali5))
                                        {
                                                eigenerSpieler.nimmKarotten();
                                        }
                                        else
                                        {
                                                eigenerSpieler.setzeFigur(64 - mali5);
                                        }
                                }
                                eigenerSpieler.setzeFigur(64 - mali5);
                        }

                        //
##################################################################
                }
                else
                {
                        System.out.println("Eintreten in Feldphase ab Feld 22 ELSE");
                        if (spielbrett.holeNaechstesSpielfeldNachTyp(
                                        Spielfeldtyp.POSITION_2,
eigenerSpieler.holeFeldnummer()) < gegner
                                        .holeFeldnummer())
                        {
                                eigenerSpieler.setzeFigur(spielbrett

        .holeNaechstesSpielfeldNachTyp(Spielfeldtyp.POSITION_2,

        eigenerSpieler.holeFeldnummer()));

                        }
                        else if (((eigenerSpieler.holeFeldnummer() >= 34) && (eigenerSpieler
                                        .holeFeldnummer() <= 41))
                                        && (eigenerSpieler.kannAufFeldZiehen(42) == true))
                        {
                                eigenerSpieler.setzeFigur(42);
                        }
                        else if (((eigenerSpieler.holeFeldnummer() >= 34) && (eigenerSpieler
                                        .holeFeldnummer() <= 41))
                                        && (spielbrett.holeSpielfeldtyp(eigenerSpieler
                                                        .holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN))
```

```
                        {
                                eigenerSpieler.nimmKarotten();
                        }
                        else if (((eigenerSpieler.holeFeldnummer() >= 34) && (eigenerSpieler
                                        .holeFeldnummer() <= 41))
                                        && (eigenerSpieler.kannAufFeldZiehen(42) != true))
                        {
                                if (eigenerSpieler.kannAufFeldZiehen(spielbrett

        .holeNaechstesSpielfeldNachTyp(Spielfeldtyp.KAROTTEN,

        eigenerSpieler.holeFeldnummer())))
                                {
                                        eigenerSpieler.setzeFigur(spielbrett
                                                        .holeNaechstesSpielfeldNachTyp(
                                                                        Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                }
                                else
                                {
                                        eigenerSpieler.zurueckAufLetztenIgel();
                                }

                        }
                        else if ((eigenerSpieler.holeFeldnummer() >= 22)
                                        && (eigenerSpieler.holeFeldnummer() <= 29))
                        {
                                if (eigenerSpieler.kannAufFeldZiehen(32)
                                                && (spielbrett.holeSpielfeldtyp(32) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(32);
                                }
                                else if (eigenerSpieler.kannAufFeldZiehen(31)
                                                && (spielbrett.holeSpielfeldtyp(31) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(31);

                                }
                                else if (eigenerSpieler.kannAufFeldZiehen(29)
                                                && (spielbrett.holeSpielfeldtyp(29) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(29);
                                }
                                else if (eigenerSpieler.kannAufFeldZiehen(28)
                                                && (spielbrett.holeSpielfeldtyp(28) ==
Spielfeldtyp.KAROTTEN))
```

```
                {
                        eigenerSpieler.setzeFigur(28);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(27)
                                && (spielbrett.holeSpielfeldtyp(27) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(27);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(26)
                                && (spielbrett.holeSpielfeldtyp(26) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(26);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(25)
                                && (spielbrett.holeSpielfeldtyp(25) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(25);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(24)
                                && (spielbrett.holeSpielfeldtyp(24) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(24);
                }
                else if (spielbrett.holeSpielfeldtyp(eigenerSpieler
                                .holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN)
                {
                        eigenerSpieler.nimmKarotten();
                }
                else
                {
                        System.out.println("Igel");

                        eigenerSpieler.zurueckAufLetztenIgel();
                }

        }
        else if ((eigenerSpieler.holeFeldnummer() >= 30)
                        && (eigenerSpieler.holeFeldnummer() <= 34))
        {

                if (eigenerSpieler.kannAufFeldZiehen(36)
                                && (spielbrett.holeSpielfeldtyp(36) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(36);
                }
```

```
                        else if (eigenerSpieler.kannAufFeldZiehen(35)
                                && (spielbrett.holeSpielfeldtyp(35) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(35);

                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(34)
                                && (spielbrett.holeSpielfeldtyp(34) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(34);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(33)
                                && (spielbrett.holeSpielfeldtyp(33) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(33);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(32)
                                && (spielbrett.holeSpielfeldtyp(32) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(32);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(31)
                                && (spielbrett.holeSpielfeldtyp(31) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(31);
                        }

                        else if (spielbrett.holeSpielfeldtyp(eigenerSpieler
                                .holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN)
                        {
                                eigenerSpieler.nimmKarotten();
                        }
                        else
                        {
                                System.out.println("Igel");

                                eigenerSpieler.zurueckAufLetztenIgel();
                        }
                }
                else if (eigenerSpieler.holeFeldnummer() <= (41 - mali2))
                {
                        System.out.println("Eintreten in Feldphase ab Feld 42");
                        if (spielbrett.holeSpielfeldtyp(42) == Spielfeldtyp.HASE)
                        {
```

```
                                mali2++;
                        }
                        if (eigenerSpieler.kannAufFeldZiehen(42 - mali2)
                                        && karo2 == false)
                        {
                                eigenerSpieler.setzeFigur(spielbrett
                                                .holeNaechstesSpielfeldNachTyp(
                                                                Spielfeldtyp.KAROTTEN,
eigenerSpieler

        .holeFeldnummer()));
                                karo2 = true;
                        }
                        else
                        {
                                if (!eigenerSpieler.kannAufFeldZiehen(42 - mali2))
                                {
                                        eigenerSpieler.nimmKarotten();
                                }
                                else
                                {
                                        eigenerSpieler.setzeFigur(42 - mali2);
                                }
                        }
                        eigenerSpieler.setzeFigur(42 - mali2);
                }
                else if ((eigenerSpieler.holeFeldnummer() >= 42)
                                && (eigenerSpieler.holeFeldnummer() <= 48))
                {

                        if (eigenerSpieler.kannAufFeldZiehen(51)
                                        && (spielbrett.holeSpielfeldtyp(51) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(51);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(49)
                                        && (spielbrett.holeSpielfeldtyp(49) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(49);

                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(48)
                                        && (spielbrett.holeSpielfeldtyp(48) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(48);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(47)
```

```
                                        && (spielbrett.holeSpielfeldtyp(47) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(47);
                                }
                                else if (eigenerSpieler.kannAufFeldZiehen(46)
                                        && (spielbrett.holeSpielfeldtyp(46) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(46);
                                }
                                else if (eigenerSpieler.kannAufFeldZiehen(45)
                                        && (spielbrett.holeSpielfeldtyp(45) ==
Spielfeldtyp.KAROTTEN))
                                {
                                        eigenerSpieler.setzeFigur(45);
                                }

                                else if (spielbrett.holeSpielfeldtyp(eigenerSpieler
                                        .holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN)
                                {
                                        eigenerSpieler.nimmKarotten();
                                }
                                else
                                {
                                        System.out.println("Igel");

                                        eigenerSpieler.zurueckAufLetztenIgel();
                                }
                        }
                }
        }

        void ende()
        {
                if (eigenerSpieler.kannAufFeldZiehen(64) == true)
                {

                        eigenerSpieler.setzeFigur(64);
                }
                else if (eigenerSpieler.holeKarottenAnzahl() > 105)
                {
                        System.out.println("ende hat zuviele karotten");
                        if (spielbrett.holeSpielfeldtyp(eigenerSpieler.holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN)
                        {
                                eigenerSpieler.gibKarottenAb();
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(63)
```

```
                                        && (spielbrett.holeSpielfeldtyp(63) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(63);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(62)
                                        && (spielbrett.holeSpielfeldtyp(62) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(62);

                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(61)
                                        && (spielbrett.holeSpielfeldtyp(61) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(61);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(60)
                                        && (spielbrett.holeSpielfeldtyp(60) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(60);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(59)
                                        && (spielbrett.holeSpielfeldtyp(59) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(59);
                        }
                }
                else if ((eigenerSpieler.holeFeldnummer() > 50)
                                && (eigenerSpieler.holeFeldnummer() < 57))
                {

                        System.out.println("ende zwischen 50 un 57");
                        // if (spielbrett.holeSpielfeldtyp(eigenerSpieler.holeFeldnummer())
                        // == Spielfeldtyp.KAROTTEN)
                        // {
                        // eigenerSpieler.gibKarottenAb();
                        // }
                        if (eigenerSpieler.kannAufFeldZiehen(63)
                                        && (spielbrett.holeSpielfeldtyp(63) ==
Spielfeldtyp.KAROTTEN))
                        {
                                eigenerSpieler.setzeFigur(63);
                        }
                        else if (eigenerSpieler.kannAufFeldZiehen(62)
                                        && (spielbrett.holeSpielfeldtyp(62) ==
Spielfeldtyp.KAROTTEN))
                        {
```

```
                        eigenerSpieler.setzeFigur(62);

                }
                else if (eigenerSpieler.kannAufFeldZiehen(61)
                        && (spielbrett.holeSpielfeldtyp(61) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(61);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(60)
                        && (spielbrett.holeSpielfeldtyp(60) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(60);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(59)
                        && (spielbrett.holeSpielfeldtyp(59) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(59);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(58)
                        && (spielbrett.holeSpielfeldtyp(58) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(58);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(55)
                        && (spielbrett.holeSpielfeldtyp(55) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(55);
                }
                else if (eigenerSpieler.kannAufFeldZiehen(54)
                        && (spielbrett.holeSpielfeldtyp(54) ==
Spielfeldtyp.KAROTTEN))
                {
                        eigenerSpieler.setzeFigur(54);
                }
                else if (spielbrett.holeSpielfeldtyp(eigenerSpieler
                        .holeFeldnummer()) == Spielfeldtyp.KAROTTEN)
                {
                        eigenerSpieler.nimmKarotten();
                }
                else
                {
                        System.out.println("Igel");

                        eigenerSpieler.zurueckAufLetztenIgel();
                }
        }
```

```
                else
                {
                        System.out.println(" ende nächstesfeld abfragen");

                        if (spielbrett.holeSpielfeldtyp(eigenerSpieler.holeFeldnummer()) ==
Spielfeldtyp.KAROTTEN)
                        {
                                System.out.println("Ende frisst karotten");

                                eigenerSpieler.nimmKarotten();
                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer())) !=
Spielfeldtyp.HASE)
                        {
                                System.out.println("Nächstes mögliche");
                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()));

                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer()) + 1) !=
Spielfeldtyp.HASE)
                        {
                                System.out.println("Nächstes mögliche1");

                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()) + 1);

                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer()) + 2) !=
Spielfeldtyp.HASE)
                        {
                                System.out.println("Nächstes mögliche2");

                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()) + 2);

                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer()) + 3) !=
Spielfeldtyp.HASE)
                        {
```

```
                                System.out.println("Nächstes mögliche3");

                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()) + 3);

                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer()) + 4) !=
Spielfeldtyp.HASE)
                        {
                                System.out.println("Nächstes mögliche4");

                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()) + 4);

                        }
                        else if (spielbrett.holeSpielfeldtyp(spielbrett
                                        .naechstesFreiesFeld(eigenerSpieler, eigenerSpieler
                                                        .holeFeldnummer()) + 5) !=
Spielfeldtyp.HASE)
                        {
                                System.out.println("Nächstes mögliche5");

                                eigenerSpieler.setzeFigur(spielbrett.naechstesFreiesFeld(
                                                eigenerSpieler,
eigenerSpieler.holeFeldnummer()) + 5);

                        }
                        else
                        {
                                System.out.println("Ende keine funktion greift");
                        }
                }
        }

        int                     gewarteteRunden     = 0;
        private boolean         GenugGefuttert      = false;

        @Override
        public void zugAngefordert()
        {
                System.out.println("***** Runden: *****");
                System.out.println("spielbrett.holeRunde " + spielbrett.holeRunde());
                System.out.println("gewarteteRunden " + gewarteteRunden);
                if (eigenerSpieler.mussAussetzen())
                {
                        eigenerSpieler.setzeAus();
                        gewarteteRunden++;
```

```
}
if (eigenerSpieler.kannSalatFressen())
{
        eigenerSpieler.frissSalat();
        System.out.println("Salat gefressen");
        gewarteteRunden++;
}
if (eigenerSpieler.holeFeldnummer() >= 50)
{
        System.out.println("Ende wird eingeleitet");
        ende();
}

else if (spielbrett.holeRunde() == 0)
{
        Neu1();
}

else if (spielbrett.holeRunde() == 1)
{
        Neu2();
}
else if (spielbrett.holeRunde() == 2 + gewarteteRunden)
{
        Neu3();
}
else if (spielbrett.holeRunde() == 3 + gewarteteRunden)
{
        if (!GenugGefuttert)
        {
                Neu4();
        }
        else
        {

                Neu4b();
        }
}
else if (spielbrett.holeRunde() == 4 + gewarteteRunden)
{
        System.out.println("Starte NEU5");
        Neu5();
}
else if (spielbrett.holeRunde() == 5 + gewarteteRunden)
{
        Neu6();
}
else if (spielbrett.holeRunde() == 6 + gewarteteRunden)
{
        if (eigenerSpieler.holeFeldnummer() == 22)
        {
```

```
                Neu7();
        }
        else
        {
                GenugGefuttert = false;
                if (!GenugGefuttert)
                {
                        KohlBesetzt();
                }
                else
                {

                        KohlBesetztzwei();
                }
        }

}
else if (spielbrett.holeRunde() == 7 + gewarteteRunden)
{
        if (eigenerSpieler.holeFeldnummer() == 19)
        {
                Neu8();
        }
        else
        {
                GenugGefuttert = false;
                if (!GenugGefuttert)
                {
                        KohlBesetzt();
                }
                else
                {

                        KohlBesetztzwei();
                }
        }

}

else if (eigenerSpieler.holeFeldnummer() >= 22)
{
        if (eigenerSpieler.kannSalatFressen())
        {
                eigenerSpieler.frissSalat();
                System.out.println("Salat gefressen");
        }
        ab22();
}
else
        System.out.println("********* NICHTS GEFUNDEN *******");
```

```
	}

	@Override
	public void spielBeendet(String[] statistik, boolean abgebrochen)
	{
		// Spiel wurde abgeschlossen
		if (abgebrochen)
		{
			System.out.println("Spiel wurde abgebrochen!");
		}
		else
		{
			System.out.println(statistik[0]);
			if (statistik[0].equals("1"))
			{
				System.out.println("Gewinner: Rot");
			}
			else if (statistik[0].equals("0"))
			{
				System.out.println("Verlierer: Rot");
			}

			System.out.println("Erreichtes Feld Rot: " + statistik[1]);

			if (statistik[2].equals("1"))
			{
				System.out.println("Gewinner: Blau");
			}
			else if (statistik[2].equals("0"))
			{
				System.out.println("Verlierer: Blau");
			}

			System.out.println("Erreichtes Feld Blau: " + statistik[3]);
		}

	}
}
```