Sommersemester -Juli 2007-

Dokumentation

„Heroes of SEPVenture“

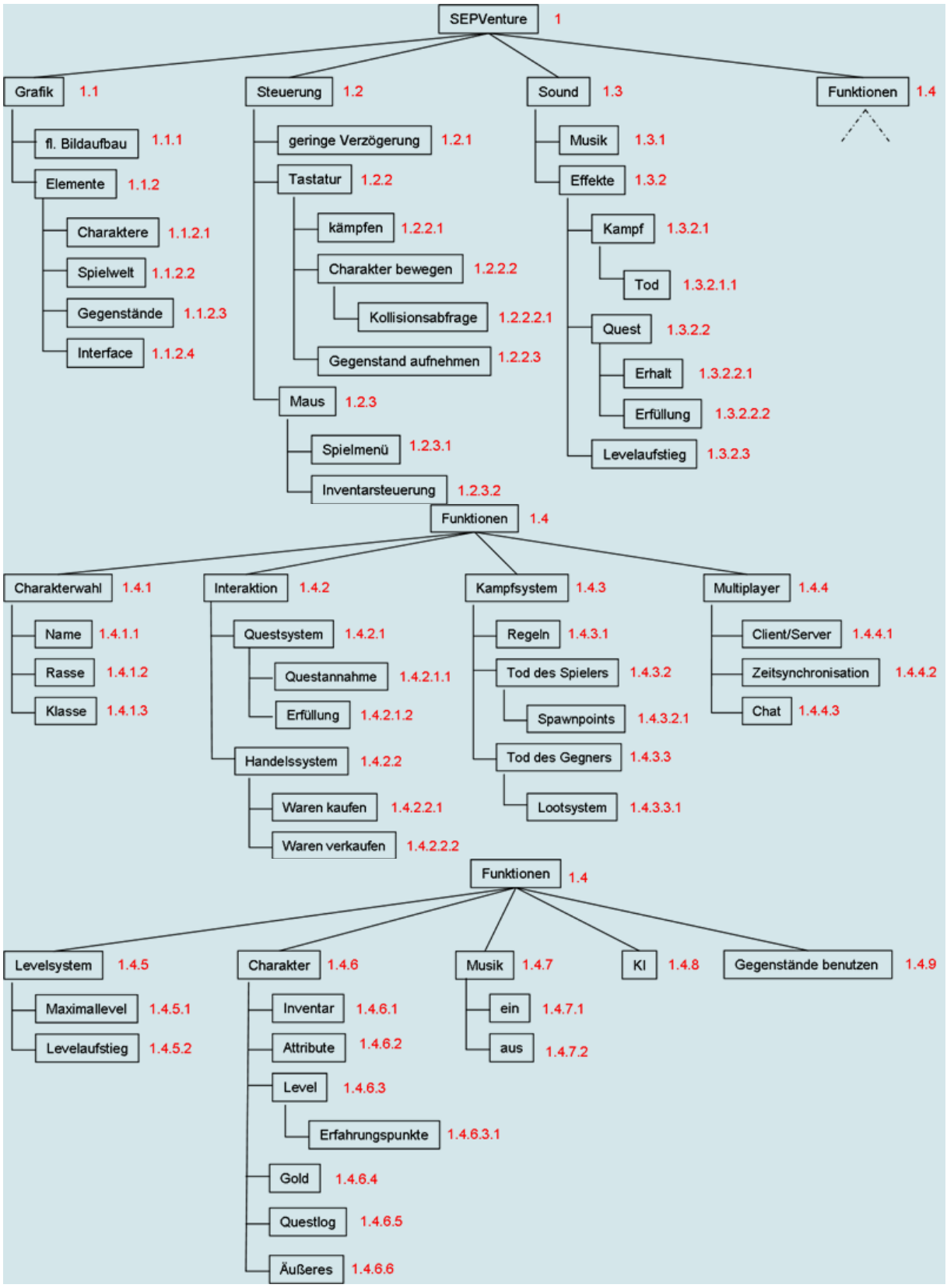
Gruppe N

Jiew-Fat Au, Philipp Bohn, Sebastian Dettmann, Dustin Hebgen, Boris Köther, Jan Rekers, Benedikt Ritter, Frank Schürmann, Annette Trost, Lena Varnholt

Inhaltsverzeichnis

1.0 Anforderungen

1.1 Featurebaum



1 SEP Venture

Dies ist der vorläufige Projektname unseres Online-Rollenspiels der Gruppe N.

1.1 Grafik

Hier werden alle Punkte genannt, die relevant für die Grafik sind.
Die Umsetzung des Spiels erfolgt in einer 2D Engine.

1.1.1 fl. Bildaufbau

Damit das Geschehen in der Spielwelt in Echtzeit ablaufen kann, wird für die Grafik ein flüssiger Bildaufbau benötigt.

1.1.2 Elemente

Elemente sind die wesentlichen Bestandteile der Grafik, die der Spieler auf seinem Monitor zu sehen bekommt. Sie haben Merkmale, an denen der Spieler erkennt, um was es sich in dem Spiel handelt und die es ihm ermöglichen, bei ähnlichen Elementen (wie z.B. dieselbe Rasse Engel, aber unterschiedliche Klassen (Magier oder Krieger)) durch das Äußere die Unterschiede zu erkennen. (Zu den Elementen gehören, Charaktere und Gegenstände, ebenso wie landschaftliche Begebenheiten und Interfaces.)

1.1.2.1 Charaktere

Es gibt zwei verschiedene Arten von Charaktertypen.

1. Variante sind die Spielfiguren, welche ein Spieler auswählen kann, um mit diesem das Spiel zu bestreiten.
   Engel --> Magier, Engel --> Krieger
   Dämon --> Magier, Dämon --> Krieger

2. Variante sind weitere Figuren, wie z.B. virtuelle Personen (Händler, Questgeber) und Gegner die vom Computer gesteuert werden.

Jeder Charakter unterscheidet sich durch sein Äußeres.

1.1.2.2 Spielwelt

Die Spielwelt ist in drei große Abschnitte unterteilt, die sich thematisch und vom Levelbereich her unterscheiden.
Es gibt für die beiden Fraktionen je einen Startabschnitt in der die ersten 5 Level absolviert werden können. Spieler der Fraktion Himmel starten im Abschnitt "Himmel", Spieler der Fraktion Hölle starten logischerweise in der Hölle. Es wird für Spieler nicht möglich sein, den Startabschnitt der anderen Fraktion zu betreten.
Der dritte Abschnitt, in dem Spieler beider Fraktionen aufeinander treffen werden, wird der Bereich "Erde" sein. Hier können Spieler das Maximallevel erreichen.
Der Spielabschnitt der Erde ist in 9 Teilabschnitte unterteilt, der Spielabschnitt von Himmel und Hölle ist jeweils in 4 Teilabschnitte unterteilt. Jeder dieser Teilausschnitte wird den kompletten Bildschirm füllen. Kommt der Spieler an den Rand eines Teilabschnitts, so wird der nächste Teilabschnitt angezeigt.
Jeder Teilabschnitt wiederum ist unterteilt in 20x15 Felder. Eine Spielfigur wird ein Feld einnehmen. Dieses 20x15 Felder große Raster, dass nur die Information enthält, ob ein Feld betretbar ist oder nicht, wird über einen vorgezeichneten Hintergrund gelegt.
Die Spielwelt ist durch verschiedene computergezeichnete Bilder dargestellt (je ein Bild pro Teilabschnitt), auf denen sich die Charaktere bewegen und die Gegenstände sich befinden.

1.1.2.3 Gegenstände

Unter Gegenständen versteht man die grafische Repräsentation aller Gegenstände im Spiel. Bei diesen Gegenständen handelt es sich um:

- Waffen (Zauberstab(Magier), Wurfmesser(Krieger)) und Rüstung
- Heilmittel (kleiner Trank, großer Trank)
- Gegenstände, die zur Erfüllung einer Quest benötigt werden (Schlüssel, Brücke, Kind)
- Gold

1.1.2.4 Interface

Das grafische Interface stellt Bedienung und Spielelemente zu Verfügung und unterteilt den Bildschirm in seine wesentlichen Funktionen.
So gibt es folgende Menüs:
- Spiel-starten Menü mit den Punkten:
Login-Abschnitt (Felder zum Eingeben von Benutzername und Passwort und Button zum Bestätigen) und Button der zur Charaktererstellung weiterführt
-Charakter-Erstell-Bildschirm (Feld zum eingeben des Namens, Wahl von Rasse/Klasse mit Bildanzeige und die Auswahl zur Vergabe eines Attributpunktes, die anderen sind vorher festgelegt)
- Inventar (alle Gegenstände, die im Inventar des Charakters vorhanden sind, werden in einer Tabelle angezeigt)
- Questlog (Alle Quests, die der Charakter angenommen hat, werden aufgelistet. Die bestandenen Quests werden durch ein „abgeschlossen“ markiert, abgebrochene oder nicht bestandene Quests mit dem Wort „offen“.)
- Kampf-Bildschirm
- Charakterübersicht (zeigt dem Spieler Information über seinen Charakter: Tabelle mit Attributen und ihren Werten; Erfahrungspunkte => Level; vorhandenes Gold; etc.)
- Handel-Bildschirm (Links ist das Inventar der Käufers mit darunter stehender Anzeige des vorhandenen Goldes zu sehen, rechts das Inventar des Verkäufers, in der Mitte oben ist ein Modus-Wechseln-Button, unten ein Kaufen-Button bzw. Verkaufen Button. Der Spieler kann nun mit Hilfe des Modus-Wechseln-Buttons, zwischen seinem Inventar und dem des Händers wechseln. Ist er in seinem Inventar kann er mit dem Verkaufen-Button Gegenstände verkaufen, ist er im Inventar des Händlers, kann er mit den Kaufen-Button Gegenstände von ihm kaufen. Zwischen den beiden Buttons wird jeweils Name, Beschreibung und der Preis des im Inventar markierten Gegenstandes angezeigt.)

1.2 Steuerung

Hier werden alle Punkte genannt, die es dem Spieler ermöglichen sämtliche Features des Spiels durch Eingaben zu nutzen.

1.2.1 geringe Verzögerung

Geringe Verzögerungen auf Eingaben werden benötigt, um das Spiel in "Echtzeit" laufen zu lassen.

1.2.2 Tastatur

Hier wird die Tastatursteuerung zum Spielen des Spiels festgelegt.

1.2.2.1 kämpfen

Der Kampf findet in einem separaten Kampfbildschirm statt, beim Kampf kann der Spieler mit der Maus auswählen was für eine Aktionen sein Charakter durchführen soll, mögliche Aktionen sind: Angriff, kl. Heiltrank benutzen, gr. Heiltrank benutzen oder Spezialangriff (Buttons). Die Geschwindigkeit mit der man wieder an den Zug kommt, wird durch die

Gewandtheit bestimmt. Ein Kampf endet indem der Charakter des Spielers keine Lebenspunkte mehr hat oder der Gegner kampfunfähig ist.

1.2.2.2 Charakter bewegen

Der Spieler bewegt seine Figur mit Hilfe der Tastaturbelegung über das Spielfeld zu seinem gewünschten Ort.
Pfeil nach rechts: Charakter bewegt sich nach rechts
Pfeil nach links: Charakter bewegt sich nach links
Pfeil nach oben: Charakter bewegt sich nach oben
Pfeil nach unten: Charakter bewegt sich nach unten

1.2.2.2.1 Kollisionsabfrage

Bei jedem Schritt den die Spielfigur macht, wird geprüft, ob dieser Schritt möglich ist oder ob ein Hindernis (z.B. ein Baum oder ein Fluss) im Weg ist, das es unmöglich macht, sich weiter in diese Richtung zu bewegen.

1.2.2.3 Gegenstand aufnehmen

Ein Gegenstand, den man aufnehmen kann, liegt auf dem Spielfeld. Um ihn aufzunehmen, muss man über ihn drüber laufen und der Gegenstand wird dann im Inventar gespeichert.

1.2.3 Maus

Wird unterstützend zur Tastatur genutzt, um das Spiel dem Anwender gegenüber benutzerfreundlicher zu machen und ermöglicht es ihm z.B. Buttons anzuklicken und sich im Menü zu bewegen.

1.2.3.1 Spielmenü

Stellt dem Spieler verschiedene Grundfunktionen zu Verfügung.

1.2.3.2 Inventarsteuerung

Dient dazu, Gegenstände aus dem Inventar aufzunehmen, auszuwählen und zu benutzen.

1.3 Sound

Der Sound wird benötigt, um dem Spieler beim Spielen eine Atmosphäre zu bieten und ihm auditiv Geschehen zu vermitteln.
Hier fallen alle Aspekte drunter, die die musikalische Gestaltung des Spiels betreffen.

1.3.1 Musik

Hiermit ist die Hintergrundmusik des Spiels gemeint, die den Spieler in eine bestimmte Stimmung versetzt.
So kann sie z.B. spannungsfördernd sein o.ä.

1.3.2 Effekte

Effekte sind Geräusche, die bei bestimmten Handlungen auftreten und die diese realistischer erscheinen lassen.

1.3.2.1 Kampf

Hierbei handelt es sich um alle Effekte, die bei einer Kampfhandlung abgespielt werden.

1.3.2.1.1 Tod

Mit Hilfe eines solchen Effekts wird dem Spieler auditiv vermittelt, dass sein Charakter gerade gestorben ist und er den Kampf verloren hat oder auch, dass dein Gegner gerade gestorben ist und er somit den Kampf gewonnen hat.

1.3.2.2 Quest

Vermittelt dem Spieler auditiv, dass er Aufgaben angenommen oder beendet hat.

1.3.2.2.1 Erhalt

Dieser Effekt wird abgespielt, wenn der Spieler eine neue Quest annimmt.

1.3.2.2.2 Erfüllung

Dieser Effekt wird abgespielt, wenn eine Quest erfüllt wurde.

1.3.2.3 Levelaufstieg

Beim Aufstieg in ein höheres Level ertönt ein entsprechender Sound.

1.4 Funktionen

Stellt den gesamten Umfang der Handlungsmöglichkeiten des Spielers zur Verfügung.

1.4.1 Charakterwahl

Ermöglicht es dem Spieler zwischen verschiedenen Rassen und Klassen zu wählen und seinem Charakter einen Namen zu geben. Die Attributpunkte der Figur sind jeweils vorgegeben, der Spieler kann danach noch 1 Punkt selber verteilen.

1.4.1.1 Name

Jeder Spieler hat seinen eigenen Charakternamen anhand dessen er identifiziert werden kann und welchen er selbst wählen kann.

1.4.1.2 Rasse

Jede Rasse hat ihre eigenen Stärken und Schwächen, die es dem Spieler ermöglichen nach seinen eigenen Vorlieben das Spiel zu bestreiten. Die Rassen haben hierbei dieselben Attribute, deren Werte jedoch je nach Rasse und Klasse unterschiedlich sind.
Die zu unterscheidenden Rassen sind:
Engel
Dämonen
(Mensch als NPC)

1.4.1.3 Klasse

Ermöglicht es dem Spieler seinen Charakter noch differenzierter (über die Dauer des Spiels hinweg) zu individualisieren.
Er muss jedoch am Anfang des Spiel festlegt werden, welchen Beruf bzw. welche Klasse der Charakter ergreifen will.
Zur Auswahl stehen dabei:
Magier
Krieger
Je nach Wahl der Rasse und Klasse unterscheiden sich die Charaktere in ihren Attributwerten.

Engel/Magier:

Stärke	10
Ausdauer	10
Gewandtheit	11
Intelligenz	12

Engel/Krieger:

Stärke	11
Ausdauer	12
Gewandtheit	10
Intelligenz	10

Dämon/Magier:

Stärke	10
Ausdauer	10
Gewandtheit	12
Intelligenz	11

Dämon/Krieger:

Stärke	12
Ausdauer	11
Gewandtheit	10
Intelligenz	10

1.4.2 Interaktion

Ermöglicht es dem Spieler mit anderen Charakteren in Kontakt zu treten und auch "mit dem System an sich" (z.B. werden "von dem System" die Quests gestellt und ebenso der Handel mit NPCs ermöglicht).

1.4.2.1 Questsystem

Fordert vom Spieler, dass er eine Quest erfüllt um seinen Charakter weiterzuentwickeln.
Eine Quest kann aus mehreren Teilaufgaben bestehen.

Himmel

Quest Level 1 :
Questgeber für Quest 1 steht im Südwesten. Und erteilt den Auftrag: „Hier hast du drei Schlüssel, jeder passt zu einer Truhe. Finde die Truhe, die das Wertvollste enthält und bringe mir diesen Gegenstand.“ (Wenn man die Truhe öffnet, wird der entsprechende Gegenstand in das Inventar des Charakters geladen, so kann die Truhe auch nie leer sein.) In der ersten Truhe befindet sich ein kleiner Tran, in der zweiten Truhe ein großer Trank und in der dritten Truhe Gold. Übergibt der Charakter dem Questgeber einen falschen Gegenstand, so sagt dieser: „Das ist aber nicht das Wertvollste!“ Reicht der Charakter ihm das Gold so sagt der Questgeber: „Vielen Dank für das Gold, dafür darfst du die anderen gefundenen Gegenstände behalten!“
Quest Level 2 :
Questgeber für Quest 2 versperrt im Süden den Weg zu einem besonderen Bereich im Himmel, der erst zulässig ist, wenn der Charakter sich im 5. Level befindet. Ist dies der Fall, so kann er vom Questgeber die Quest annehmen, was zur Folge hat, dass der Questgeber zur Seite geht und den Charakter in den besonderen Bereich eintreten lässt. (Questgeber muss zur ursprünglichen Position zurückkehren) Innerhalt des Spezialbereichs, muss der Charakter gegen einen NPC kämpfen, um sich Zugang zum Weg auf die Erde über eine Brücke zu beschaffen. Erst dann kann er den Bereich wieder verlassen und beim Austritt (Questgeber geht zur Seite, wenn er die Information bekommt, dass der Spieler im Bereich steht) aus diesem Bereich öffnet sich ein Informationsfenster mit der Meldung: „Gratulation! Jetzt bist du bereit für das Leben auf der Erde!“
Der Charakter muss nun die Stelle finden, an der eine Brücke ihn zur Erde führt.

Hölle

Quest Level 1 :
Questgeber für Quest 1 steht im Südwesten und erteilt den Auftrag: „Hier hast du drei Schlüssel, jeder passt zu einer Truhe. Finde die Truhe, die das Wertvollste enthält und bringe mir diesen Gegenstand.“ (Wenn man die Truhe öffnet, wird der entsprechende Gegenstand in das Inventar des Charakters geladen, so kann die Truhe auch nie leer sein.) In der ersten Truhe befindet sich ein kleiner Tran, in der zweiten Truhe ein großer Trank und in der dritten Truhe Gold. Übergibt der Charakter dem Questgeber

einen falschen Gegenstand, so sagt dieser: „Das ist aber nicht das Wertvollste!“ Reicht der Charakter ihm das Gold so sagt der Questgeber: „Vielen Dank für das Gold, dafür darfst du die anderen gefundenen Gegenstände behalten!“
Quest Level 2 :
Questgeber für Quest 2 versperrt im Nordosten den Weg zu einem besonderen Bereich in der Hölle, der erst zulässig ist, wenn der Charakter sich im 5. Level befindet. Ist dies der Fall, so kann er vom Questgeber die Quest annehmen, was zur Folge hat, dass der Questgeber zur Seite geht und den Charakter in den besonderen Bereich eintreten lässt. (Questgeber muss zur ursprünglichen Position zurückkehren) Innerhalb des Spezialbereichs, muss der Charakter gegen einen NPC kämpfen, um sich Zugang zum Weg auf die Erde über eine Brücke zu beschaffen. Erst dann kann er den Bereich wieder verlassen und beim Austritt (Questgeber geht zur Seite, wenn er die Information bekommt, dass der Spieler im Bereich steht) aus diesem Bereich öffnet sich ein Informationsfenster mit der Meldung: „Gratulation! Jetzt bist du bereit für das Leben auf der Erde!“ Der Charakter muss nun die Stelle finden, an der eine Brücke ich zur Erde führt.

Erde
Quest Level 3 :
Der Questgeber für Quest 3 steht links in der Mitte am linken Rand und erteilt den Auftrag: "Sammle Kraft für deinen großen Kampf"
Der Charakter muss sich entweder in der oberen rechten Hälfte des Spielfeldes Tränke erkämpfen oder bei genug Gold diese kaufen um einen besonders starken Gegner besiegen zu können. Wenn der Spieler meint genügend Tränke gesammelt zu haben (er weiß ja nicht wie viele er braucht), geht er zum Questgeber zurück und erhält eine der beiden Infos:
a) "Du hast nicht genügend Tränke für den großen Trank gesammelt."
b) "Du bist nun bereit dich dem großen Gegner im Süden zu stellen."
Im Falle von a muss der Charakter weiter Tränke sammeln, im Falle von b muss der Charakter den starken Gegner finden und ihn besiegen, wobei der Spieler x Erfahrungspunkte extra bekommt.
Quest Level 4 :
Der Questgeber für Quest 4 steht links unten in der Ecke und erteilt den Auftrag: "Rette mein Kind aus den Fängen der Kidnapper, dann wirst du Reichtum erlangen!"
Das Kind wird in einer Baumallee (Mitte des Spielfeldes), welche von beiden Seiten durch Gegner versperrt ist gefangen gehalten. Der Charakter muss gegen den schwächeren Gegner von beiden kämpfen (der andere ist unbesiegbar) und somit das Kind befreien, welches er zum Questgeber zurückbringen muss. (Kind und Gegner müssen für die anderen Spieler respawnen)
Hat der Charakter dies getan sagt der Questgeber: "Danke, Retter meines Kindes! Nun wirst du Reichtum erlangen...Im Nordosten wirst du einen Geheimgang finden, der dich zum Reichtum führt...." (Kollision an dem entsprechenden Baum muss für den Spieler aufgehoben werden)
In der geheimen Höhle, welche durch einen Baum zugänglich ist, muss man den Weg zum Gold finden, welches noch von einem zu besiegenden Gegner bewacht wird. (Auch hier müssen Gold und Gegner respawnen)

1.4.2.1.1 Questannahme

Der Spieler muss eine Quest annehmen, damit er die nächste Aufgabe im Spiel lösen kann und somit seinen Charakter weiterentwickelt und neue Gegenstände finden und nutzen kann. Bei Annahme einer Quest, wird diese ins Questlog geladen und mit „offen“ als noch nicht erfüllt gekennzeichnet.

1.4.2.1.2 Erfüllung

Wenn der Charakter eine Quest erledigt hat, erhält er eine bestimmte Anzahl an Erfahrungspunkten, je nachdem um welche Quest es sich handelt.

Außerdem erhält er eine Belohnung (z.B. einen Trank oder Gold) und die Erfüllung der Quest wird mit „abgeschlossen“ im Questlog verzeichnet.

1.4.2.2 Handelssystem

Regelt den Handel von Gegenständen zwischen Händler und Charakter.

1.4.2.2.1 Waren kaufen

Befindet der Charakter in unmittelbarer Nähe eines Händlers (Anzeige in unterer rechter Ecke des Bildschirms), so kann er den Handel mit der Entertaste starten. Der Handelsbildschirm wird geöffnet und der Charakter kann aus dem Inventar der Händlers per Mausklick oder Steuerung den gewünschten Gegenstand auswählen und dann auf den Kaufen-Button klicken. Hat der Charakter genügend Gold, so wird der Gegenstand in sein Inventar aufgenommen und sein Goldvorrat wird um den Wert des Gegenstandes reduziert.

1.4.2.2.2 Waren verkaufen

Befindet der Charakter in unmittelbarer Nähe eines Händlers (Anzeige in unterer rechter Ecke des Bildschirms), so kann er den Handel mit der Entertaste starten. Der Handelsbildschirm wird geöffnet und der Charakter muss auf den Wechseln-Button klicken, um in den Verkauf Modus zu gelangen. Dann kann er aus seinem Inventar per Mausklick oder Steuerung den gewünschten Gegenstand auswählen und dann auf den Verkaufen-Button klicken. Dann wird der Gegenstand aus seinem Inventar entfernt und ihm das entsprechende Gold dafür gutgeschrieben. Questgegenstände sind unverkäuflich.

1.4.3 Kampfsystem

Hier werden die Features und Regeln, wie ein Kampf zwischen einem Spieler und einem NPC abläuft, beschrieben.

1.4.3.1 Regeln

Der Kampf findet im Hintergrund in Echtzeit statt.

Jede Aktion dauert eine gewisse Zeit. Die Zeit, bis erneut eine Aktion ausgeführt werden kann basiert auf der Gewandtheit der Spielfigur. Den Berechnungen liegen ausschließlich die verschiedenen Attribute zu Grunde. Um die Kämpfe ein wenig zufälliger zu gestalten, liegt außerdem ein Würfelsystem zu Grunde.

Dies soll verhindern, dass man Kämpfe nur dadurch gewinnt, dass man möglichst schnell Angriffe "spammt".

Attribute können in aktiv und passiv eingeteilt werden. Aktive Attribute können durch den Spieler direkt bei Levelaufstieg

gesteigert werden. Passive Attribute sind abhängig von aktiven Attributen oder eventuelle Gegenstands-Boni.
Die Basis eines jeden Attributs beträgt 10 (außer es steht beim jeweiligen Attribut explizit etwas anderes!).
Attribute gehen nicht direkt in die Berechnungen ein, sondern über ihren Modifikator.

Der Modifikator steigert sich jeweils um 1 für jeden 2. Attributpunkt. (Beispiel: Ein Charakter hat eine Basisstärke von 10 auf Level 1. Auf Level 7 besitzt er nun eine Stärke von 17,
vorausgesetzt der Charakter bekommt jedes Leve einen Attributpunkt und verteilt diesen auf das Attribut Stärke.
Der Modifikator beträgt also 7 / 2 = 3. Alle Würfe, in welche nun das Attribut Stärke eingeht, erhalten somit für diesen
Charakter einen Bonus von +3.)

Aktive Attribute der Charaktere:
 * Stärke
 * Gewandheit
 * Ausdauer
 * Intelligenz

Passive Attribute der Charaktere:

* Rüstung
  Errechnet sich aus: //Basiswert (10) + Gewandheitsmodifikator + Rüstungswert der angelegten Rüstung
* Lebenspunkte
  //Basiswert (10) + Ausdauermodifikator//

Ferner können Gegenstände auch Attribute haben, welche allesamt fest für diesen definiert sind. Diese haben
entweder durch anlegen einen direkten Effekt auf die Attribute des Charakters (z.b. Rüstungswert, +Stärke etc.) haben oder
können durch explizite Benutzung des Gegenstands in Kraft treten (Lebenspunkte wiederherstellen ...).

Rüstungen:
 * Rüstungswert

Waffen:
 * Basisschaden (MinSchaden - MaxSchaden ... ein Kurzschwert hätte zb. einen Schaden von 1w4, was bedeutet, dass ein virtueller 4-seitiger Würfel geworfen wird)

Tränke:
 * Wiederherstellung von Lebenspunkten

Kommen wir zum Kampf und den Regeln an sich:
Spieler schlägt zu
 * Trefferberechnung:
   1w20 + Stärkemodifikator >= Rüstungswert des Opfers sein

* Schadensberechnung bei Treffer:
  Basisschaden der Waffe (z.b. 2w6) + Stärkemodifikator

1.4.3.2 Tod des Spielers

Fallen die Lebenspunkte eines Charakters auf Null oder weniger, so stirbt der Charakter. Bei seinem Tod verliert er 5% seines Goldes und respawnt anschließend an dem Punkt, wo er das Spiel begonnen hat.

1.4.3.2.1 Spawnpoints

Definiert Punkte in der Spielwelt, an denen Charaktere in der Spielwelt starten, an denen Charaktere wieder auferstehen, wenn sie gestorben sind oder an denen besiegte NPCs wieder auferstehen.

1.4.3.3 Tod des Gegners

Fallen die Lebenspunkte des NPCs auf Null oder weniger, so stirbt er. Der Spieler hat den NPC besiegt und gewinnt somit Erfahrungspunkte, Gold und einen Gegenstand (dies wird ihm in einem separaten Ausgabefenster beim Kampf) und kann anschließend ganz normal im Spiel fortfahren. Der Gegner hingegen verschwindet für eine gewisse Zeit und respawnt dann wieder an seinem alten Standpunkt.

1.4.3.3.1 Lootsystem

Hat ein Charakter einen NPC besiegt, so erhält er einen zufällig ausgewählten Gegenstand, der vom Gegner hinterlassen wird.

1.4.4 Multiplayer

Umfasst alle Funktionen, die den Mehrspieler-Part des Spiels ausmachen.

1.4.4.1 Client/Server

Die Architektur des Spiels beruht auf einem Client-Server-System. Der Server verwaltet die aktuelle Spielwelt, außerdem die Charaktere. Jeder Charakter erhält eine eigene Datei auf dem Server, in dem seine Werte und sein Inventar gespeichert werden. Meldet sich ein Client beim Server an, so erhält er den ausgewählten Charakter sowie die aktuelle Karte. Der Server ist nur für die Koordination der Spieler auf der Karte sowie die KI zuständig. Die andere Spiellogik ist auf dem Client realisiert, um den Kommunikationsaufwand zwischen Client und Server gering zu halten. Die Graphik- und Soundausgabe wird ebenfalls über den Client gesteuert.

1.4.4.2 Zeitsynchronisation

Die Zeitsynchronisation sorgt dafür, dass alle Spieler, die auf einem Server angemeldet sind, stets die gleiche Spielwelt sehen. Bewegt sich ein Spieler auf einem Ausschnitt, auf dem noch ein anderer Spieler steht, so müssen beide Spieler die Veränderung sehen.

1.4.4.3 Chat

Der Spieler hat während des Spiels die Möglichkeit, ein Chatfenster zu öffnen, in dem er Texte eingeben kann. Diese Texte sind für alle Spieler, die das Chatfenster geöffnet haben und beim selben Server angemeldet sind, sichtbar. Vor dem Text wird der Name des Charakters angezeigt, der diesen Text eingegeben hat.

1.4.5 Levelsystem

Das Levelsystem umfasst alle Regeln, welche die Entwicklung eines Charakters steuern.

1.4.5.1 Maximallevel

Für unser Spiel wird festgelegt, dass maximal 10 Level vorhanden sind. Im 10. Level kann man nicht mehr weiter aufsteigen.

1.4.5.2 Levelaufstieg

Um ein Level aufzusteigen, muss man eine bestimmte Anzahl an Erfahrungspunkten sammeln. Im ersten Level muss man 100 Erfahrungspunkte sammeln, im zweiten Level 150, im dritten Level 250, im vierten Level 400, im fünften Level 700, im sechsten Level 1200, im siebten Level 1800, im achten Level 2500, im neunten Level 3500 und im letzen 5000.
Übersicht:
Level 1: 0-100
Level 2: 101-250
Level 3: 251-500
Level 4: 501-900
Level 5: 901-1600
Level 6: 1601-2800
Level 7: 2801-4600
Level 8: 4601-7100
Level 9: 7101-10600
Level 10: 10601-15600
Man erhält eine bestimmte Anzahl an Erfahrungspunkten, die der Stärke des Gegners entspricht, wenn man diesen besiegt und eine bestimmt Anzahl der Erfahrungspunkte erhält man bei Erfüllung des Quests.
Bei einem Levelaufstieg erhält man automatisch eine höhere Anzahl von Lebenspunkten: im ersten Level hat man als Ausgangsposition 60 von 60 Lebenspunkten, im zweiten Level 70 von 70, im dritten Level 80 von 80, im vierten Level 90 von 90, im fünften Level 100 von 100, im sechsten Level 110 von 110, im siebten Level 120 von 120, im achten Level 130 von 130, im neunten Level 140 von 140 und im zehnten Level 150 von 150.
Des Weiteren erhält man beim Levelaufstieg jeweils 1 Punkt, den man dann einem seiner Attribute zuordnen kann. Die anderen Attributwerte werden intern entsprechend erhöht.

1.4.6 Charakter

Alle Features, die den Charakter betreffen.

1.4.6.1 Inventar

Hier findet die komplette Inventarverwaltung eines Charakters statt.
Ein Charakter kann 30 Gegenständen mit sich führen.
Darunter fallen:
Waffen (Wurfmesser, Zauberstab) --> zum angreifen und verteidigen
Heilmittel (kleine Tränke, große Tränke) --> um die Lebenspunkte wieder aufzufrischen
Gegenstände, die zur Erfüllung einer Quest dienen(Schlüssel, Brücke, Kind)

1.4.6.2 Attribute

An dieser Stelle werden sämtliche Charakterattribute gespeichert, deren Ausmaß in Punkten gezählt wird.
Je höher die Anzahl der Punkte, desto besser das Attribut bzw. die Eigenschaft.
Die zu unterscheidenden Attribute lauten:
Stärke
Ausdauer
Gewandtheit
Intelligenz

Es gibt auch noch passive Attribute, die aus den oben genannten, aktiven Attributen berechnet werden:
Lebenspunkte
Rüstung

1.4.6.3 Level

Hier wird gespeichert, in welchem der 10 Level sich der Charakter aktuell befindet.
Level 1-5 spielen jeweils im Himmel oder in der Hölle, je nachdem ob der Charakter ein Engel oder ein Dämon ist.
Level 6-10 spielen dann auf der Erde, wo auch Engel und Dämonen aufeinandertreffen können.

1.4.6.3.1 Erfahrungspunkte

Dies sind Punkte, deren Erhöhung notwendig ist um ein Level aufzusteigen.
Man bekommt (je nach der Stärke des Gegners) eine gewissen Menge an Erfahrungspunkten, wenn man einen NPC oder einen anderen Charakter besiegt. Außerdem erhält man auch Erfahrungspunkte, wenn man eine Quest erfolgreich erfüllt.

1.4.6.4 Gold

Ein Charakter besitzt eine gewisse Menge an Gold, welches im Spiel als einzige Währung dient. Mit seinem Gold kann der Spieler Gegenstände(z. B. Tränke) kaufen.

1.4.6.5 Questlog

Jeder Charakter besitzt ein Questlog. Sobald der Charakter eine neue Quest erhält, wird diese im Questlog eingetragen und mit „offen“ (noch nicht erfüllt) markiert. Erfüllt der Charakter die Aufgabe erfolgreich, so wird die Quest im Questlog mit „abgeschossen“ als bestanden markiert.

1.4.6.6 Äußeres

Jeder Charakter hat ein entsprechendes Aussehen, wobei folgende graphische Repräsentationen vorhanden sind:
Engel→ Magier/Krieger
Damön→ Magier/Krieger
Gegner

1.4.7 Musik

Alle Features, welche die Spielmusik beeinflussen.

1.4.7.1 ein

Schaltet die Spielmusik ein.

1.4.7.2 aus

Schaltet die Spielmusik aus.

1.4.8 KI

Die Künstliche Intelligenz dient dazu, sämtliche NPCs im Spiel zu steuern.

1.4.9 Gegenstand benutzen

Um einen Gegenstand aus dem Inventar zu benutzen, klickt man diesen Gegenstand an und klickt den Benutzen-Button an.

1.2 Szenarien

1.2.1 Server starten und stoppen

Bevor sich ein Client an einem Server anwählen kann, muss der Server gestartet werden. Dazu wird auf dem Server das Java-Programm gestartet. Startet der Server zum ersten Mal, so erstellt er eine neue Spielwelt. Nun haben Clients die Möglichkeit, sich an diesem Server anzumelden.
Der Server kann gestoppt werden. Dazu wird der aktuelle Stand der Spielwelt in eine Datei gespeichert, anschließend werden alle Clients abgemeldet. Im Anschluss wird das Java-Programm beendet. Beim nächsten Start des Servers wird der gespeicherte Stand der Spielwelt wieder hergestellt. Die Clients müssen sich manuell neu am Server anmelden.

1.2.2 Spieler startet Spiel

Der Spieler startet ein Spiel, in dem er die Datei Client.java ausführt. Voraussetzung ist, das ein Server bereits gestartet ist, an dem er sich anmelden kann. Daraufhin erscheint ein neues Menüfenster, mit Login-Einheit mit Bestätigungsbutton und ein Charaktererstell-Button. Hat man Benutzername und Passwort eingegeben und drückt den Bestätigungsbutton, betritt der Spieler die Spielwelt (siehe Szenario 4), mit dem Charakter den er früher schon einmal erstellt hat.

1.2.3 Charakter erstellen

Der Spieler startet das Spiel zum ersten Mal. Ein GUI (1.1.2.4) erscheint, in dem der Spieler zwischen verschiedenen Menüpunkten wählen kann. Er / Sie klickt auf den Button “Charakter erstellen” (1.4.1) und ein neues Fenster “Charaktererstellung” öffnet sich. Nun wird der Spieler aufgefordert, seinem zukünftigen Charakter (1.4.6) einen Namen (1.4.1.1) zu geben. Als nächstes muss der Spieler eine Rasse (1.4.1.2) (Engel, Dämon) und eine Klasse (1.4.1.3) (Krieger, Magier) wählen. Die einzelnen Klassen unterscheiden sich in der Stärke von einzelnen Attributen (1.4.6.2.) Anschließend hat der Spieler die Möglichkeit einen Attributpunkt zu vergeben. Damit ist die Erstellung des Charakters abgeschlossen und der Charakter betritt die Welt (siehe Szenario4).

1.2.4 Charakter betritt die Welt

Nach dem erfolgreichen Login (siehe Szenario2) lädt der Client vom Server (1.4.4.1) den Charakter des Spielers. Das beinhaltet die Rasse des Charakters, seine Attribute und Inventar und seine letzte Position. Außerdem erhält der Client die aktuelle Spielfeldbelegung. Nun wird die Spielwelt im Client geladen und der Charakter erscheint an seiner zuletzt gespeicherten Position. Auch die Charaktere und NPCs im Umfeld werden dargestellt. Der Server führt dazu eine Synchronisation mit den angemeldeten Clients aus.

1.2.5 Charakter erkundet die Spielwelt

Der Spieler bewegt seinen Charakter (1.1.2.1) frei über die Spiellandschaft (1.1.2.2.) Die Steuerung (1.2) des Charakters erfolgt über die Pfeiltasten der Tastatur. Bei jeder Bewegung erfolgt ein Datenaustausch zwischen Client und Server (1.4.4.1, 1.4.4.2), indem der Client seine aktuelle Position und die gewünschte Richtung zum Server überträgt. Dieser ermittelt durch eine Kollisionsüberprüfung ob die auszuführende Bewegung legal ist oder ob das Gelände unpassierbar ist.

1.2.6 Charakter kämpft gegen einen NPC

Charakter (1.1.2.1) trifft auf einen NPC, d.h. sie stehen auf 2 benachbarten Feldern.
Der Spieler klickt auf den NPC auf dem Spielfeld und wählt aus dem Menü «Kämpfen».

Der Kampfbildschirm wird geöffnet. Der Spieler hat 4 Möglichkeiten Aktionen auszuwählen.
1. Angriff, der Spieler greift den Gegner an
2. Spezialangriff, je nach Rasse der Spieler wird entweder der Schaden verdoppelt (Teufel) oder der abgezogene Schaden zu den eigenen Lebenspunkten addiert (Engel)
3. kleinen Heiltrank benutzen, wenn der Spieler im Besitz von kleinen Heiltränken ist, werden seine HP um 20 erhöht und ein kleiner Heiltrank verschwindet aus dem Inventar
4. großen Heiltrank benutzen, siehe kleiner Heiltrank bloß die HP werden um 50 erhöht
Entweder der Charakter gewinnt (siehe Szenario7) oder verliert den Kampf gegen den NPC (siehe Szenario8).

1.2.7 Charakter gewinnt Kampf gegen NPC

Gewinnt der Charakter im Kampf gegen den NPC, so bedeutet dies den Tod des Gegners (1.4.3.3). Es erfolgt ein Soundeffekt (1.3.2.1.1). Die Erfahrungspunkte (1.4.6.3.1) des Charakters steigen und es erscheint ein Fenster mit dem Gold, den Erfahrungspunkten und dem Gegenstand das der Spieler bekommt. der Gegner wird für eine geringe Zeit von dem Spiel entfernt, aber dieser kommt dann nach einer gewissen Zeit wieder aufs Spielfeld.

1.2.8 Charakter verliert Kampf gegen NPC

Sinken während eines Kampfes die Erfahrungspunkte (1.4.6.2) des Charakters auf Null, so bedeutet dies den Tod des Charakters (1.4.3.2.). Es ertönt ein spezieller Ton (1.3.2.1.1), der dieses Ereignis anzeigt und das Kampffenster schließt sich. Dem Charakter wird 5% seines Goldvorrats (1.4.6.4) abgezogen und er respawnt am Spawnpoint (1.4.3.2.1), dass heißt er startet mit vollen Lebenspunkte und kann nun erneut versuchen den NPC zu besiegen. Befindet man sich in unmittelbarer Nähe zu einem Gegner, so kann der Spieler durch drücken der Enter-Taste den Kampf starten. (Infoanzeige erscheint auch im unteren, rechten Bildschirm)

1.2.9 Charakter erhält eine Quest

Der Charakter (1.1.2.1) bewegt sich durch die Spielwelt (1.1.2.2). Er trifft auf einen Questgeber (1.4.1.3). Der Spieler hat nun die Möglichkeit den Questgeber anzusprechen. So teilt der Questgeber ihm seine Quest (1.3.2.2) mit. Nimmt der Spiel der die Quest an (1.3.2.2.1), so wird diese ins Questlog (1.1.2.3) geladen und als “offen” markiert.

1.2.10 Charakter besteht eine Quest

Der Charakter (1.1.2.1) hat eine “offen”e Quest (1.3.2.2) in seinem Questlog. Nach dem erfolgreichen Bestehen (1.4.2.1.2) der Aufgabe ertönt ein Signalton (1.3.2.2.2) und der Charakter (1.1.2.1) erhält eine Belohnung. Dies kann ein besonderer Gegenstand (1.1.2.3) sein, oder ein neuer sichtbarer Bereich auf der Spielwelt (1.1.2.2). Die Quest (1.3.2.2) ist in seinem Questlog (1.4.6.5) nun als “abgeschlossen” markiert.

1.2.11 Charakter kauft/verkauft Gegenstände bei einem Händler

Der Charakter (1.1.2.1) bewegt sich durch die Spielwelt (1.1.2.2). Er trifft auf einen Händler (1.4.1.3). Der Spieler startet mit der Enter-Taste den Handel. Er wählt (1.2.3.2) aus der Inventarliste (1.4.6.1) des Händlers einen Gegenstand (1.1.2.3). Hat der Charakter genug Gold (1.4.6.4)wird dieser in die Inventarliste des Charakters verschoben und dem Charakter wird Gold entsprechend dem Wert des Gegenstandes abgezogen (1.4.2.2.1). Ebenso kann der Spieler Waren auch wieder verkaufen, wofür ihm dann wieder Gold gutgeschrieben wird. (1.4.2.2.2).

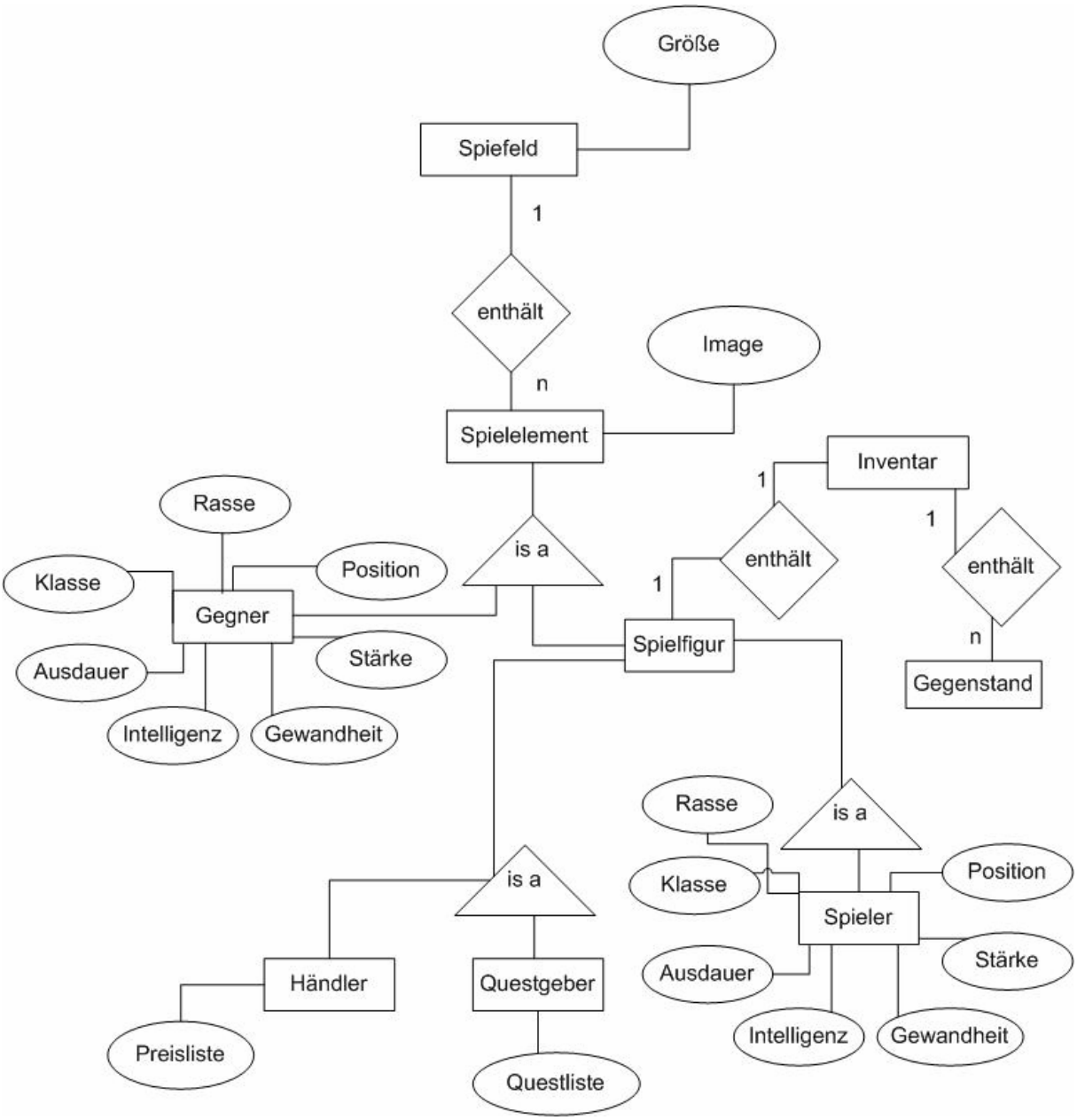
1.2.12 Charakter steigt einen Level auf

Hat ein Charakter eine bestimmte Anzahl an Erfahrungspunkten (1.4.6.3.1) gesammelt, so steigt er ein Level (1.4.6.3) im Spiel (1) auf (1.4.5.2), wobei ein Levelaufstiegssound (1.3.2.3) ertönt. Nun hat der Spieler die Möglichkeit, die Attribute (1.4.6.2) seines Charakters zu erhöhen. Ein Aufstieg ist jedoch nur bis zu einem bestimmten Maximallevel (1.4.5.1) möglich.

1.2.13 Charakterübersicht

Der Spieler drückt die Taste c, wodurch das Menü Charakterübersicht geöffnet wird, welches dem Spieler nützliche Informationen über seinen Charakter bietet. Diese Übersicht zeigt eine Tabelle in der die Attribute (1.4.6.2) mit ihren jeweiligen aktuellen Werten eingetragen sind. In einer darauffolgenden Zeile steht das aus den Erfahrungspunkten (1.4.6.3.1) errechnete Level (1.4.6.3), in dem sich der Charakter des Spielers gerade befindet. Außerdem gibt es eine weitere Zeile in der Übersicht, welche anzeigt, über wie viel Gold (1.4.6.4) der Charakter verfügt. Um die Charakterübersicht zu verlassen, drückt der Spieler die Taste c erneut.

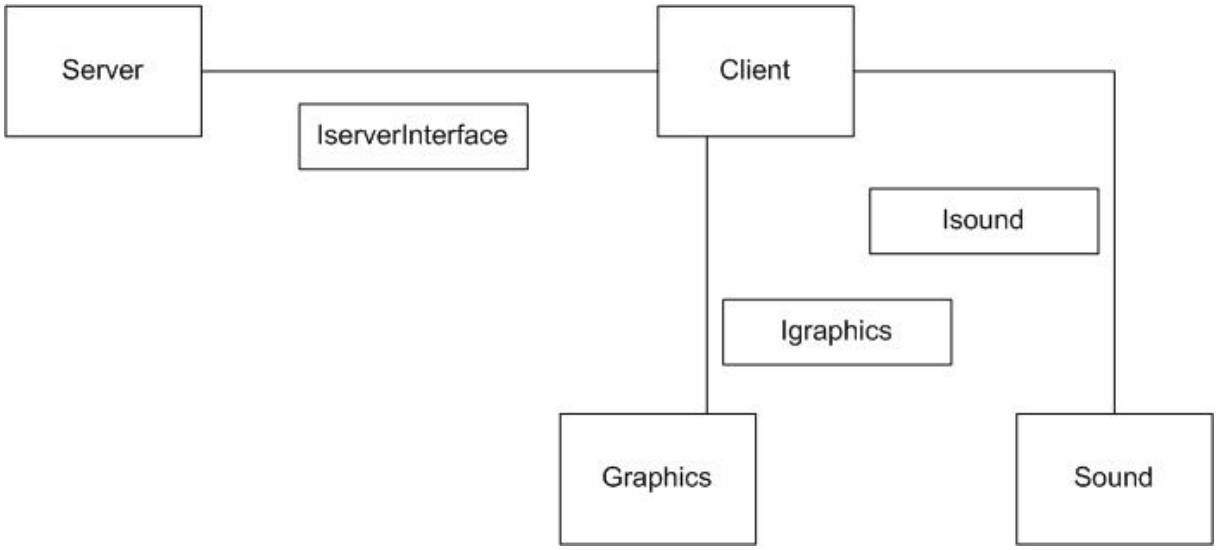
1.3 Entity-Relationship Diagramm



Auf einem Spielfeld variabler Größe befinden sich n Spielelemente. Ein Spielelement enthält außer seiner graphischen Repräsentation keine weiteren Daten.
Von den Spielelementen werden die Entitäten Gegner und Spielfigur abgeleitet. Alle feindlichen Nicht-Spieler-Charaktere sind Gegner-Entitäten mit den im Diagramm dargestellten Attributen.
Eine Spielfigur enthält ein Inventar, dass wiederum n Gegenstände enthalten kann.
Von Spielfigur werden die drei Entitäten Händler (mit einer Preisliste), Questgeber (mit einer Questliste) und Spieler abgeleitet. In Spieler sind alle für die vom Benutzer gesteuerte Figur relevanten Attribute hinterlegt. Dazu zählen neben Rasse und Klasse auch die Gewandtheit, die Intelligenz, die Ausdauer, die Stärke oder die Position.

2.0 Architektur

2.1 Komponentendiagramm



1. Server:

Komponente besteht aus den Klassen Server, Karte, Spieler, Gegner und der Menge der Dateien, die benötigt werden, um die einzelnen Kartenausschnitte und die Daten der Spieler zu speichern.

→ zuständig für die Gegnerbewegung die Verarbeitung und Speicherung der Spielerdaten und die eigentliche Spielerbewegung(Kollisionsabfragen)
→ stellt der Komponente Client verschiedene Funktionalitäten zu Verfügung (übertragen des aktuellen Kartenausschnittes, speichern und übertragen der Spielerdaten, Kollisionsabfragen...)

2. Client:

Komponente besteht aus den Klassen Client, Spieler, Gegner, Kampf, Kampfbildschirm, Chat, Inventar, Händler, Spielelement, Spielfigur, Quest, Questgeber, Questliste,... und Spielfigur.

→ Schnittstelle zwischen Spieler und Spiel.
→ zuständig für den Kampf, die Charaktere(Spielercharakter, Händler, Questgeber), sonstige Abfragen, die das Spielgeschehen beeinflussen.
→ bezieht alle grafischen Elemente und Benutzereingaben aus der Komponente Graphics.

3. Graphics:

Komponente besteht aus den Klassen Graphics, Fensterausschnitt, Fensterhandel, FensterInventar, FensterCharakter, FensterKarte, FensterDialog, FensterQuestlog und FensterMenu.

→ stellt der Komponente Client alle grafischen Elemente zu Verfügung, verarbeitet Benutzereingaben und gibt sie an Client weiter.

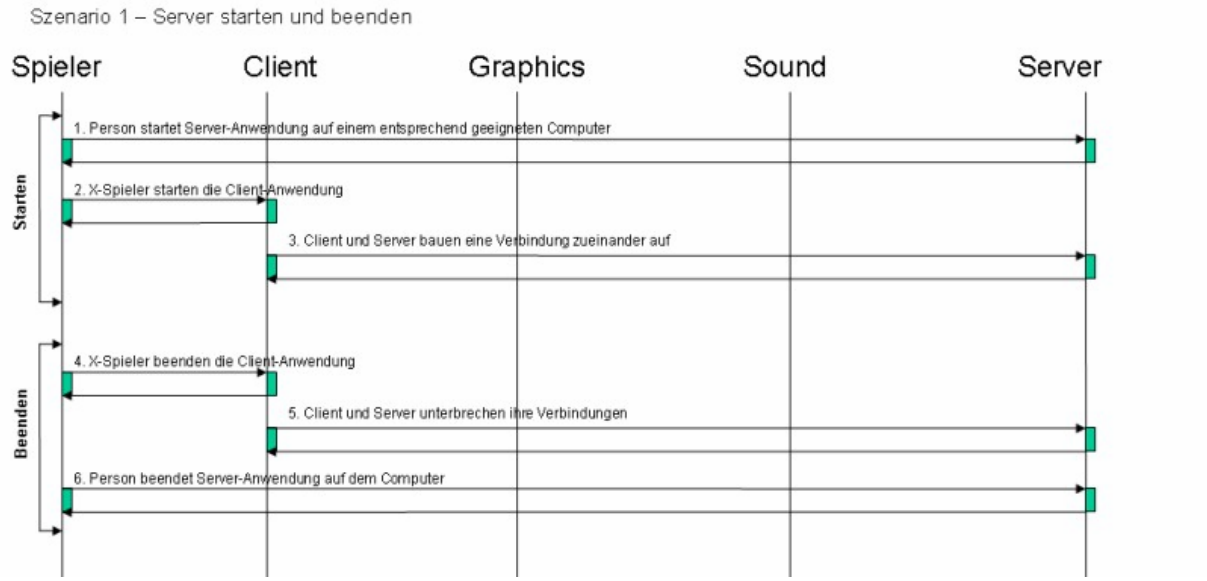
4. Sounds:

Komponente besteht aus der Klasse Sound.

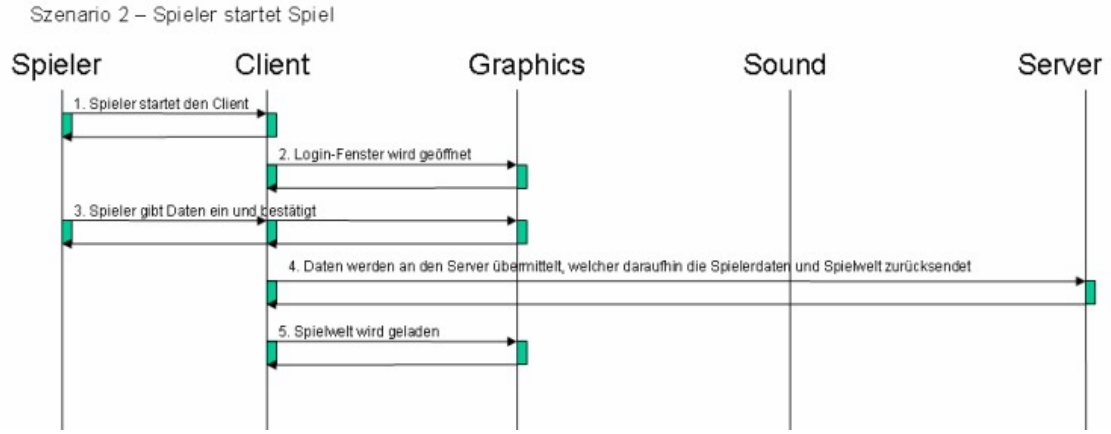
→ Stellt der Komponente Client alle benötigten Sounds zu Verfügung(und verwaltet diese)

2.2 Sequenzdiagramm

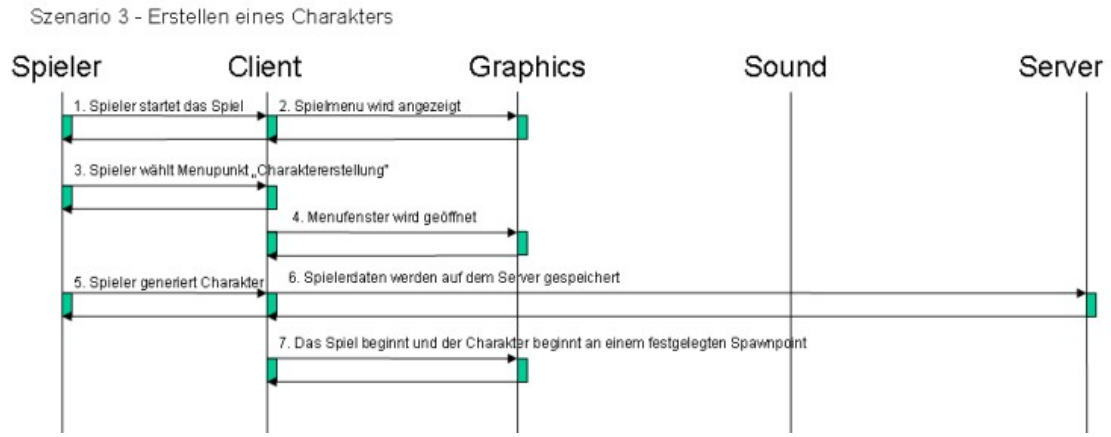
2.2.1 Szenario 1: Server starten und beenden



2.2.2 Szenario 2: Spieler startet Spiel

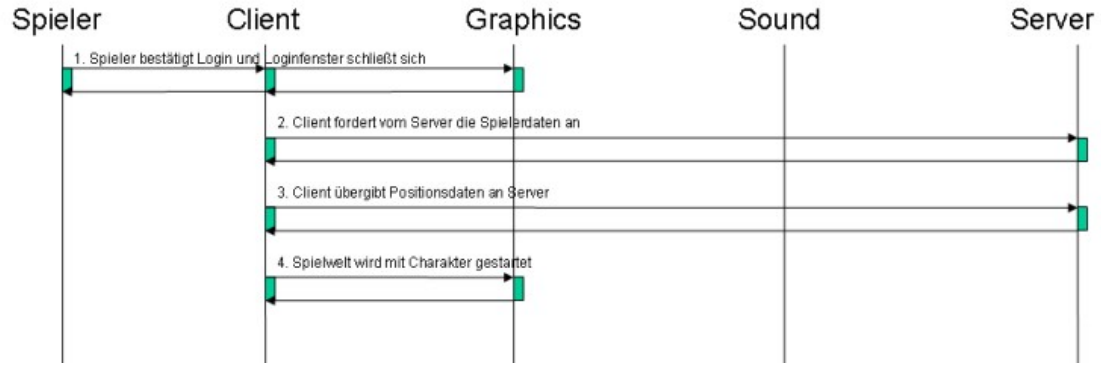


2.2.3 Szenario 3: Charakter erstellen



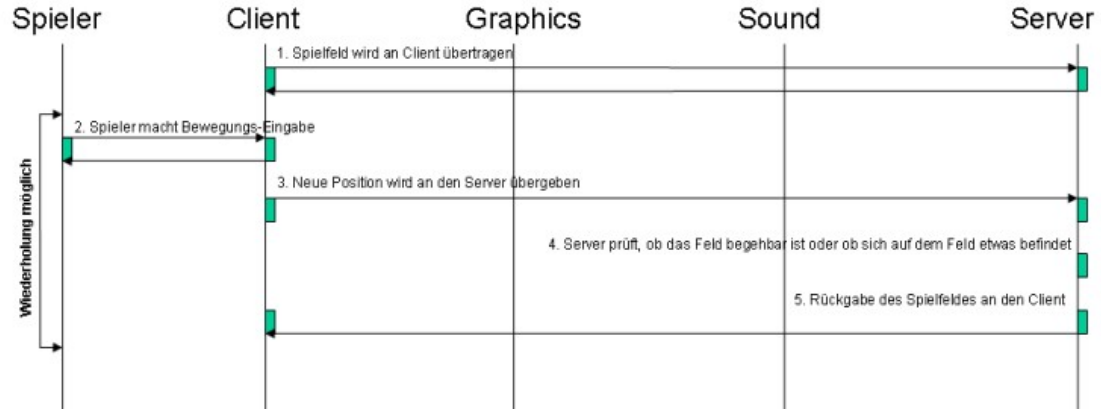
2.2.4 Szenario 4: Charakter betritt die Welt

Szenario 4 - Charakter betritt die Welt



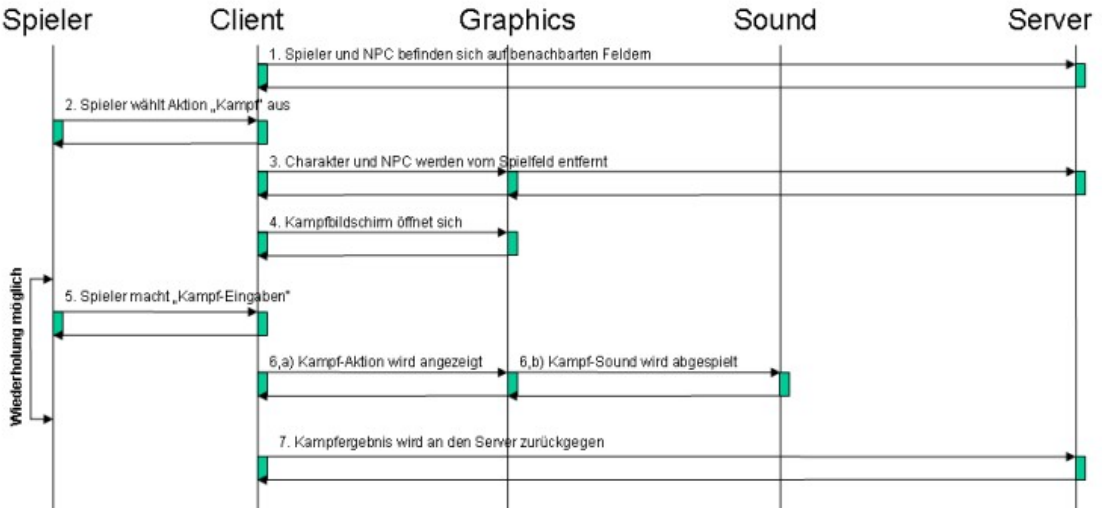
2.2.5 Szenario 5: Charakter erkundet die Spielwelt

Szenario 5 – Charakter erkundet die Spielwelt



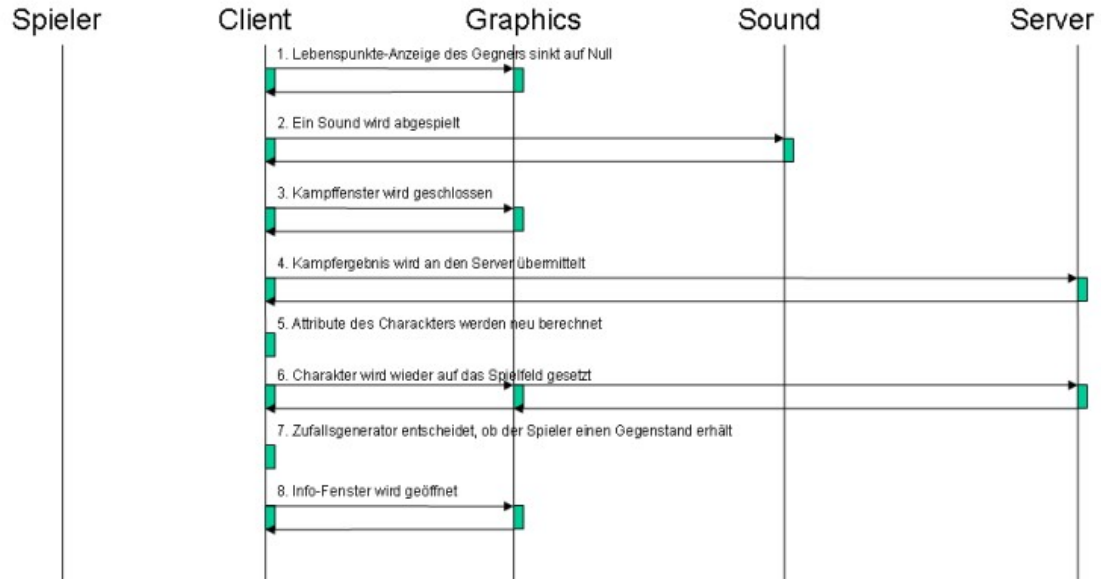
2.2.6 Szenario 6: Charakter kämpft gegen NPC

Szenario 6 – Charakter kämpft gegen NPC



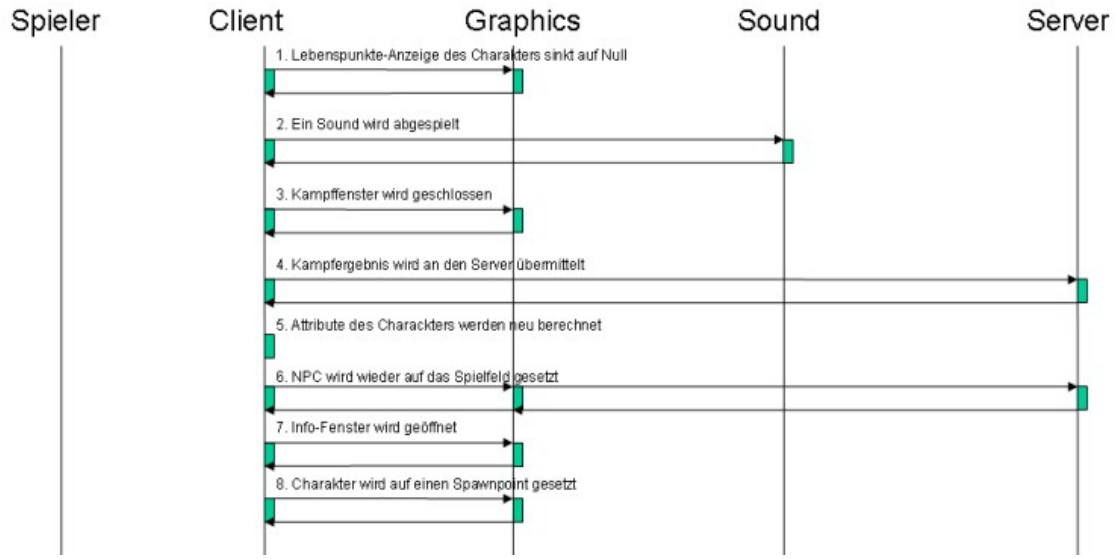
2.2.7 Szenario 7: Charakter gewinnt Kampf gegen NPC

Szenario 7 - Charakter gewinnt Kampf gegen NPC

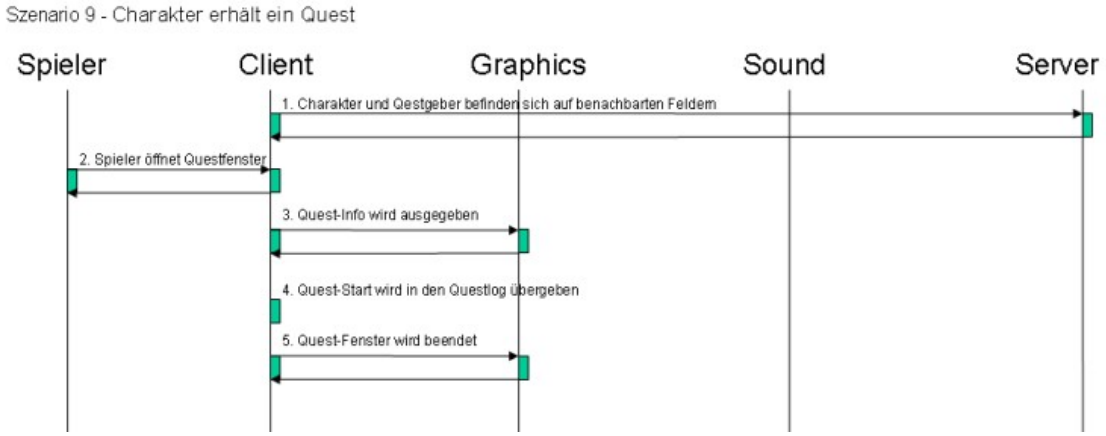


2.2.8 Szenario 8: Charakter verliert Kampf gegen NPC

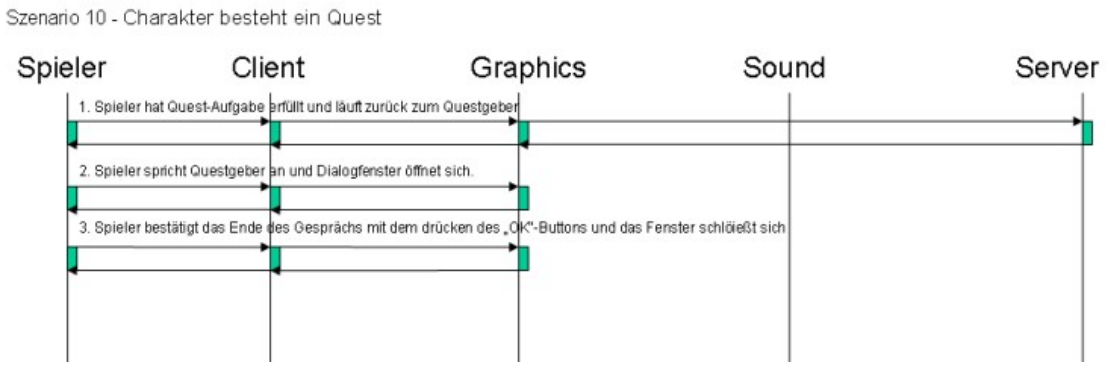
Szenario 8 - Charakter verliert Kampf gegen NPC



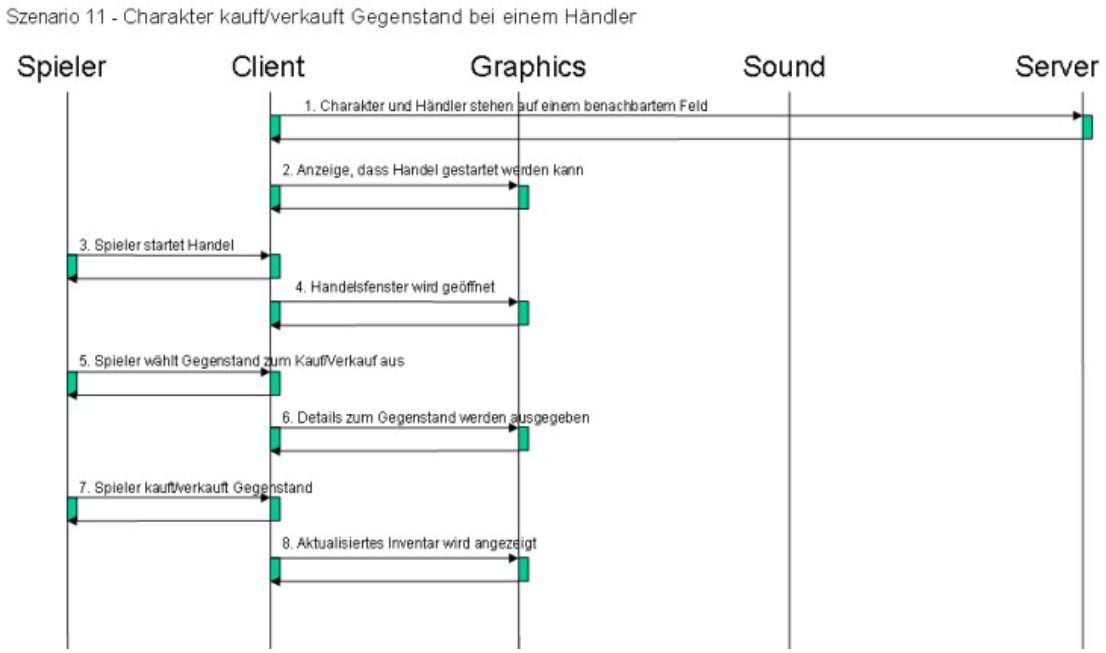
2.2.9 Szenario 9: Charakter erhält eine Quest



2.2.10 Szenario 10: Charakter besteht eine Quest

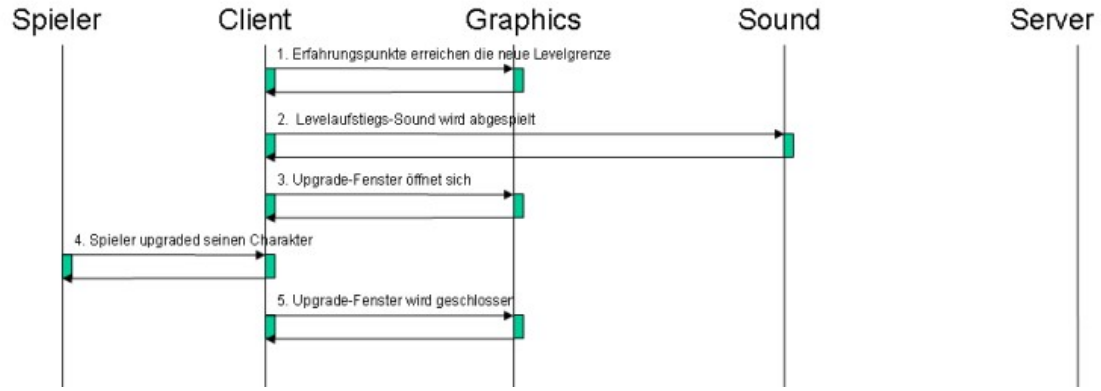


2.2.11 Szenario 11: Charakter kauft/verkauft Gegenstand



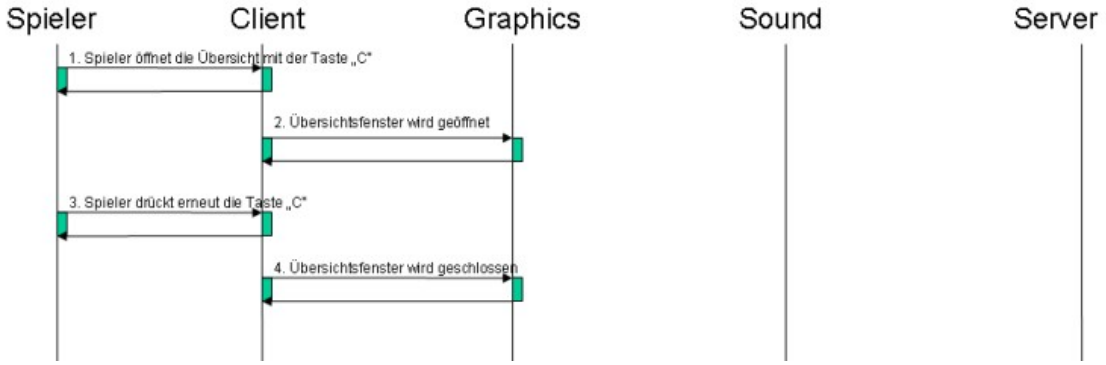
2.2.12 Szenario 12: Charakter steigt ein Level auf

Szenario 12 – Charakter steigt ein Level auf

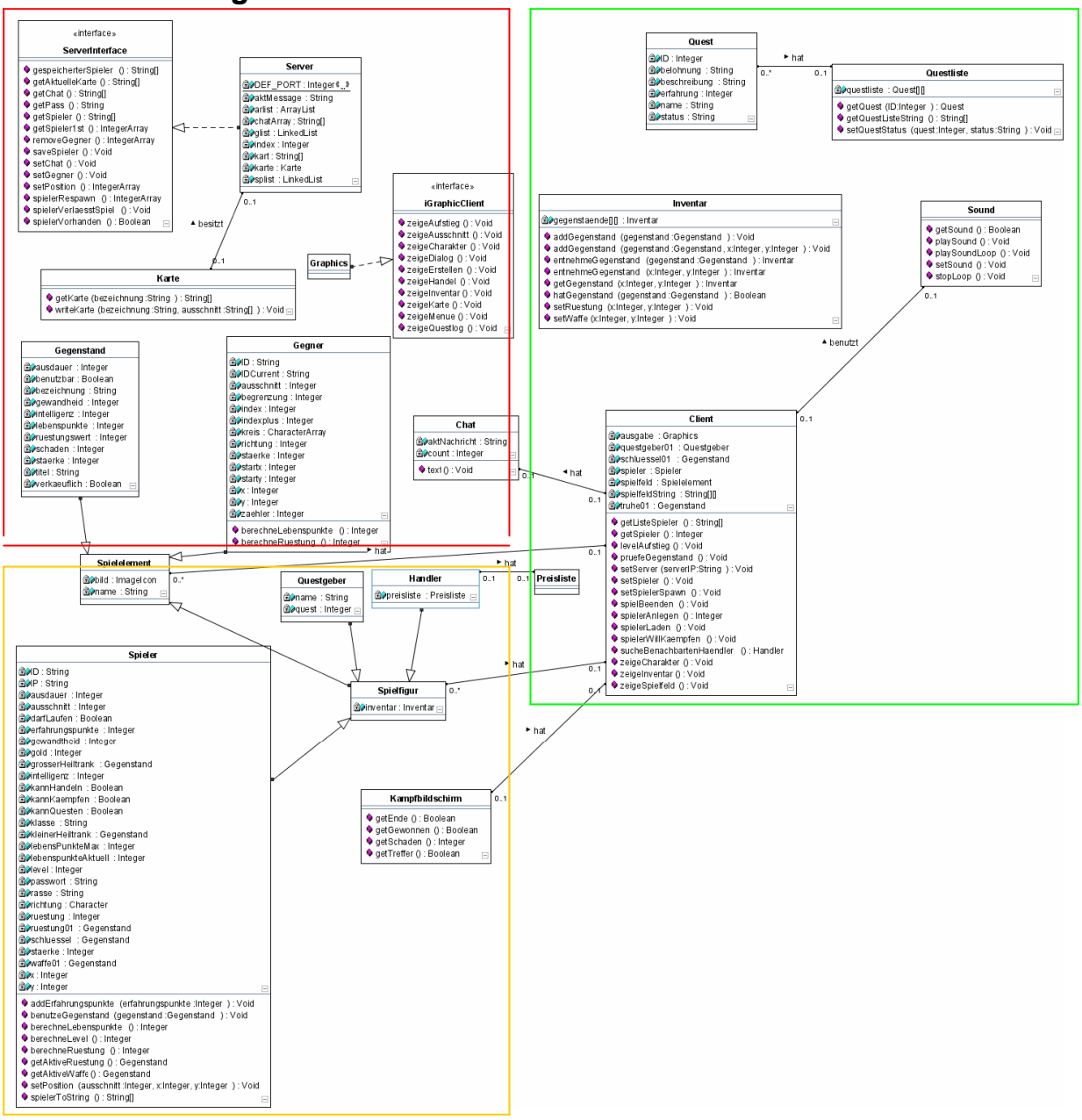


2.2.13 Szenario 13: Charakterübersicht

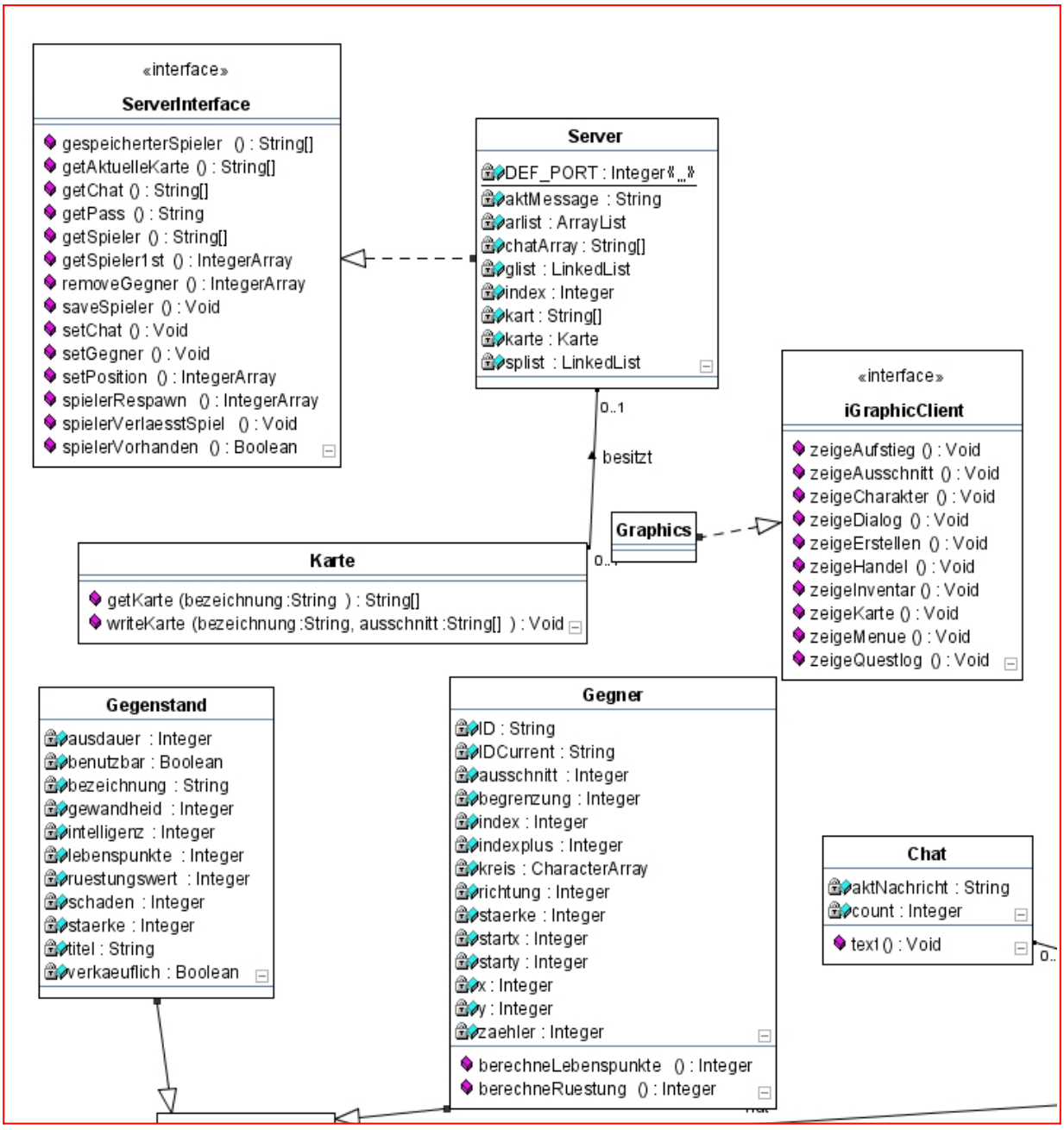
Szenario 13 – Übersicht des eigenen Charakters



2.3 Klassendiagramm



Obiges Klassendiagramm stellt in vereinfachter Form die Architektur unseres Spiels da.
Es tauchen nicht sämtliche Attribute, Methoden oder Klassen auf.
Vor allem triviale Get- und Set-Methoden, Konstruktoren oder innere Klassen wurden weggelassen um das Diagramm übersichtlicher zu gestalten.
Es folgt eine kurze Beschreibung der wichtigsten Klassen und deren Zweck:



Klasse *Server*

Die Klasse Server sorgt für die Synchronisation der Clients sowie der Verwaltung der einzelnen Spieleraccounts. Dies umfasst z.B. die Datenspeicherung bei der Charaktererstellung
oder beim ausloggen eines Clients. Ferner erfolgt die Bewegung sämtlicher Gegner sowie die Verteilung der Chatnachrichten auf dem Server.

Klasse *Karte*

Die Klasse Karte stellt Verwaltungsinstrumente für alle im Spiel vorkommenden Karten zur Verfügung, z.B. das Auslesen der Karten aus Textdateien und übertragen in entsprechende
Arrays, aus der später die Welt aufgebaut wird.

Klasse *Gegner*

Erweitert Spielelemente um Attribute und Funktionen, die einen im Spiel vorkommenden Gegner ausmachen, z.b. seine Charakterattribute oder seine Laufwege.

Klasse *Graphics*

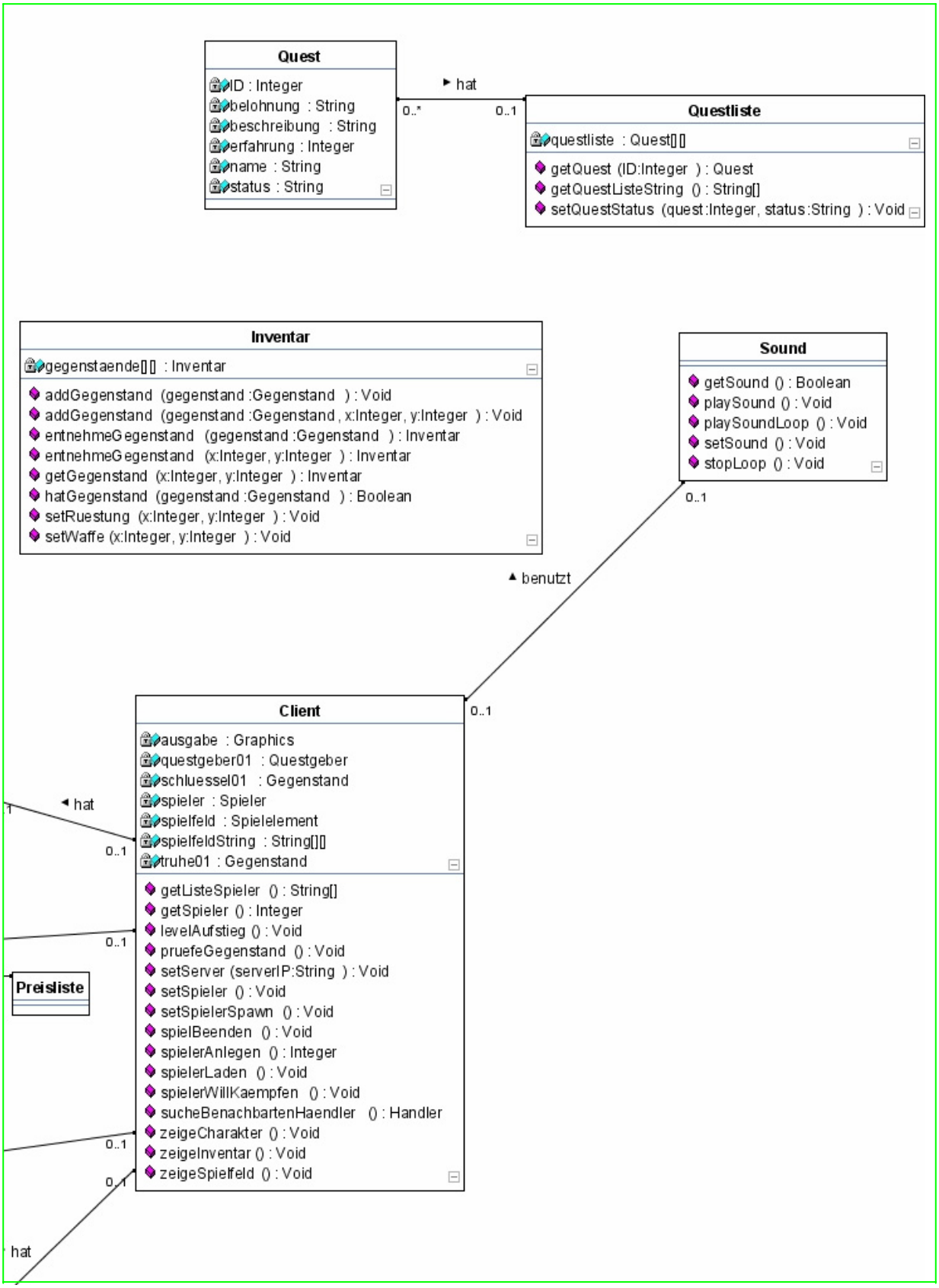
Wird im Client instantiiert. Beinhaltet Klassen für die meisten im Spiel vorkommende Fenster (Dialoge, Questlogs, Charaktererstellung, ...)
sowie den Großteil ihrer Funktionalität.

Klasse *Chat*

Stellt einen globalen Chat bereit, der es ermöglicht den Spielern unabhängig von ihrer aktuellen Position in der Welt miteinander zu kommunizieren.

Klasse *Gegenstand*

Erweitert Spielelemente um alle Attribute, die im Spiel vorkommende Gegenstände besitzen, z.B. ein Heiltrank.



Klasse *Client*

Die Klasse Client ist das zentrale Element auf Spielerseite.

Die Klasse sorgt für den Verbindungsaufbau zum Server, die Verwaltung der Spielfigur sowie allen Gegenständen und NPCs (mit Ausnahme der Gegner).

Der Großteil der anderen Klassen wir über die Klasse Client instantiiert.

Klasse *Sound*

Stellt sämtliche Methoden bereit, um im Spiel Soundeffekte sowie Musik abzuspielen.

Klasse *Quest*

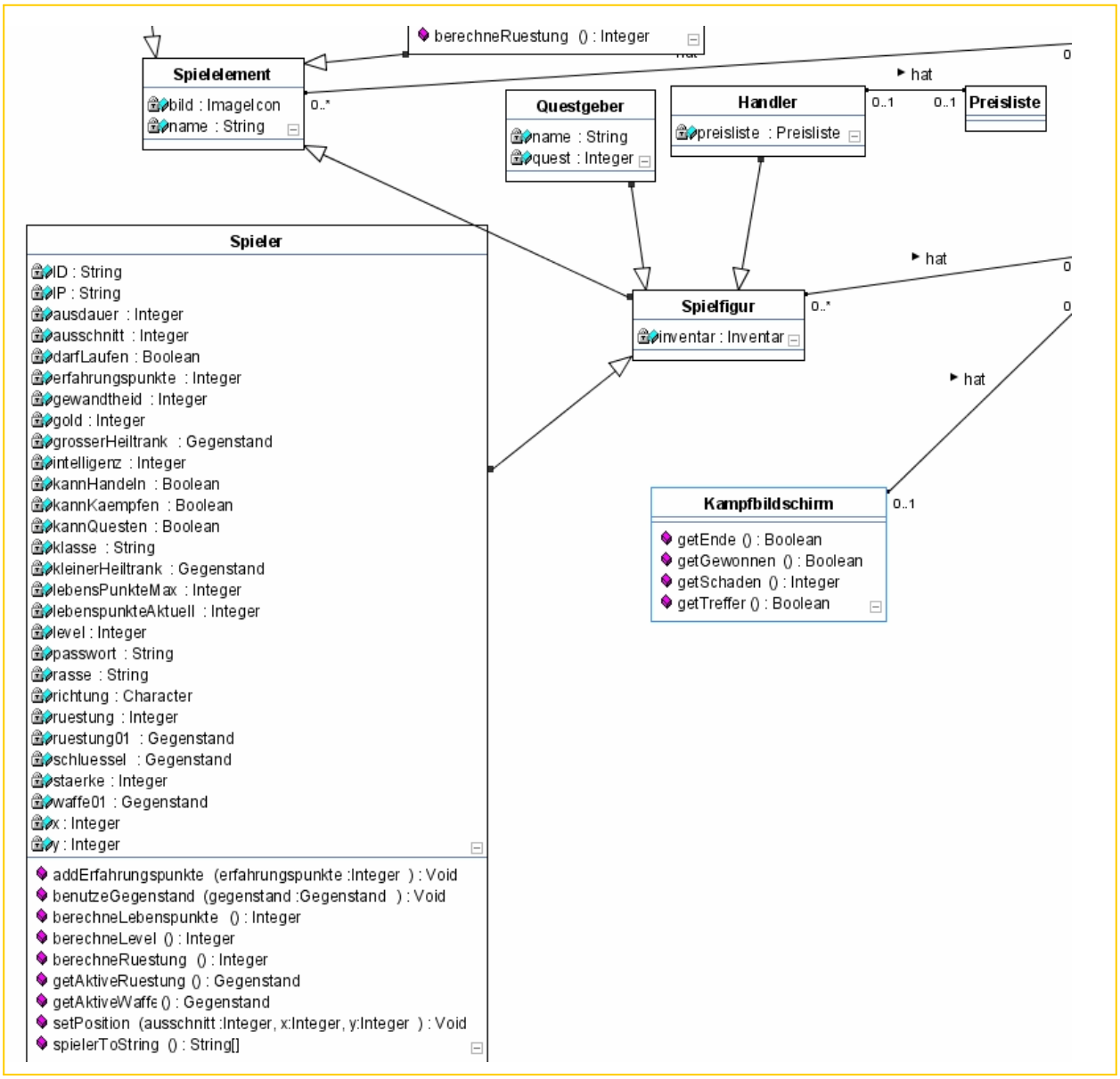
Beinhaltet Informationen zu den im Spiel vorkommenden Quests (z.B. deren Beschreibung, die Belohnung etc.)

Klasse *Questliste*

Wird benutzt um angenommene Quests eines Spielers zu speichern und zu verwalten. (Statusänderung bei Abschluss der Quest etc.)

Klasse *Inventar*

Speichert ein mehrdimensionales Array von Gegenständen und stellt Methoden zur Verwaltung bereit, z.B. Gegenstände dem Inventar hinzufügen oder entnehmen.



Klasse *Kampfbildschirm*

Stellt das Kampfsystem bereit: Dies umfasst das Kampffenster sowie dessen Funktionalität inkl. Implementation der Kampfregeln.

Klasse *Spielelement*

Die Klasse Spielelement ist der Ausgangspunkt für alle Objekte, welche in der Spielwelt visuell dargestellt werde müssen und eine gewisse Funktionalität bieten.

Klasse *Spielfigur*

Erweitert die Klasse Spielfigur um ein Objekt der Klasse Inventar und stellt somit die Basis für alle Charaktere im Spiel, mit Ausnahme der Gegner, welche direkt von der Klasse Spielelement abgeleitet werden.

Klasse *Questgeber*

Erweitert Spielfigur um Attribute und Funktionen, die einen im Spiel vorkommenden Questgeber ausmachen, z.b. seinen Namen oder die von ihm angebotene Quest.

Klasse *Haendler*

Erweitert Spielfigur um Attribute und Funktionen, die einen im Spiel vorkommenden Händler ausmachen, z.b. die Preisliste für alle im Spiel vorkommenden Gegenstände.

Klasse *Preisliste*

Wird ausschließlich von im Spiel vorkommenden Händlern benutzt und speichert die Preise zu den Gegenständen, die ein Händler an- und verkauft.

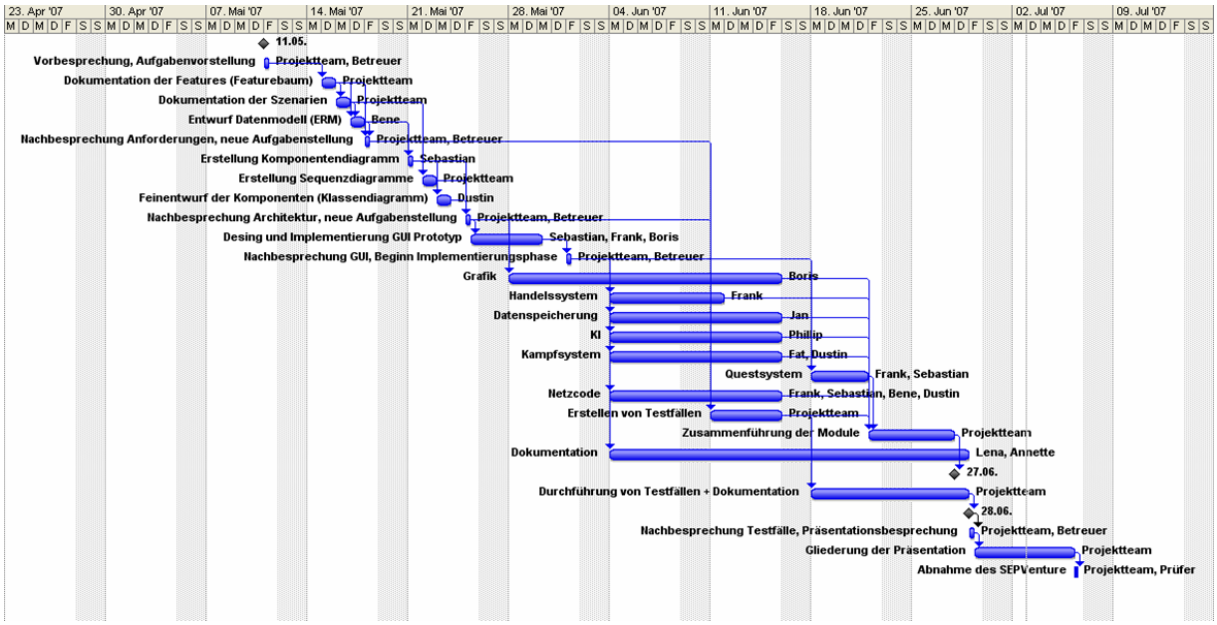
Klasse *Spieler*

Stellt alle Attribute und Methoden zur Verfügung, die ein Spieler benötigt, z.B. seine Charakterattribute, seine Erfahrungspunkte oder auch Verwaltungsinformationen wie seinen Accountnamen oder das Passwort.

3.0 GUI-Prototyp

4.0 Projektplan

	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressourcennamen
1	**Projektstart SEPVenture**	0 Std.	Fr 11.05.07	Fr 11.05.07		
2	*Vorbesprechung, Aufgabenvorstellung*	2 Std.	Fr 11.05.07	Fr 11.05.07		Projektteam, Betreuer
3	Dokumentation der Features (Featurebaum)	1 Tag	Di 15.05.07	Di 15.05.07	2	Projektteam
4	Dokumentation der Szenarien	1 Tag	Mi 16.05.07	Mi 16.05.07	3	Projektteam
5	Entwurf Datenmodell (ERM)	1 Tag	Do 17.05.07	Do 17.05.07	3;4	Bene
6	*Nachbesprechung Anforderungen, neue Aufgabenstellung*	2 Std.	Fr 18.05.07	Fr 18.05.07	3;4;5	Projektteam, Betreuer
7	Erstellung Komponentendiagramm	1 Std.	Mo 21.05.07	Mo 21.05.07	5	Sebastian
8	Erstellung Sequenzdiagramme	1 Tag	Di 22.05.07	Di 22.05.07	7;4	Projektteam
9	Feinentwurf der Komponenten (Klassendiagramm)	1 Tag	Mi 23.05.07	Mi 23.05.07	7	Dustin
10	*Nachbesprechung Architektur, neue Aufgabenstellung*	2 Std.	Fr 25.05.07	Fr 25.05.07	7;8;9	Projektteam, Betreuer
11	Desing und Implementierung GUI Prototyp	3 Tage	Fr 25.05.07	Mi 30.05.07	10	Sebastian, Frank, Boris
12	*Nachbesprechung GUI, Beginn Implementierungsphase*	2 Std.	Fr 01.06.07	Fr 01.06.07	11	Projektteam, Betreuer
13	Grafik	15 Tage	Mo 28.05.07	Fr 15.06.07	10	Boris
14	Handelssystem	6 Tage	Mo 04.06.07	Mo 11.06.07	12	Frank
15	Datenspeicherung	10 Tage	Mo 04.06.07	Fr 15.06.07	12	Jan
16	KI	10 Tage	Mo 04.06.07	Fr 15.06.07	12	Phillip
17	Kampfsystem	10 Tage	Mo 04.06.07	Fr 15.06.07	12	Fat, Dustin
18	Questsystem	4 Tage	Mo 18.06.07	Do 21.06.07	12	Frank, Sebastian
19	Netzcode	10 Tage	Mo 04.06.07	Fr 15.06.07	12	Frank, Sebastian, Bene, Dus
20	Erstellen von Testfällen	5 Tage	Mo 11.06.07	Fr 15.06.07	6;10	Projektteam
21	Zusammenführung der Module	4 Tage	Fr 22.06.07	Mi 27.06.07	13;14;15;16;17;20;18;19	Projektteam
22	Dokumentation	19 Tage	Mo 04.06.07	Do 28.06.07	12	Lena, Annette
23	**Alpha Version SEPVenture**	0 Tage	Mi 27.06.07	Mi 27.06.07	21	
24	Durchführung von Testfällen + Dokumentation	9 Tage	Mo 18.06.07	Do 28.06.07	20	Projektteam
25	**Beta Version SEPVenture**	0 Std.	Do 28.06.07	Do 28.06.07	24	
26	*Nachbesprechung Testfälle, Präsentationsbesprechung*	2 Std.	Fr 29.06.07	Fr 29.06.07	25	Projektteam, Betreuer
27	Gliederung der Präsentation	5 Tage	Fr 29.06.07	Fr 06.07.07	26	Projektteam
28	**Abnahme des SEPVenture**	2 Std.	Fr 06.07.07	Fr 06.07.07	27	Projektteam, Prüfer



5.0 Implementierung

5.1 Module (beispielhaft)

5.1.1 Graphics

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Rectangle;
import java.awt.Toolkit;
import java.awt.event.*;
import java.rmi.Naming;
import java.rmi.Remote;

import javax.swing.*;

interface Igraphic_client {
	abstract public void zeigeAusschnitt(Spieler spieler, Spielelement[][] feldFiguren,
Client client);
	abstract public void zeigeHandel(Spieler spieler,Haendler haendler, Client client);
	abstract public void zeigeInventar(Spieler spieler, Client client);
	abstract public void zeigeCharakter(Spieler spieler);
	abstract public void zeigeDialog(String titel, String anzeige);
	abstract public void zeigeQuestlog(Spieler spieler);
	abstract public void zeigeKarte(Spieler spieler);
	abstract public void zeigeMenue(Client client);
	abstract public void zeigeErstellen(Client client);
	abstract public void zeigeAufstieg(Client client, Spieler sp);
}
```

```
public class Graphics implements Igraphic_client{
	private FensterAusschnitt fensterAusschnitt;
	private FensterHandel fensterHandel;
	private FensterInventar fensterInventar;
	private FensterDialog fensterDialog;
	private FensterQuestlog fensterQuestlog;
	private FensterCharakter fensterCharakter;
	private FensterKarte fensterKarte;
	private FensterMenue fensterMenue;
	private FensterErstellen fensterErstellen;
	private FensterAufstieg fensterAufstieg;
	private int ausschnitt;

	public void zeigeAusschnitt(Spieler spieler, Spielelement[][] feldFiguren, Client
client) {
		if (spieler != null && feldFiguren != null && client != null) {
			if (fensterAusschnitt == null)
				fensterAusschnitt = new
FensterAusschnitt(spieler,feldFiguren,client);
			else
				fensterAusschnitt.zeichnen(spieler,feldFiguren);
		}
	}

	public void zeigeHandel(Spieler spieler,Haendler haendler, Client client) {
		if (spieler != null && haendler != null && client != null) {
			if (fensterHandel == null || fensterHandel.istSichtbar() == false)
				fensterHandel = new FensterHandel(spieler,haendler,client);
			else {
				fensterHandel.inventarAnzeigen();
				fensterHandel.spielerGoldAnzeigen();
				fensterHandel.ausgewaehltenGegenstandAnzeigen();
			}
		}
	}

	public void zeigeInventar(Spieler spieler, Client client) {
		if (spieler != null && client != null) {
			if (fensterInventar == null || fensterInventar.istSichtbar() == false)
				fensterInventar = new FensterInventar(spieler,client);
			else {
				fensterInventar.inventarAnzeigen();
				fensterInventar.spielerGoldAnzeigen();
				fensterInventar.ausgewaehltenGegenstandAnzeigen();
			}
		}
	}

	public void zeigeCharakter(Spieler spieler) {
		if (spieler != null)
```

```
                    if (fensterCharakter == null || fensterCharakter.istSichtbar() == false)
                        fensterCharakter = new FensterCharakter(spieler);
    }

    public void zeigeKarte(Spieler spieler) {
        if (spieler != null)
            if (fensterKarte == null || fensterKarte.istSichtbar() == false)
                fensterKarte = new FensterKarte(spieler);
    }

    public void zeigeDialog(String titel, String anzeige) {
        fensterDialog = new FensterDialog(titel,anzeige);
    }

    public void zeigeQuestlog(Spieler spieler) {
        fensterQuestlog = new FensterQuestlog(spieler);
    }

    public void zeigeMenue(Client client) {
        fensterMenue = new FensterMenue(client);
    }
    public void zeigeErstellen(Client client) {
        fensterErstellen = new FensterErstellen(client);
    }

    public void zeigeAufstieg(Client client, Spieler sp){
        fensterAufstieg = new FensterAufstieg(client, sp);
    }

    ////////////////////////////////
    // Klasse FensterAusschnitt   //
    ////////////////////////////////

    private class FensterAusschnitt extends JFrame implements KeyListener,
WindowListener{
        private Client client;
        private Spieler spieler;
        private JFrame fensterAusschnitt;
        private JLabel ausschnittHintergrund;
        private JLabel[][] bildFiguren;
        private JPanel layerFiguren;
        private JLayeredPane layerContainerAusschnitt;
        private Icon ausschnittHintergrundBild;
        private Spielelement[][] feldElemente;
        private int ausschnittID = -1;
        private JPanel layerGui;
        private JLabel anzeigeLebenspunkte;
        private JLabel anzeigeAktion;
        private Dimension d;
        private java.util.Timer time = new java.util.Timer();
```

```
                private FensterAusschnitt(Spieler spieler, Spielelement[][] feldElemente,
Client client) {
                        this.client = client;
                        fensterAusschnitt = new JFrame("SEPVenture");
                        fensterAusschnitt.setSize(806,632);

        fensterAusschnitt.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                    fensterAusschnitt.setVisible(true);
                    fensterAusschnitt.setResizable(false);
                    fensterAusschnitt.addKeyListener(this);
                    fensterAusschnitt.addWindowListener(this);
                    d = Toolkit.getDefaultToolkit().getScreenSize();
                    fensterAusschnitt.setLocation((d.width - fensterAusschnitt.getSize().width )
/ 2, (d.height - fensterAusschnitt.getSize().height) / 2 );
                    layerContainerAusschnitt = new JLayeredPane();
                    fensterAusschnitt.setContentPane(layerContainerAusschnitt);
                    ausschnittHintergrund = new JLabel(ausschnittHintergrundBild);
                    ausschnittHintergrund.setSize(800,600);
                    ausschnittHintergrund.setLocation(0,0);
                    layerContainerAusschnitt.add(ausschnittHintergrund,new Integer(0));

                    //Layer für Anzeigen der Lebenspunkte o.ä.
                layerGui = new JPanel();
                layerGui.setSize(800,600);
                layerGui.setLayout(null);
                layerGui.setOpaque(false);
                    anzeigeLebenspunkte = new JLabel();
                    anzeigeLebenspunkte.setFont(new Font("Courier New",Font.BOLD,16));
                    anzeigeLebenspunkte.setForeground(Color.LIGHT_GRAY);
                    anzeigeLebenspunkte.setLocation(10,575);
                    anzeigeLebenspunkte.setSize(200,20);
                    anzeigeAktion = new JLabel("",SwingConstants.TRAILING);
                    anzeigeAktion.setFont(new Font("Courier New",Font.BOLD,16));
                    anzeigeAktion.setForeground(Color.LIGHT_GRAY);
                    anzeigeAktion.setLocation(480,575);
                    anzeigeAktion.setSize(300,20);
                    layerGui.add(anzeigeLebenspunkte);
                    layerGui.add(anzeigeAktion);
                    layerContainerAusschnitt.add(layerGui,new Integer(300));
                    zeichnen(spieler,feldElemente);
                }

                private void zeichnen(Spieler spieler,Spielelement[][] feldElemente) {
                        this.spieler = spieler;
                        this.feldElemente = feldElemente;
                        //neuer Ausschnitt?
                        if (ausschnittID != spieler.getAusschnitt()) {
                                this.ausschnittID = spieler.getAusschnitt();
                           switch(spieler.getAusschnitt()) {
```

```
                        case 0: ausschnittHintergrundBild = new
ImageIcon("img/welt11.jpg"); break;
                        case 1: ausschnittHintergrundBild = new
ImageIcon("img/welt12.jpg"); break;
                        case 2: ausschnittHintergrundBild = new
ImageIcon("img/welt13.jpg"); break;
                        case 3: ausschnittHintergrundBild = new
ImageIcon("img/welt21.jpg"); break;
                        case 4: ausschnittHintergrundBild = new
ImageIcon("img/welt22.jpg"); break;
                        case 5: ausschnittHintergrundBild = new
ImageIcon("img/welt23.jpg"); break;
                        case 6: ausschnittHintergrundBild = new
ImageIcon("img/welt31.jpg"); break;
                        case 7: ausschnittHintergrundBild = new
ImageIcon("img/welt32.jpg"); break;
                        case 8: ausschnittHintergrundBild = new
ImageIcon("img/welt33.jpg"); break;
                        case 9: ausschnittHintergrundBild = new
ImageIcon("img/hoelle11.jpg"); break;
                        case 10: ausschnittHintergrundBild = new
ImageIcon("img/hoelle12.jpg"); break;
                        case 11: ausschnittHintergrundBild = new
ImageIcon("img/hoelle21.jpg"); break;
                        case 12: ausschnittHintergrundBild = new
ImageIcon("img/hoelle22.jpg"); break;
                        case 13: ausschnittHintergrundBild = new
ImageIcon("img/himmel11.jpg"); break;
                        case 14: ausschnittHintergrundBild = new
ImageIcon("img/himmel12.jpg"); break;
                        case 15: ausschnittHintergrundBild = new
ImageIcon("img/himmel21.jpg"); break;
                        case 16: ausschnittHintergrundBild = new
ImageIcon("img/himmel22.jpg"); break;
                        case 17: ausschnittHintergrundBild = new
ImageIcon("img/hoehle.jpg"); break;
                        }
                        ausschnittHintergrund.setIcon(ausschnittHintergrundBild);
                        if (layerFiguren == null) {
                                layerFiguren = new JPanel();
                                layerFiguren.setSize(800,600);
                                layerFiguren.setLayout(null);
                                layerFiguren.setLocation(0,0);
                                layerFiguren.setOpaque(false);
                                bildFiguren = new
JLabel[feldElemente.length][feldElemente[0].length];
                        }
                }
                //Spieler und andere Figuren neu zeichnen
                layerContainerAusschnitt.remove(layerFiguren);
                layerFiguren.removeAll();
```

```
                for (int x=0; x<feldElemente.length; x++)
                    for (int y=0; y<feldElemente[0].length; y++)
                        if (feldElemente[x][y] != null) {
                            bildFiguren[x][y] = new
JLabel(feldElemente[x][y].getBild());
                            bildFiguren[x][y].setSize(40,40);
                            bildFiguren[x][y].setLocation(x*40,y*40);
                            layerFiguren.add(bildFiguren[x][y]);
                        }
                layerContainerAusschnitt.add(layerFiguren,new Integer(100));
                anzeigeLebenspunkte.setText("Lebenspunkte:" +
spieler.getLebenspunkteAktuell() + "/" + spieler.getLebenspunkteMax());
                if (spieler.darfKaempfen())
                    anzeigeAktion.setText("Enter = Kampf starten");
                else if (spieler.darfHandeln())
                    anzeigeAktion.setText("Enter = Handel starten");
                else if (spieler.darfQuesten())
                    anzeigeAktion.setText("Enter = Questgeber ansprechen");
                else
                    anzeigeAktion.setText("");
            }


            public void keyPressed(KeyEvent ke){
                if(ke.getKeyCode() == KeyEvent.VK_LEFT)
                    try {
                        client.setSpieler(-1,0);
                        client.setLaufen(false);
                        time.schedule(new verzoegerung(),250);

                        }
                    catch (Exception e) {};
            if(ke.getKeyCode() == KeyEvent.VK_RIGHT)
                try {
                    client.setSpieler(1,0);
                    client.setLaufen(false);
                        time.schedule(new verzoegerung(),250);}
                    catch (Exception e) {};

            if(ke.getKeyCode() == KeyEvent.VK_UP)
                try {
                    client.setSpieler(0,-1);
                    client.setLaufen(false);
                        time.schedule(new verzoegerung(),250);}
                    catch (Exception e) {};
            if(ke.getKeyCode() == KeyEvent.VK_DOWN)
                try {
                    client.setSpieler(0,1);
                    client.setLaufen(false);
                        time.schedule(new verzoegerung(),250);}
                    catch (Exception e) {};
```

```
                        if(ke.getKeyCode() == KeyEvent.VK_N)
                try {
                        client.starteChat();}
                        catch (Exception e) {};
        if(ke.getKeyCode() == KeyEvent.VK_I)
                try {
                        client.zeigeInventar();
                        client.sound.playSound("inventar");
                }catch (Exception e) {};
                if(ke.getKeyCode() == KeyEvent.VK_S)
                {
                        if(client.sound.getSound())
                                                client.sound.setSound(false);

                                                else                    {

client.sound.setSound(true);

client.sound.playSoundloop("Hintergrundmusik");
                                                }
                }


                        if(ke.getKeyCode() == KeyEvent.VK_C)
                        try {
                                client.zeigeCharakter();}
                                catch (Exception e) {};
        if(ke.getKeyCode() == KeyEvent.VK_Q)
                try {
                        zeigeQuestlog(spieler); }
                        catch (Exception e) {};
                if(ke.getKeyCode() == KeyEvent.VK_M)
                try {
                        zeigeKarte(spieler); }
                        catch (Exception e) {};
                if(ke.getKeyCode() == KeyEvent.VK_ESCAPE)
                        try {
                                client.spielBeenden();
                        }
                        catch (Exception e) {};
        if(ke.getKeyCode() == KeyEvent.VK_ENTER) {
                try {
                        if (spieler.darfKaempfen())
                                client.spielerWillKaempfen();
                        else if (spieler.darfHandeln())
                                client.spielerWillHandeln();
                        else if (spieler.darfQuesten())
                                client.spielerWillQuesten();}
                        catch (Exception e) {};

        }
```

```
            }
            class verzoegerung extends java.util.TimerTask
            {
                    public void run()
                    {
                            client.setLaufen(true);
                    }
            }

            public void windowClosing(WindowEvent e) {
                    try {
                    client.spielBeenden(); }
                    catch (Exception ee) {};
            }

            public void keyReleased(KeyEvent ke)  {}
            public void keyTyped(KeyEvent ke) {}
        public void windowActivated (WindowEvent e) { }
        public void windowClosed (WindowEvent e) { }
        public void windowDeactivated (WindowEvent e) { }
        public void windowDeiconified (WindowEvent e) { }
        public void windowIconified (WindowEvent e) { }
        public void windowOpened (WindowEvent e) { }
    }

    ////////////////////////////////////
    // Klasse FensterHandel         //
    ////////////////////////////////////

    private class FensterHandel extends Frame implements KeyListener, WindowListener,
MouseListener {
            private JLabel bildMarkierung;
            private boolean kaufen;
            private Client client;
            private Spieler spieler;
            private Haendler haendler;
            private Gegenstand ausgewaehlterGegenstand;
            private JLabel labelHaendler;
            private JLabel labelSpieler;
            private JLabel gegenstandName;
            private JLabel gegenstandPreis;
            private JLabel spielerGold;
            private JLabel bildHintergrund;
            private JLabel[][] bildGegenstandSpieler;
            private JLabel[][] bildGegenstandHaendler;
            private JTextArea gegenstandBeschreibung;
            private JPanel layerAuswahl;
            private JPanel layerAnzeige;
            private JPanel layerGegenstaendeSpieler;
            private JButton buttonKaufen;
            private JButton buttonVerkaufen;
```

```
		private JButton buttonSwitch;
		private JButton buttonBeenden;
		private JLayeredPane layerContainerHandel;
		JFrame fensterHandel = new JFrame("Handel");
		private boolean sichtbar;
		private Dimension d;

		private FensterHandel(Spieler spieler, Haendler haendler, Client client) {
			this.client = client;
			this.haendler = haendler;
			this.spieler = spieler;
			sichtbar = true;
			kaufen = true;
			fensterHandel.setSize(707,383);
			fensterHandel.setResizable(false);
			fensterHandel.setVisible(true);
			fensterHandel.addKeyListener(this);
			fensterHandel.addWindowListener(this);
			fensterHandel.addMouseListener(this);
			d = Toolkit.getDefaultToolkit().getScreenSize();
			fensterHandel.setLocation((d.width - fensterHandel.getSize().width ) /
2, (d.height - fensterHandel.getSize().height) / 2 );

			layerContainerHandel = new JLayeredPane();
			fensterHandel.setContentPane(layerContainerHandel);

			//Hintergrundbild
		bildHintergrund = new JLabel(new ImageIcon("img/handel.jpg"));
		bildHintergrund.setSize(700,350);
		bildHintergrund.setLocation(0,0);
		layerContainerHandel.add(bildHintergrund,new Integer(0));

		//Anzeigen und Buttons
		layerAnzeige = new JPanel();
		layerAnzeige.setSize(700,350);
		layerAnzeige.setLocation(0,0);
		layerAnzeige.setLayout(null);
		layerAnzeige.setOpaque(false);
		buttonSwitch = new JButton("Wechseln");
		buttonSwitch.setLocation(300,50);
		buttonSwitch.setSize(100,30);
		buttonSwitch.setVisible(true);
		buttonSwitch.setFocusable(false);
		buttonSwitch.addActionListener(new ActionListener() {
		   public void actionPerformed(ActionEvent e) {
			kaufen = !kaufen;
				if (kaufen)
					layerAuswahl.setLocation(444,47);
				else
					layerAuswahl.setLocation(56,47);
				bildMarkierung.setLocation(0,0);
```

```
                    ausgewaehltenGegenstandAnzeigen();
    }
});
buttonBeenden = new JButton("Beenden");
buttonBeenden.setLocation(554,310);
buttonBeenden.setSize(100,30);
buttonBeenden.setVisible(true);
buttonBeenden.setFocusable(false);
buttonBeenden.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        fensterHandel.dispose();
        windowClosing(null);
    }
});
buttonKaufen = new JButton("Kaufen");
buttonKaufen.setLocation(300,220);
buttonKaufen.setSize(100,30);
buttonKaufen.setFocusable(false);
buttonKaufen.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        kaufen();
    }
});
buttonVerkaufen = new JButton("Verkaufen");
buttonVerkaufen.setLocation(300,220);
buttonVerkaufen.setSize(100,30);
buttonVerkaufen.setFocusable(false);
buttonVerkaufen.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        verkaufen();
    }
});
spielerGold = new JLabel();
spielerGold.setSize(140,20);
spielerGold.setLocation(46,300);
gegenstandName = new JLabel();
gegenstandName.setSize(140,20);
gegenstandName.setLocation(280,100);
gegenstandBeschreibung = new JTextArea();
gegenstandBeschreibung.setSize(140,60);
gegenstandBeschreibung.setLocation(280,120);
gegenstandBeschreibung.setLineWrap(true);
gegenstandBeschreibung.setWrapStyleWord(true);
gegenstandBeschreibung.setFocusable(false);
gegenstandBeschreibung.setOpaque(false);
gegenstandPreis = new JLabel();
gegenstandPreis.setSize(140,20);
gegenstandPreis.setLocation(280,180);
labelSpieler = new JLabel("Spieler");
labelSpieler.setSize(100,20);
labelSpieler.setLocation(46,12);
```

```
    labelHaendler = new JLabel("Händler");
    labelHaendler.setSize(100,20);
    labelHaendler.setLocation(442,12);
    layerAnzeige.add(buttonSwitch);
    layerAnzeige.add(buttonBeenden);
    layerAnzeige.add(buttonKaufen);
    layerAnzeige.add(buttonVerkaufen);
    layerAnzeige.add(gegenstandName);
    layerAnzeige.add(gegenstandPreis);
    layerAnzeige.add(gegenstandBeschreibung);
    layerAnzeige.add(labelSpieler);
    layerAnzeige.add(labelHaendler);
    layerAnzeige.add(spielerGold);
    layerContainerHandel.add(layerAnzeige,new Integer(100));

    //Gegenstände
    layerGegenstaendeSpieler = new JPanel();
    layerGegenstaendeSpieler.setLocation(55,47);
    layerGegenstaendeSpieler.setSize(590,240);
    layerGegenstaendeSpieler.setLayout(null);
    layerGegenstaendeSpieler.setOpaque(false);
    bildGegenstandSpieler = new JLabel[5][6];
    bildGegenstandHaendler = new JLabel[5][6];
    layerContainerHandel.add(layerGegenstaendeSpieler,new Integer(200));

    //Auswahlmarkierung
    layerAuswahl = new JPanel();
    layerAuswahl.setSize(200,240);
    layerAuswahl.setLocation(444,47);
    layerAuswahl.setLayout(null);
    layerAuswahl.setOpaque(false);
    bildMarkierung = new JLabel(new ImageIcon("img/auswahl.gif"));
    bildMarkierung.setSize(40,40);
    bildMarkierung.setLocation(0,0);
    layerAuswahl.add(bildMarkierung);
    layerContainerHandel.add(layerAuswahl,new Integer(300));

    ausgewaehltenGegenstandAnzeigen();
    inventarAnzeigen();
    spielerGoldAnzeigen();
}

private void kaufen() {
        int x = bildMarkierung.getX() / 40;
        int y = bildMarkierung.getY() / 40;
        client.spielerWillKaufen(haendler,x,y);
}

private void verkaufen() {
        int x = bildMarkierung.getX() / 40;
        int y = bildMarkierung.getY() / 40;
```

```
                client.spielerWillVerkaufen(haendler,x,y);
        }

        private void auswahlRechts() {
                if (bildMarkierung.getX() < 160) {
                        bildMarkierung.setLocation(bildMarkierung.getX() + 40,
bildMarkierung.getY());
                        ausgewaehltenGegenstandAnzeigen();
                }
                else if (bildMarkierung.getX() == 160 && bildMarkierung.getY() <
200) {
                        bildMarkierung.setLocation(0, bildMarkierung.getY() + 40);
                        ausgewaehltenGegenstandAnzeigen();
                }
        }

        private void auswahlLinks() {
                if (bildMarkierung.getX() > 0) {
                        bildMarkierung.setLocation(bildMarkierung.getX() - 40,
bildMarkierung.getY());
                        ausgewaehltenGegenstandAnzeigen();
                }
                else if (bildMarkierung.getX() == 0 && bildMarkierung.getY() > 0) {
                        bildMarkierung.setLocation(160, bildMarkierung.getY() - 40);
                        ausgewaehltenGegenstandAnzeigen();
                }
        }

        private void auswahlHoch() {
                if (bildMarkierung.getY() > 0) {
                        bildMarkierung.setLocation(bildMarkierung.getX(),
bildMarkierung.getY() - 40);
                        ausgewaehltenGegenstandAnzeigen();
                }
                else if (bildMarkierung.getX() > 0 && bildMarkierung.getY() == 0) {
                        bildMarkierung.setLocation(bildMarkierung.getX() - 40,200);
                        ausgewaehltenGegenstandAnzeigen();
                }
        }

        private void auswahlRunter() {
                if (bildMarkierung.getY() < 200) {
                        bildMarkierung.setLocation(bildMarkierung.getX(),
bildMarkierung.getY() + 40);
                        ausgewaehltenGegenstandAnzeigen();
                }
                else if (bildMarkierung.getX() < 160 && bildMarkierung.getY() ==
200) {
                        bildMarkierung.setLocation(bildMarkierung.getX() + 40,0);
                        ausgewaehltenGegenstandAnzeigen();
                }
```

```
	}

	private void ausgewaehltenGegenstandAnzeigen() {
		Inventar inventar;
		if (kaufen)
			inventar = haendler.getInventar();
		else
			inventar = spieler.getInventar();
		int x = bildMarkierung.getX() / 40;
		int y = bildMarkierung.getY() / 40;
		ausgewaehlterGegenstand = inventar.getGegenstand(x,y);
		if (ausgewaehlterGegenstand != null) {
		gegenstandName.setText(ausgewaehlterGegenstand.getTitel());
		if (kaufen) {
			buttonKaufen.setVisible(true);
			buttonVerkaufen.setVisible(false);
		}
		else {
			buttonKaufen.setVisible(false);
			buttonVerkaufen.setVisible(true);
		}

	gegenstandBeschreibung.setText(ausgewaehlterGegenstand.getBezeichnung());
		if (ausgewaehlterGegenstand.isVerkaeuflich())
			gegenstandPreis.setText("Preis: " +
haendler.getPreis(ausgewaehlterGegenstand.getName()) + " Gold");
		else {
			gegenstandPreis.setText("unverkäuflich");
			buttonVerkaufen.setVisible(false);
		}
		}
		else {
			gegenstandName.setText("");
			gegenstandBeschreibung.setText("");
			gegenstandPreis.setText("");
			buttonKaufen.setVisible(false);
			buttonVerkaufen.setVisible(false);
		}
	}

	private void spielerGoldAnzeigen() {
		spielerGold.setText("Du hast " + spieler.getGold() + " Goldstücke.");
	}

	private boolean istSichtbar() {
		return sichtbar;
	}

	private void inventarAnzeigen() {
		layerGegenstaendeSpieler.removeAll();
		for (int y=0; y<spieler.getInventar().getHoehe(); y++) {
```

```
                    for (int x=0; x<spieler.getInventar().getBreite(); x++) {
                        if (spieler.getInventar().getGegenstand(x,y) != null)
                            bildGegenstandSpieler[x][y] = new
JLabel(spieler.getInventar().getGegenstand(x,y).getBild());
                        else
                            bildGegenstandSpieler[x][y] = new JLabel();
                        bildGegenstandSpieler[x][y].setSize(40, 40);
                        bildGegenstandSpieler[x][y].setLocation(x*40, y*40);
                        if (haendler.getInventar().getGegenstand(x,y) != null)
                            bildGegenstandHaendler[x][y] = new
JLabel(haendler.getInventar().getGegenstand(x,y).getBild());
                        else
                            bildGegenstandHaendler[x][y] = new JLabel();
                        bildGegenstandHaendler[x][y].setSize(40, 40);
                        bildGegenstandHaendler[x][y].setLocation(x*40+389, y*40);
                        layerGegenstaendeSpieler.add(bildGegenstandHaendler[x][y]);
                        layerGegenstaendeSpieler.add(bildGegenstandSpieler[x][y]);
                    }
                }
                    layerContainerHandel.add(layerGegenstaendeSpieler,new
Integer(200));
            }

            public void keyPressed(KeyEvent ke) {
                if(ke.getKeyCode() == KeyEvent.VK_LEFT)
                auswahlLinks();
            if(ke.getKeyCode() == KeyEvent.VK_RIGHT)
                auswahlRechts();
            if(ke.getKeyCode() == KeyEvent.VK_UP)
                auswahlHoch();
            if(ke.getKeyCode() == KeyEvent.VK_DOWN)
                auswahlRunter();
            if(ke.getKeyCode() == KeyEvent.VK_H) {
                fensterHandel.dispose();
            windowClosing(null);
            }
            if(ke.getKeyCode() == KeyEvent.VK_ENTER)
                if (buttonKaufen.isVisible())
                    kaufen();
                else if (buttonVerkaufen.isVisible())
                    verkaufen();
            }

            public void windowClosing (WindowEvent e) {
                client.setLaufen(true);
                sichtbar = false;
            }

            public void mousePressed (MouseEvent e) {
            if (e.getX()>447 && e.getX()<647 && e.getY()>73 && e.getY()<313) {
                kaufen = true;
```

```
                        layerAuswahl.setLocation(444,47);
                        bildMarkierung.setLocation(((e.getX()-447)/40)*40, ((e.getY()-
73)/40)*40);
                                ausgewaehltenGegenstandAnzeigen();
                    }
                else if (e.getX()>56 && e.getX()<256 && e.getY()>73 && e.getY()<313) {
                        kaufen = false;
                        layerAuswahl.setLocation(56,47);
                        bildMarkierung.setLocation(((e.getX()-56)/40)*40, ((e.getY()-
73)/40)*40);
                                ausgewaehltenGegenstandAnzeigen();
                    }
            }

            public void keyReleased(KeyEvent ke) { }
            public void keyTyped(KeyEvent ke) { }
            public void windowActivated (WindowEvent e) { }
            public void windowClosed (WindowEvent e) { }
            public void windowDeactivated (WindowEvent e) { }
            public void windowDeiconified (WindowEvent e) { }
            public void windowIconified (WindowEvent e) { }
            public void windowOpened (WindowEvent e) { }
            public void mouseReleased (MouseEvent e) { }
            public void mouseClicked(MouseEvent e) { }
            public void mouseEntered (MouseEvent e) { }
            public void mouseExited (MouseEvent e) { }
        }

                ////////////////////////////////////
        // Klasse FensterInventar       //
        ////////////////////////////////////

        private class FensterInventar extends Frame implements KeyListener,
WindowListener, MouseListener {
                private JLabel bildMarkierung;
                private Spieler spieler;
                private JLabel gegenstandName;
                private JLabel spielerGold;
                private JLabel bildHintergrund;
                private JLabel[][] bildGegenstandSpieler;
                private JTextArea gegenstandBeschreibung;
                private JPanel layerAuswahl;
                private JPanel layerAnzeige;
                private JPanel layerGegenstaendeSpieler;
                private JButton buttonBenutzen;
                private JButton buttonWaffe;
                private JButton buttonBeenden;
                private JLayeredPane layerContainerInventar;
                JFrame fensterInventar = new JFrame("Inventar");
                private Client client;
                private boolean sichtbar;
```

```
		private Gegenstand ausgewaehlterGegenstand;
		private Dimension d;

		private FensterInventar(Spieler spieler, Client client) {
			this.spieler = spieler;
			this.client = client;
			sichtbar = true;
			fensterInventar.setSize(446,383);
			fensterInventar.setResizable(false);
			fensterInventar.setVisible(true);
			fensterInventar.addKeyListener(this);
			fensterInventar.addMouseListener(this);
			fensterInventar.addWindowListener(this);
			d = Toolkit.getDefaultToolkit().getScreenSize();
			fensterInventar.setLocation((d.width - fensterInventar.getSize().width )
/ 2, (d.height - fensterInventar.getSize().height) / 2 );

			layerContainerInventar = new JLayeredPane();
			fensterInventar.setContentPane(layerContainerInventar);

			//Hintergrundbild
		bildHintergrund = new JLabel(new ImageIcon("img/inventar.jpg"));
		bildHintergrund.setSize(440,350);
		bildHintergrund.setLocation(0,0);
		layerContainerInventar.add(bildHintergrund,new Integer(0));

		//Anzeigen und Buttons
		layerAnzeige = new JPanel();
		layerAnzeige.setSize(440,350);
		layerAnzeige.setLocation(0,0);
		layerAnzeige.setLayout(null);
		layerAnzeige.setOpaque(false);
		buttonBeenden = new JButton("Beenden");
		buttonBeenden.setLocation(300,280);
		buttonBeenden.setSize(100,30);
		buttonBeenden.setVisible(true);
		buttonBeenden.setFocusable(false);
		buttonBeenden.addActionListener(new ActionListener() {
		   public void actionPerformed(ActionEvent e) {
		      fensterInventar.dispose();
		      windowClosing(null);
		   }
		});
		buttonBenutzen = new JButton("Benutzen");
		buttonBenutzen.setLocation(300,50);
		buttonBenutzen.setSize(100,30);
		buttonBenutzen.setVisible(true);
		buttonBenutzen.setFocusable(false);
		buttonBenutzen.addActionListener(new ActionListener() {
		   public void actionPerformed(ActionEvent e) {
		      benutzeGegenstand();
```

```
    }
});
buttonWaffe = new JButton("Waehlen");
buttonWaffe.setLocation(300,50);
buttonWaffe.setSize(100,30);
buttonWaffe.setVisible(false);
buttonWaffe.setFocusable(false);
buttonWaffe.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        gegenstandWaehlen();
    }
});
spielerGold = new JLabel();
spielerGold.setSize(140,20);
spielerGold.setLocation(46,300);
gegenstandName = new JLabel();
gegenstandName.setSize(140,20);
gegenstandName.setLocation(280,100);
gegenstandBeschreibung = new JTextArea();
gegenstandBeschreibung.setSize(140,60);
gegenstandBeschreibung.setLocation(280,120);
gegenstandBeschreibung.setLineWrap(true);
gegenstandBeschreibung.setWrapStyleWord(true);
gegenstandBeschreibung.setFocusable(false);
gegenstandBeschreibung.setOpaque(false);
layerAnzeige.add(buttonBeenden);
layerAnzeige.add(buttonBenutzen);
layerAnzeige.add(buttonWaffe);
layerAnzeige.add(gegenstandName);
layerAnzeige.add(gegenstandBeschreibung);
layerAnzeige.add(spielerGold);
layerContainerInventar.add(layerAnzeige,new Integer(100));

//Gegenstände
layerGegenstaendeSpieler = new JPanel();
layerGegenstaendeSpieler.setLocation(55,47);
layerGegenstaendeSpieler.setSize(590,240);
layerGegenstaendeSpieler.setLayout(null);
layerGegenstaendeSpieler.setOpaque(false);
bildGegenstandSpieler = new JLabel[5][6];
layerContainerInventar.add(layerGegenstaendeSpieler,new Integer(200));

//Auswahlmarkierung
layerAuswahl = new JPanel();
layerAuswahl.setSize(200,240);
layerAuswahl.setLocation(56,47);
layerAuswahl.setLayout(null);
layerAuswahl.setOpaque(false);
bildMarkierung = new JLabel(new ImageIcon("img/auswahl.gif"));
bildMarkierung.setSize(40,40);
bildMarkierung.setLocation(0,0);
```

```
            layerAuswahl.add(bildMarkierung);
            layerContainerInventar.add(layerAuswahl,new Integer(300));

            ausgewaehltenGegenstandAnzeigen();
            inventarAnzeigen();
            spielerGoldAnzeigen();
        }

        private void benutzeGegenstand() {
            int x = bildMarkierung.getX() / 40;
            int y = bildMarkierung.getY() / 40;
            client.benutzeGegenstand(x,y);
        }

        private void gegenstandWaehlen() {
            client.setWaffeRuestung(bildMarkierung.getX() / 40,
bildMarkierung.getY() / 40);
            ausgewaehltenGegenstandAnzeigen();
        }

        private void auswahlRechts() {
            if (bildMarkierung.getX() < 160) {
                bildMarkierung.setLocation(bildMarkierung.getX() + 40,
bildMarkierung.getY());
                ausgewaehltenGegenstandAnzeigen();
            }
            else if (bildMarkierung.getX() == 160 && bildMarkierung.getY() <
200) {
                bildMarkierung.setLocation(0, bildMarkierung.getY() + 40);
                ausgewaehltenGegenstandAnzeigen();
            }
        }

        private void auswahlLinks() {
            if (bildMarkierung.getX() > 0) {
                bildMarkierung.setLocation(bildMarkierung.getX() - 40,
bildMarkierung.getY());
                ausgewaehltenGegenstandAnzeigen();
            }
            else if (bildMarkierung.getX() == 0 && bildMarkierung.getY() > 0) {
                bildMarkierung.setLocation(160, bildMarkierung.getY() - 40);
                ausgewaehltenGegenstandAnzeigen();
            }
        }

        private void auswahlHoch() {
            if (bildMarkierung.getY() > 0) {
                bildMarkierung.setLocation(bildMarkierung.getX(),
bildMarkierung.getY() - 40);
                ausgewaehltenGegenstandAnzeigen();
            }
```

```
                else if (bildMarkierung.getX() > 0 && bildMarkierung.getY() == 0) {
                        bildMarkierung.setLocation(bildMarkierung.getX() - 40,200);
                        ausgewaehltenGegenstandAnzeigen();
                }
        }

        private void auswahlRunter() {
                if (bildMarkierung.getY() < 200) {
                        bildMarkierung.setLocation(bildMarkierung.getX(),
bildMarkierung.getY() + 40);
                        ausgewaehltenGegenstandAnzeigen();
                }
                else if (bildMarkierung.getX() < 160 && bildMarkierung.getY() ==
200) {
                        bildMarkierung.setLocation(bildMarkierung.getX() + 40,0);
                        ausgewaehltenGegenstandAnzeigen();
                }
        }

        private void ausgewaehltenGegenstandAnzeigen() {
                Inventar inventar = spieler.getInventar();
                int x = bildMarkierung.getX() / 40;
                int y = bildMarkierung.getY() / 40;
                ausgewaehlterGegenstand = inventar.getGegenstand(x,y);
                if (ausgewaehlterGegenstand != null) {
                gegenstandName.setText(ausgewaehlterGegenstand.getTitel());

        gegenstandBeschreibung.setText(ausgewaehlterGegenstand.getBezeichnung());
                if (ausgewaehlterGegenstand.isBenutzbar()) {
                        buttonBenutzen.setVisible(true);
                        buttonWaffe.setVisible(true);
                }
                else {
                        buttonBenutzen.setVisible(false);
                        if (!((x == 0 && y ==0) || (x == 0 && y==1)) &&
(ausgewaehlterGegenstand.getName().contains("waffe") ||
ausgewaehlterGegenstand.getName().contains("ruestung")))
                                buttonWaffe.setVisible(true);
                        else
                                buttonWaffe.setVisible(false);
                }
                }
                else {
                        gegenstandName.setText("");
                        gegenstandBeschreibung.setText("");
                        buttonBenutzen.setVisible(false);
                        buttonWaffe.setVisible(false);
                }
        }

        private void spielerGoldAnzeigen() {
```

```
                spielerGold.setText("Du hast " + spieler.getGold() + " Goldstücke.");
        }

        private boolean istSichtbar() {
                return sichtbar;
        }

        private void inventarAnzeigen() {
                layerGegenstaendeSpieler.removeAll();
                for (int y=0; y<spieler.getInventar().getHoehe(); y++) {
                for (int x=0; x<spieler.getInventar().getBreite(); x++) {
                        if (spieler.getInventar().getGegenstand(x,y) != null)
                                bildGegenstandSpieler[x][y] = new
JLabel(spieler.getInventar().getGegenstand(x,y).getBild());
                        else
                                bildGegenstandSpieler[x][y] = new JLabel();
                        bildGegenstandSpieler[x][y].setSize(40, 40);
                        bildGegenstandSpieler[x][y].setLocation(x*40, y*40);
                        layerGegenstaendeSpieler.add(bildGegenstandSpieler[x][y]);
                }
              }
                layerContainerInventar.add(layerGegenstaendeSpieler,new
Integer(200));
        }

        public void keyPressed(KeyEvent ke) {
                if(ke.getKeyCode() == KeyEvent.VK_LEFT)
                auswahlLinks();
        if(ke.getKeyCode() == KeyEvent.VK_RIGHT)
                auswahlRechts();
        if(ke.getKeyCode() == KeyEvent.VK_UP)
                auswahlHoch();
        if(ke.getKeyCode() == KeyEvent.VK_DOWN)
                auswahlRunter();
        if(ke.getKeyCode() == KeyEvent.VK_I) {
                fensterInventar.dispose();
        windowClosing(null);
        }
        if(ke.getKeyCode() == KeyEvent.VK_ENTER)
                if (buttonBenutzen.isVisible())
                        benutzeGegenstand();
                else if (buttonWaffe.isVisible())
                        gegenstandWaehlen();
        }

        public void mousePressed (MouseEvent e) {
        if (e.getX()>56 && e.getX()<256 && e.getY()>73 && e.getY()<313) {
                layerAuswahl.setLocation(56,47);
                bildMarkierung.setLocation(((e.getX()-56)/40)*40, ((e.getY()-
73)/40)*40);
                        ausgewaehltenGegenstandAnzeigen();
```

```
            }
        }

            public void windowClosing (WindowEvent e) {
                sichtbar = false;
            }

        public void keyReleased(KeyEvent ke) { }
        public void keyTyped(KeyEvent ke) { }
        public void mouseReleased (MouseEvent e) { }
        public void mouseClicked(MouseEvent e) { }
        public void mouseEntered (MouseEvent e) { }
        public void mouseExited (MouseEvent e) { }
        public void windowActivated (WindowEvent e) { }
        public void windowClosed (WindowEvent e) { }
        public void windowDeactivated (WindowEvent e) { }
        public void windowDeiconified (WindowEvent e) { }
        public void windowIconified (WindowEvent e) { }
        public void windowOpened (WindowEvent e) { }
    }

    ////////////////////////////////////
    // Klasse FensterCharakter       //
    ////////////////////////////////////

    private class FensterCharakter extends Frame implements KeyListener,
WindowListener{
            private Spieler spieler;
            private JLabel spielerBild;
            private JLabel bildHintergrund;
            private JPanel layerAnzeige;
            private JLabel spielerStaerke;
            private JLabel spielerGewandtheit;
            private JLabel spielerIntelligenz;
            private JLabel spielerAusdauer;
            private JLabel spielerRasse;
            private JLabel spielerKlasse;
            private JLabel spielerName;
            private JLabel spielerErfahrung;
            private JLabel spielerLevel;
            private JLabel spielerGold;
            private JButton buttonBeenden;
            private JLayeredPane layerContainerCharakter;
            JFrame fensterCharakter = new JFrame("Charakteruebersicht");
            private boolean sichtbar;
            private Dimension d;

            private FensterCharakter(Spieler spieler) {
                this.spieler = spieler;
                sichtbar = true;
                fensterCharakter.addWindowListener(this);
```

```
                        fensterCharakter.addKeyListener(this);
                        fensterCharakter.setSize(446,283);
                        fensterCharakter.setResizable(false);
                        fensterCharakter.setVisible(true);
                        d = Toolkit.getDefaultToolkit().getScreenSize();
                        fensterCharakter.setLocation((d.width -
fensterCharakter.getSize().width ) / 2, (d.height - fensterCharakter.getSize().height) / 2 );

                        layerContainerCharakter = new JLayeredPane();
                        layerContainerCharakter.setLayout(null);
                        fensterCharakter.setContentPane(layerContainerCharakter);

                        //Hintergrundbild
                        bildHintergrund = new JLabel(new
ImageIcon("img/hintergrundstart.jpg"));
        bildHintergrund.setBounds(0,0,550,450);
                    /*  bildHintergrund = new JLabel(new ImageIcon(""));
                      bildHintergrund.setSize(440,350);
                      bildHintergrund.setLocation(0,0);
                      */
                      //Anzeige
                      layerAnzeige = new JPanel();
                      layerAnzeige.setSize(440,350);
                      layerAnzeige.setLocation(0,0);
                      layerAnzeige.setLayout(null);
                      layerAnzeige.setOpaque(false);
                      spielerBild = new JLabel(new ImageIcon("img/" +
spieler.getRasse().toLowerCase() + spieler.getKlasse().toLowerCase() + ".png"));
                      spielerBild.setSize(100,200);
                      spielerBild.setLocation(20,20);
                      spielerName = new JLabel(spieler.getName());
                      spielerName.setBounds(150,30,200,30);
                      spielerName.setFont(new Font("Courier New",Font.BOLD,16));
                      spielerName.setForeground(Color.DARK_GRAY);
                      spielerRasse = new JLabel("Rasse: " + spieler.getRasse());
                      spielerRasse.setBounds(150,60,150,30);
                      spielerKlasse = new JLabel("Klasse: " + spieler.getKlasse());
                      spielerKlasse.setBounds(300,60,150,30);
                      spielerErfahrung = new JLabel("Erfahrungspunkte: " +
spieler.geterfahrungspunkte());
                      spielerErfahrung.setBounds(150,90,150,30);
                      spielerLevel = new JLabel("Level: " + spieler.getLevel());
                      spielerLevel.setBounds(300,90,150,30);
                      spielerGold = new JLabel("Gold: " + spieler.getGold());
                      spielerGold.setBounds(150,180,100,30);
                      spielerStaerke = new JLabel("Staerke: " + spieler.getstaerke());
                      spielerStaerke.setBounds(150,120,150,30);
                      spielerGewandtheit = new JLabel("Gewandtheit: " +
spieler.getgewandtheit());
                      spielerGewandtheit.setBounds(150,150,150,30);
                      spielerIntelligenz = new JLabel("Intelligenz: " + spieler.getintelligenz());
```

```
            spielerIntelligenz.setBounds(300,120,150,30);
            spielerAusdauer = new JLabel("Ausdauer: " + spieler.getausdauer());
            spielerAusdauer.setBounds(300,150,150,30);
            buttonBeenden = new JButton("Beenden");
            buttonBeenden.setBounds(300,200,100,30);
            buttonBeenden.setVisible(true);
            buttonBeenden.setFocusable(false);
            buttonBeenden.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                  fensterCharakter.dispose();
                  windowClosing(null);
               }
            });

            layerAnzeige.add(buttonBeenden);
            layerAnzeige.add(spielerName);
            layerAnzeige.add(spielerBild);
            layerAnzeige.add(spielerStaerke);
            layerAnzeige.add(spielerGewandtheit);
            layerAnzeige.add(spielerIntelligenz);
            layerAnzeige.add(spielerAusdauer);
            layerAnzeige.add(spielerErfahrung);
            layerAnzeige.add(spielerRasse);
            layerAnzeige.add(spielerKlasse);
            layerAnzeige.add(spielerLevel);
            layerAnzeige.add(spielerGold);
            layerContainerCharakter.add(bildHintergrund,new Integer(0));
            layerContainerCharakter.add(layerAnzeige,new Integer(100));
        }

        public void keyPressed(KeyEvent ke) {
                if(ke.getKeyCode() == KeyEvent.VK_C) {
                fensterCharakter.dispose();
                sichtbar = false;
                }
        }

    public void keyReleased(KeyEvent ke) { }
    public void keyTyped(KeyEvent ke) { }

        private boolean istSichtbar() {
                return sichtbar;
        }

        public void windowClosing (WindowEvent e) {
                sichtbar = false;
        }

    public void windowActivated (WindowEvent e) { }
    public void windowClosed (WindowEvent e) { }
    public void windowDeactivated (WindowEvent e) { }
```

```
    public void windowDeiconified (WindowEvent e) { }
    public void windowIconified (WindowEvent e) { }
    public void windowOpened (WindowEvent e) { }
}

////////////////////////////////////
// Klasse FensterKarte       //
////////////////////////////////////

private class FensterKarte extends Frame implements KeyListener, WindowListener{

        private Spieler spieler;
        //private JPanel layerGegenstaendeSpieler;
        private JButton buttonBeenden;
        private JLayeredPane layerContainerKarte;
        private JLabel bildHintergrund;
        private JLabel bildMarkierung;
        private JPanel layerAnzeige;
        JFrame fensterKarte = new JFrame("Karte");
        private boolean sichtbar;
        private Dimension d;

        private FensterKarte(Spieler spieler) {
    this.spieler = spieler;
    ausschnitt = spieler.getAusschnitt();
    //Keine Karte für Bonuslevel
    if (ausschnitt == 17)
        return;
    sichtbar = true;
    fensterKarte.addWindowListener(this);
    fensterKarte.addKeyListener(this);
    fensterKarte.setResizable(false);
    fensterKarte.setVisible(true);
    layerContainerKarte = new JLayeredPane();
    layerContainerKarte.setLayout(null);
    fensterKarte.setContentPane(layerContainerKarte);
    bildHintergrund = new JLabel();

   //Anzeige
   layerAnzeige = new JPanel();
   layerAnzeige.setSize(600,450);
   layerAnzeige.setLocation(0,0);
   layerAnzeige.setLayout(null);
   layerAnzeige.setOpaque(false);
   bildMarkierung = new JLabel(new ImageIcon("img/auswahlmap.gif"));
   bildMarkierung.setSize(200,150);
   layerAnzeige.add(bildMarkierung);
   layerContainerKarte.add(bildHintergrund,new Integer(0));
   layerContainerKarte.add(layerAnzeige,new Integer(100));
   if (ausschnitt >= 0 && ausschnitt <= 8) {
       bildHintergrund.setIcon(new ImageIcon("img/maperde.jpg"));
```

```
            bildHintergrund.setBounds(0,0,600,450);
            fensterKarte.setSize(606,482);
        }
        else if (ausschnitt >= 9 && ausschnitt <= 12) {
            bildHintergrund.setIcon(new ImageIcon("img/maphoelle.jpg"));
            bildHintergrund.setBounds(0,0,400,300);
            fensterKarte.setSize(406,332);
        }
        else if (ausschnitt >= 13 && ausschnitt <= 16) {
            bildHintergrund.setIcon(new ImageIcon("img/maphimmel.jpg"));
            bildHintergrund.setBounds(0,0,400,300);
            fensterKarte.setSize(406,332);
        }
        d = Toolkit.getDefaultToolkit().getScreenSize();
         fensterKarte.setLocation((d.width - fensterKarte.getSize().width ) / 2, (d.height -
fensterKarte.getSize().height) / 2 );
        if (ausschnitt == 0 || ausschnitt == 9 || ausschnitt == 13)
            bildMarkierung.setLocation(0,0);
        else if (ausschnitt == 1 || ausschnitt == 10 || ausschnitt == 14)
            bildMarkierung.setLocation(200,0);
        else if (ausschnitt == 2)
            bildMarkierung.setLocation(400,0);
        else if (ausschnitt == 3  || ausschnitt == 11 || ausschnitt == 15)
            bildMarkierung.setLocation(0,150);
        else if (ausschnitt == 4  || ausschnitt == 12 || ausschnitt == 16)
            bildMarkierung.setLocation(200,150);
        else if (ausschnitt == 5)
            bildMarkierung.setLocation(400,150);
        else if (ausschnitt == 6)
            bildMarkierung.setLocation(0,300);
        else if (ausschnitt == 7)
            bildMarkierung.setLocation(200,300);
        else if (ausschnitt == 8)
            bildMarkierung.setLocation(400,300);
    }

            private boolean istSichtbar() {
                    return sichtbar;
            }

            public void keyPressed(KeyEvent ke) {
                    if(ke.getKeyCode() == KeyEvent.VK_M) {
                    fensterKarte.dispose();
                    sichtbar = false;
                    }
            }

        public void keyReleased(KeyEvent ke) { }
        public void keyTyped(KeyEvent ke) { }

            public void windowClosing (WindowEvent e) {
```

```
                sichtbar = false;
            }

        public void windowActivated (WindowEvent e) { }
        public void windowClosed (WindowEvent e) { }
        public void windowDeactivated (WindowEvent e) { }
        public void windowDeiconified (WindowEvent e) { }
        public void windowIconified (WindowEvent e) { }
        public void windowOpened (WindowEvent e) { }
    }


    ////////////////////////////////
    // Klasse FensterDialog       //
    ////////////////////////////////

    private class FensterDialog implements KeyListener{
            private JDialog dialog;
            private Dimension d;
            private JTextArea angezeigeText;
            private JButton buttonOK;

            private FensterDialog(String titel, String anzeige) {
                dialog = new JDialog((JFrame)fensterAusschnitt, titel, true);
                dialog.setSize(200,150);
                dialog.setResizable(false);
                dialog.setLayout(null);

    dialog.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                dialog.addKeyListener(this);
                d = Toolkit.getDefaultToolkit().getScreenSize();
                dialog.setLocation((d.width - dialog.getSize().width ) / 2, (d.height -
dialog.getSize().height) / 2 );
                angezeigeText = new JTextArea("\"" + anzeige + "\"");
                angezeigeText.setSize(180,60);
                angezeigeText.setLocation(10,10);
              angezeigeText.setLineWrap(true);
              angezeigeText.setWrapStyleWord(true);
              angezeigeText.setFocusable(false);
              angezeigeText.setOpaque(false);
                buttonOK = new JButton("OK");
                buttonOK.setLocation(50,80);
                buttonOK.setSize(100,30);
                buttonOK.setFocusable(false);
                buttonOK.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            dialog.dispose();
                    }
                });
                dialog.add(buttonOK);
                dialog.add(angezeigeText);
```

```
                dialog.setVisible(true);
            }

            public void keyPressed(KeyEvent ke) {
                if(ke.getKeyCode() == KeyEvent.VK_ENTER)
                    dialog.dispose();
            }
            public void keyReleased(KeyEvent ke) { }
          public void keyTyped(KeyEvent ke) { }
        }

        ////////////////////////////////
        // Klasse FensterQuestlog     //
        ////////////////////////////////

        private class FensterQuestlog implements KeyListener{
            private JFrame fensterQuestlog = new JFrame("Questlog");
            private Spieler spieler;
            private JTextArea[] questBeschreibung;
            private JLabel[] questTitel;
            private JLabel[] questBestanden;
            private Quest quest;
            private Dimension d;
            private JButton buttonOK;
            private int abstand;
            private int anzahl;

            private FensterQuestlog(Spieler spieler) {
                this.spieler = spieler;
                fensterQuestlog.setSize(500,340);
                fensterQuestlog.setResizable(false);
                fensterQuestlog.setVisible(true);
                fensterQuestlog.addKeyListener(this);
                fensterQuestlog.setLayout(null);
                d = Toolkit.getDefaultToolkit().getScreenSize();
                fensterQuestlog.setLocation((d.width - fensterQuestlog.getSize().width
) / 2, (d.height - fensterQuestlog.getSize().height) / 2 );
                abstand = 60;
                anzahl = 0;
                questBeschreibung = new JTextArea[15];
                questTitel = new JLabel[15];
                questBestanden = new JLabel[15];
                for (int i=0; i<6; i++) {
                    quest = spieler.getQuestliste().getQuest(i);
                    if (!quest.getStatus().equals("unbekannt")) {
                        questTitel[i] = new JLabel(quest.getName());
                      questTitel[i].setSize(380,20);
                      questTitel[i].setLocation(10,10+anzahl*abstand);
                        questBeschreibung[i] = new
JTextArea(quest.getBeschreibung());
                        questBeschreibung[i].setSize(470,40);
```

```
questBeschreibung[i].setLocation(10,30+anzahl*abstand);
                                        questBeschreibung[i].setLineWrap(true);
                                        questBeschreibung[i].setWrapStyleWord(true);
                                        questBeschreibung[i].setFocusable(false);
                                        questBeschreibung[i].setOpaque(false);
                                        questBestanden[i] = new
JLabel("",SwingConstants.TRAILING);
                                        /*
                                        if (spieler.getQuestliste().hatQuestBestanden(quest))
                                                questBestanden[i].setText("bestanden");
                                        else
                                                questBestanden[i].setText("offen");*/
                                        questBestanden[i].setText(quest.getStatus());
                                        questBestanden[i].setSize(100,20);
                                        questBestanden[i].setLocation(370,10+anzahl*abstand);
                                        fensterQuestlog.add(questTitel[i]);
                                        fensterQuestlog.add(questBeschreibung[i]);
                                        fensterQuestlog.add(questBestanden[i]);
                                        anzahl++;
                                }
                        }
                        buttonOK = new JButton("OK");
                        buttonOK.setLocation(200,260);
                        buttonOK.setSize(100,30);
                        buttonOK.setFocusable(false);
                        buttonOK.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        fensterQuestlog.dispose();
                                }
                        });
                        fensterQuestlog.add(buttonOK);
                }

                public void keyPressed(KeyEvent ke) {
                        if(ke.getKeyCode() == KeyEvent.VK_Q)
                                fensterQuestlog.dispose();
                        if(ke.getKeyCode() == KeyEvent.VK_ENTER)
                                fensterQuestlog.dispose();
                }
           public void keyReleased(KeyEvent ke) { }
           public void keyTyped(KeyEvent ke) { }
        }

        ////////////////////////////////
        // Klasse FensterMenue        //
        ////////////////////////////////

        private class FensterMenue{
                private JFrame fensterMenue = new JFrame("Heroes of SEPVenture");
                private Dimension d;
```

```
		private Client client;
		boolean serverGefunden = false;
		private JComboBox comboCharakterName;
		private JLayeredPane layerContainerMenue;
		private JPasswordField textPasswort;
		private JButton buttonBeenden;
		private JButton buttonHilfe;
		private JButton buttonErstellen;
		private JButton buttonLaden;
		private JLabel bildLogo;
		private JLabel bildHintergrund;
		private JPanel layerAnzeige;
		private JLabel labelName;
		private JLabel labelLaden;
		private JLabel labelPasswort;
		private JLabel labelErstellen;
		private String eingabe;
		private String[] listeSpieler;

		private FensterMenue(Client client){
			this.client = client;
			fensterMenue.setSize(556,482);
			fensterMenue.setResizable(false);
			fensterMenue.setVisible(true);
			fensterMenue.setLayout(null);
			fensterMenue.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			d = Toolkit.getDefaultToolkit().getScreenSize();
			fensterMenue.setLocation((d.width - fensterMenue.getSize().width ) / 2,
(d.height - fensterMenue.getSize().height) / 2 );
			layerContainerMenue = new JLayeredPane();
			layerContainerMenue.setLayout(null);
			fensterMenue.setContentPane(layerContainerMenue);
			layerAnzeige = new JPanel();
			layerAnzeige.setBounds(0,0,550,450);
			layerAnzeige.setLayout(null);
			layerAnzeige.setOpaque(false);
			comboCharakterName = new JComboBox();
			comboCharakterName.setBounds(100,290,150,20);
			comboCharakterName.setEditable(true);
			textPasswort = new JPasswordField();
			textPasswort.setBounds(100,320,150,20);
			bildHintergrund = new JLabel(new
ImageIcon("img/hintergrundStart.jpg"));
			bildHintergrund.setBounds(new Rectangle(0,0,550,450));
			bildLogo = new JLabel(new ImageIcon("img/logo.jpg"));
			bildLogo.setBounds(new Rectangle(125,20,300,225));
			labelLaden = new JLabel("Charakter laden");
			labelLaden.setBounds(85,260,100,20);
			labelName = new JLabel("Name:");
			labelName.setBounds(20,290,80,20);
			labelPasswort = new JLabel("Passwort:");
```

```
                labelPasswort.setBounds(20,320,80,20);
                labelErstellen = new JLabel("Neuen Charakter erstellen");
                labelErstellen.setBounds(340,260,150,20);
                buttonHilfe = new JButton("Hilfe");
                buttonHilfe.setBounds(60,400,150,30);
                buttonHilfe.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                zeigeDialog("Steuerung","I = Inventar\tQ = Questlog\nM = Map\tN =
Chat\nC = Charakter\tS = Sound");
                }
            });
                buttonBeenden = new JButton("Spiel Beenden");
                buttonBeenden.setBounds(340,400,150,30);
                buttonBeenden.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
                }
            });
                buttonErstellen = new JButton("Charakter erstellen");
                buttonErstellen.setBounds(340,350,150,30);
                buttonErstellen.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                        charakterErstellen();
                        fensterMenue.dispose();
                }
                catch (Exception exception) {}
            }
                });
                buttonLaden = new JButton("Charakter laden");
                buttonLaden.setBounds(60,350,150,30);
                buttonLaden.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if(comboCharakterName.getSelectedItem().equals("")) {
                        JOptionPane.showMessageDialog( null, "Bitte Spielername
eingeben!" );
                        comboCharakterName.requestFocus();
                }
                else {
                        char [] test = textPasswort.getPassword();
                                String t1="";
                                for(int i=0;i<=test.length-1;i++)
                                t1=t1+test[i];
                        int ergebnis =
getSpieler((String)comboCharakterName.getSelectedItem(),t1);
                        if (ergebnis == 0)
                                JOptionPane.showMessageDialog( null, "Fehler beim
Laden der Daten!");
                        else if (ergebnis == 1)
                                fensterMenue.dispose();
                        else if (ergebnis == 2)  {
```

```
                                        JOptionPane.showMessageDialog( null, "Falsches
Passwort!" );
                                                textPasswort.requestFocus();
                                }
                                else if (ergebnis == 3) {
                                        JOptionPane.showMessageDialog( null, "Spieler nicht
vorhanden!" );
                                        comboCharakterName.requestFocus();
                                }
                        }
                    }
                });
                        layerAnzeige.add(bildLogo);
                        layerAnzeige.add(labelName);
                        layerAnzeige.add(labelPasswort);
                        layerAnzeige.add(labelLaden);
                        layerAnzeige.add(labelErstellen);
                        layerAnzeige.add(comboCharakterName);
                        layerAnzeige.add(textPasswort);
                        layerAnzeige.add(buttonBeenden);
                        layerAnzeige.add(buttonErstellen);
                        layerAnzeige.add(buttonLaden);
                        layerAnzeige.add(buttonHilfe);

                        layerContainerMenue.add(bildHintergrund,new Integer(0));
                        layerContainerMenue.add(layerAnzeige, new Integer(100));

                        do {
                                try {
                                        eingabe = JOptionPane.showInputDialog("IP + Port des
Servers (z.B. localhost:1099):");
                                        if (eingabe == null)
                                                System.exit(0);
                                        client.setServer(eingabe);
                                        listeSpieler = client.getListeSpieler();
                                        for(String s: listeSpieler)
                                                comboCharakterName.addItem(s);
                                        serverGefunden = true;
                                }
                                catch (Exception exception) {
                                        JOptionPane.showMessageDialog( null, "Server nicht
erreichbar!\nFalsche IP?" );
                                        serverGefunden = false;
                                };
                        }
                        while (!serverGefunden);
                }

                public int getSpieler(String charakter, String passwort) {
                        try {
                                return client.getSpieler(charakter,passwort);
```

```
                }
                catch (Exception e) {
                        return 0;
                }
        }

        public void charakterErstellen() {
                zeigeErstellen(client);
        }

}


////////////////////////////////////
// Klasse FensterErstellen       //
////////////////////////////////////

private class FensterErstellen{
        private JFrame fensterErstellen = new JFrame("Neuen Charakter erstellen");
        private Dimension d;
        private Client client;
        boolean serverGefunden = false;
        private JLayeredPane layerContainerErstellen;
        private JPasswordField textPasswort;
        private JButton buttonBeenden;
        private JButton buttonLaden;
        private JLabel bildFigur;
        private JLabel bildHintergrund;
        private JPanel layerAnzeige;
        private JLabel labelName;
        private JLabel labelPasswort;
        private JLabel labelRasse;
        private JLabel labelKlasse;
        private JRadioButton rasseEngel;
        private JRadioButton rasseDaemon;
        private JRadioButton klasseKrieger;
        private JRadioButton klasseMagier;
        private JRadioButton bonusStaerke;
        private JRadioButton bonusAusdauer;
        private JRadioButton bonusGewandtheit;
        private JRadioButton bonusIntelligenz;
        private ButtonGroup groupRasse;
        private ButtonGroup groupKlasse;
        private ButtonGroup groupBonus;
        private JLabel labelStaerke;
        private JLabel labelAusdauer;
        private JLabel labelGewandtheit;
        private JLabel labelIntelligenz;
        private JLabel labelWertStaerke;
        private JLabel labelWertAusdauer;
        private JLabel labelWertGewandtheit;
```

```
		private JLabel labelWertIntelligenz;
		private JTextArea labelBeschreibung;
		private JTextArea labelBonus;
		private JTextField textName;

		private FensterErstellen(Client client){
			this.client = client;
			fensterErstellen.setSize(556,482);
			fensterErstellen.setResizable(false);
			fensterErstellen.setVisible(true);
			fensterErstellen.setLayout(null);
			fensterErstellen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			d = Toolkit.getDefaultToolkit().getScreenSize();
			fensterErstellen.setLocation((d.width - fensterErstellen.getSize().width
) / 2, (d.height - fensterErstellen.getSize().height) / 2 );
			layerContainerErstellen = new JLayeredPane();
			layerContainerErstellen.setLayout(null);
			fensterErstellen.setContentPane(layerContainerErstellen);
			layerAnzeige = new JPanel();
			layerAnzeige.setBounds(0,0,550,450);
			layerAnzeige.setLayout(null);
			layerAnzeige.setOpaque(false);
			textName = new JTextField();
			textName.setBounds(100,290,150,20);
			textPasswort = new JPasswordField();
			textPasswort.setBounds(100,320,150,20);
			bildHintergrund = new JLabel(new
ImageIcon("img/hintergrundStart.jpg"));
			bildHintergrund.setBounds(new Rectangle(0,0,550,450));
			bildFigur = new JLabel();
			bildFigur.setBounds(new Rectangle(360,30,100,200));
			labelRasse = new JLabel("Rasse:");
			labelRasse.setBounds(20,20,100,20);
			rasseEngel = new JRadioButton("Engel");
			rasseEngel.setBounds(20,50,100,20);
			rasseEngel.setOpaque(false);
			rasseEngel.setSelected(true);
			rasseEngel.addItemListener(new java.awt.event.ItemListener() {
				public void itemStateChanged(java.awt.event.ItemEvent e) {
					radioButtonsChanged();
				}
			});
			rasseDaemon = new JRadioButton("Daemon");
			rasseDaemon.setBounds(20,80,100,20);
			rasseDaemon.setOpaque(false);
			rasseDaemon.addItemListener(new java.awt.event.ItemListener() {
				public void itemStateChanged(java.awt.event.ItemEvent e) {
					radioButtonsChanged();
				}
			});
			groupRasse = new ButtonGroup();
```

```
groupRasse.add(rasseEngel);
groupRasse.add(rasseDaemon);
labelKlasse = new JLabel("Klasse:");
labelKlasse.setBounds(200,20,100,20);
klasseMagier = new JRadioButton("Magier");
klasseMagier.setBounds(200,50,100,20);
klasseMagier.setOpaque(false);
klasseMagier.setSelected(true);
klasseMagier.addItemListener(new java.awt.event.ItemListener() {
        public void itemStateChanged(java.awt.event.ItemEvent e) {
                radioButtonsChanged();
        }
});
klasseKrieger = new JRadioButton("Krieger");
klasseKrieger.setBounds(200,80,100,20);
klasseKrieger.setOpaque(false);
klasseKrieger.addItemListener(new java.awt.event.ItemListener() {
        public void itemStateChanged(java.awt.event.ItemEvent e) {
                radioButtonsChanged();
        }
});
groupKlasse = new ButtonGroup();
groupKlasse.add(klasseMagier);
groupKlasse.add(klasseKrieger);

labelStaerke = new JLabel("Staerke:");
labelStaerke.setBounds(30,140,100,20);
labelAusdauer = new JLabel("Ausdauer:");
labelAusdauer.setBounds(30,170,100,20);
labelGewandtheit = new JLabel("Gewandtheit:");
labelGewandtheit.setBounds(30,200,100,20);
labelIntelligenz = new JLabel("Intelligenz:");
labelIntelligenz.setBounds(30,230,100,20);

labelWertStaerke = new JLabel();
labelWertStaerke.setBounds(120,140,100,20);
labelWertAusdauer = new JLabel();
labelWertAusdauer.setBounds(120,170,100,20);
labelWertGewandtheit = new JLabel();
labelWertGewandtheit.setBounds(120,200,100,20);
labelWertIntelligenz = new JLabel();
labelWertIntelligenz.setBounds(120,230,100,20);

bonusStaerke = new JRadioButton();
bonusStaerke.setBounds(140,140,20,20);
bonusStaerke.setOpaque(false);
bonusStaerke.setSelected(true);
bonusStaerke.addItemListener(new java.awt.event.ItemListener() {
        public void itemStateChanged(java.awt.event.ItemEvent e) {
                radioButtonsChanged();
        }
```

```
                });
                bonusAusdauer = new JRadioButton();
                bonusAusdauer.setBounds(140,170,20,20);
                bonusAusdauer.setOpaque(false);
                bonusAusdauer.addItemListener(new java.awt.event.ItemListener() {
                        public void itemStateChanged(java.awt.event.ItemEvent e) {
                                radioButtonsChanged();
                        }
                });
                bonusGewandtheit = new JRadioButton();
                bonusGewandtheit.setBounds(140,200,20,20);
                bonusGewandtheit.setOpaque(false);
                bonusGewandtheit.addItemListener(new java.awt.event.ItemListener()
{
                        public void itemStateChanged(java.awt.event.ItemEvent e) {
                                radioButtonsChanged();
                        }
                });
                bonusIntelligenz = new JRadioButton();
                bonusIntelligenz.setBounds(140,230,20,20);
                bonusIntelligenz.setOpaque(false);
                bonusIntelligenz.addItemListener(new java.awt.event.ItemListener() {
                        public void itemStateChanged(java.awt.event.ItemEvent e) {
                                radioButtonsChanged();
                        }
                });
                groupBonus = new ButtonGroup();
                groupBonus.add(bonusStaerke);
                groupBonus.add(bonusAusdauer);
                groupBonus.add(bonusGewandtheit);
                groupBonus.add(bonusIntelligenz);
                labelBonus = new JTextArea("Tipp: Verteilen Sie einen Bonuspunkt
auf ihre Attributwerte.");
                labelBonus.setBounds(180,140,150,100);
                labelBonus.setLineWrap(true);
                labelBonus.setWrapStyleWord(true);
                labelBonus.setFocusable(false);
                labelBonus.setOpaque(false);
                labelBeschreibung = new JTextArea();
                labelBeschreibung.setBounds(320,250,200,100);
                labelBeschreibung.setLineWrap(true);
                labelBeschreibung.setWrapStyleWord(true);
                labelBeschreibung.setFocusable(false);
                labelBeschreibung.setOpaque(false);

                labelName = new JLabel("Name:");
                labelName.setBounds(20,290,80,20);
                labelPasswort = new JLabel("Passwort:");
                labelPasswort.setBounds(20,320,80,20);
                buttonBeenden = new JButton("Spiel Beenden");
                buttonBeenden.setBounds(300,400,150,30);
```

```
                buttonBeenden.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                System.exit(0);
              }
            });
                buttonLaden = new JButton("Charakter erstellen");
                buttonLaden.setBounds(100,400,150,30);
                buttonLaden.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                if(textName.getText().equals("")) {
                        JOptionPane.showMessageDialog( null, "Bitte Spielername
eingeben!" );
                        textName.requestFocus();
                }
                else {
                        int ergebnis = spielerAnlegen(CharakterSpeichern());
                        if (ergebnis == 1)
                                fensterErstellen.dispose();
                        else if (ergebnis == 2) {
                                JOptionPane.showMessageDialog( null, "Spieler bereits
vorhanden!" );
                                textName.requestFocus();
                        }
                        else if (ergebnis == 0) {
                                JOptionPane.showMessageDialog( null, "Fehler beim
Verbinden!" );
                                textName.requestFocus();
                        }
                }
              }
            });
                layerAnzeige.add(bildFigur);
                layerAnzeige.add(labelName);
                layerAnzeige.add(labelPasswort);
                layerAnzeige.add(labelRasse);
                layerAnzeige.add(labelKlasse);
                layerAnzeige.add(textPasswort);
                layerAnzeige.add(buttonBeenden);
                layerAnzeige.add(buttonLaden);
                layerAnzeige.add(rasseEngel);
                layerAnzeige.add(rasseDaemon);
                layerAnzeige.add(klasseKrieger);
                layerAnzeige.add(klasseMagier);
                layerAnzeige.add(labelStaerke);
                layerAnzeige.add(labelAusdauer);
                layerAnzeige.add(labelGewandtheit);
                layerAnzeige.add(labelIntelligenz);
                layerAnzeige.add(labelWertStaerke);
                layerAnzeige.add(labelWertAusdauer);
                layerAnzeige.add(labelWertGewandtheit);
                layerAnzeige.add(labelWertIntelligenz);
```

```
                layerAnzeige.add(bonusStaerke);
                layerAnzeige.add(bonusAusdauer);
                layerAnzeige.add(bonusGewandtheit);
                layerAnzeige.add(bonusIntelligenz);
                layerAnzeige.add(labelBeschreibung);
                layerAnzeige.add(textName);
                layerAnzeige.add(labelBonus);

                layerContainerErstellen.add(bildHintergrund,new Integer(0));
                layerContainerErstellen.add(layerAnzeige, new Integer(100));
                radioButtonsChanged();
        }

        private void radioButtonsChanged() {
                int staerke=0;
                int ausdauer=0;
                int gewandtheit=0;
                int intelligenz=0;

                if (bonusStaerke.isSelected()) {
                        staerke = 1;
                        ausdauer = 0;
                        gewandtheit = 0;
                        intelligenz = 0;
                }
                else if (bonusAusdauer.isSelected()) {
                        staerke = 0;
                        ausdauer = 1;
                        gewandtheit = 0;
                        intelligenz = 0;
                }
                else if (bonusIntelligenz.isSelected()) {
                        staerke = 0;
                        ausdauer = 0;
                        gewandtheit = 0;
                        intelligenz = 1;
                }
                else if (bonusGewandtheit.isSelected()) {
                        staerke = 0;
                        ausdauer = 0;
                        gewandtheit = 1;
                        intelligenz = 0;
                }

                if (rasseEngel.isSelected())  {
                        if(klasseMagier.isSelected()) {
                                labelBeschreibung.setText("Die Engel starten im
Himmel.\n\nMagier können Zaubersprüche sprechen.");
                                labelWertStaerke.setText(Integer.toString(10 + staerke));
                                labelWertAusdauer.setText(Integer.toString(10 +
ausdauer));
```

```
                                        labelWertGewandtheit.setText(Integer.toString(11 +
gewandtheit));
                                        labelWertIntelligenz.setText(Integer.toString(12 +
intelligenz));
                                        bildFigur.setIcon((new
ImageIcon("img/engelmagier.png")));
                                }
                                else {
                                        labelBeschreibung.setText("Die Engel starten im
Himmel.\n\nKrieger sind stark im Kampf.");
                                        labelWertStaerke.setText(Integer.toString(11 + staerke));
                                        labelWertAusdauer.setText(Integer.toString(12 +
ausdauer));
                                        labelWertGewandtheit.setText(Integer.toString(10 +
gewandtheit));
                                        labelWertIntelligenz.setText(Integer.toString(10 +
intelligenz));
                                        bildFigur.setIcon((new
ImageIcon("img/engelkrieger.png")));
                                }
                        }
                        else {
                                if(klasseMagier.isSelected()) {
                                        labelBeschreibung.setText("Die Dämonen starten in der
Hölle.\n\nMagier können Zaubersprüche sprechen.");
                                        labelWertStaerke.setText(Integer.toString(10 + staerke));
                                        labelWertAusdauer.setText(Integer.toString(10 +
ausdauer));
                                        labelWertGewandtheit.setText(Integer.toString(12 +
gewandtheit));
                                        labelWertIntelligenz.setText(Integer.toString(11 +
intelligenz));
                                        bildFigur.setIcon((new
ImageIcon("img/daemonmagier.png")));
                                }
                                else {
                                        labelBeschreibung.setText("Die Dämonen starten in der
Hölle.\n\nKrieger sind stark im Kampf.");
                                        labelWertStaerke.setText(Integer.toString(12 + staerke));
                                        labelWertAusdauer.setText(Integer.toString(11 +
ausdauer));
                                        labelWertGewandtheit.setText(Integer.toString(10 +
gewandtheit));
                                        labelWertIntelligenz.setText(Integer.toString(10 +
intelligenz));
                                        bildFigur.setIcon((new
ImageIcon("img/daemonkrieger.png")));
                                }
                        }
                }
```

```
	public int spielerAnlegen(String[] charakter) {
		try {
			return client.spielerAnlegen(charakter);
		}
		catch (Exception e) {
			return 0;
		}
	}

	private String[] CharakterSpeichern() {
		String[] tmp = new String[8];
		tmp[0] = textName.getText();
		if(rasseEngel.isSelected())
			tmp[1]="Engel";
		else
			tmp[1]="Daemon";
		if(klasseMagier.isSelected())
			tmp[2]="Magier";
		else
			tmp[2]="Krieger";
		tmp[3] = labelWertStaerke.getText();
		tmp[4] = labelWertIntelligenz.getText();
		tmp[5] = labelWertGewandtheit.getText();
		tmp[6] = labelWertAusdauer.getText();
		char [] test1 = textPasswort.getPassword();
		String t2="";
		for(int i=0;i<=test1.length-1;i++)
		t2=t2+test1[i];
		tmp[7] = t2;
		return tmp;
	}
}

////////////////////////////////////
// Klasse FensterAufstieg       //
////////////////////////////////////

private class FensterAufstieg{
	private JFrame fensterAufstieg = new JFrame ("Ihr erreicht ein neues Level!");
	private Dimension d;
	private Client client;
	boolean serverGefunden = false;
	private JLayeredPane layerContainerAufstieg;
	private JButton buttonAufsteigen;
	private JLabel bildFigur;
	private JLabel bildHintergrund;
	private JPanel layerAnzeige;
	private JLabel labelRasse;
	private JLabel labelRasseValue;
	private JLabel labelKlasse;
	private JLabel labelKlasseValue;
```

```
		private JRadioButton bonusStaerke;
		private JRadioButton bonusAusdauer;
		private JRadioButton bonusGewandtheit;
		private JRadioButton bonusIntelligenz;
		private ButtonGroup groupBonus;
		private JLabel labelStaerke;
		private JLabel labelAusdauer;
		private JLabel labelGewandtheit;
		private JLabel labelIntelligenz;
		private JLabel labelWertStaerke;
		private JLabel labelWertAusdauer;
		private JLabel labelWertGewandtheit;
		private JLabel labelWertIntelligenz;
		private JTextArea labelBeschreibung;
		private JTextArea labelBonus;


		private Spieler spieler;
		private int staerke, ausdauer, gewandtheit, intelligenz;

		private FensterAufstieg(Client client, Spieler sp){
			this.client = client;
			this.spieler = sp;

			fensterAufstieg.setSize(556,350);
			fensterAufstieg.setResizable(false);
			fensterAufstieg.setVisible(true);
			fensterAufstieg.setLayout(null);
			fensterAufstieg.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			d = Toolkit.getDefaultToolkit().getScreenSize();
			fensterAufstieg.setLocation((d.width - fensterAufstieg.getSize().width )
/ 2, (d.height - fensterAufstieg.getSize().height) / 2 );
			layerContainerAufstieg = new JLayeredPane();
			layerContainerAufstieg.setLayout(null);
			fensterAufstieg.setContentPane(layerContainerAufstieg);
			layerAnzeige = new JPanel();
			layerAnzeige.setBounds(0,0,550,450);
			layerAnzeige.setLayout(null);
			layerAnzeige.setOpaque(false);
			bildHintergrund = new JLabel(new
ImageIcon("img/hintergrundStart.jpg"));
			bildHintergrund.setBounds(new Rectangle(0,0,550,450));
			bildFigur = new JLabel();
			bildFigur.setBounds(new Rectangle(360,30,100,200));
			labelRasse = new JLabel("Rasse:");
			labelRasseValue = new JLabel(spieler.getRasse());
			labelRasse.setBounds(20,20,100,20);
			labelRasseValue.setBounds(130,20,100,20);
			labelKlasse = new JLabel("Klasse:");
			labelKlasse.setBounds(200,20,100,20);
			labelKlasseValue = new JLabel(spieler.getKlasse());
```

```
		labelKlasseValue.setBounds(330, 20, 100, 20);


		labelStaerke = new JLabel("Staerke:");
		labelStaerke.setBounds(30,140,100,20);
		labelAusdauer = new JLabel("Ausdauer:");
		labelAusdauer.setBounds(30,170,100,20);
		labelGewandtheit = new JLabel("Gewandtheit:");
		labelGewandtheit.setBounds(30,200,100,20);
		labelIntelligenz = new JLabel("Intelligenz:");
		labelIntelligenz.setBounds(30,230,100,20);

		labelWertStaerke = new JLabel();
		labelWertStaerke.setBounds(120,140,100,20);
		labelWertAusdauer = new JLabel();
		labelWertAusdauer.setBounds(120,170,100,20);
		labelWertGewandtheit = new JLabel();
		labelWertGewandtheit.setBounds(120,200,100,20);
		labelWertIntelligenz = new JLabel();
		labelWertIntelligenz.setBounds(120,230,100,20);

		bonusStaerke = new JRadioButton();
		bonusStaerke.setBounds(140,140,20,20);
		bonusStaerke.setOpaque(false);
		bonusStaerke.setSelected(true);
		bonusStaerke.addItemListener(new java.awt.event.ItemListener() {
			public void itemStateChanged(java.awt.event.ItemEvent e) {
				radioButtonsChanged();
			}
		});
		bonusAusdauer = new JRadioButton();
		bonusAusdauer.setBounds(140,170,20,20);
		bonusAusdauer.setOpaque(false);
		bonusAusdauer.addItemListener(new java.awt.event.ItemListener() {
			public void itemStateChanged(java.awt.event.ItemEvent e) {
				radioButtonsChanged();
			}
		});
		bonusGewandtheit = new JRadioButton();
		bonusGewandtheit.setBounds(140,200,20,20);
		bonusGewandtheit.setOpaque(false);
		bonusGewandtheit.addItemListener(new java.awt.event.ItemListener()
{
			public void itemStateChanged(java.awt.event.ItemEvent e) {
				radioButtonsChanged();
			}
		});
		bonusIntelligenz = new JRadioButton();
		bonusIntelligenz.setBounds(140,230,20,20);
		bonusIntelligenz.setOpaque(false);
		bonusIntelligenz.addItemListener(new java.awt.event.ItemListener() {
```

```
                public void itemStateChanged(java.awt.event.ItemEvent e) {
                    radioButtonsChanged();
                }
            });
            groupBonus = new ButtonGroup();
            groupBonus.add(bonusStaerke);
            groupBonus.add(bonusAusdauer);
            groupBonus.add(bonusGewandtheit);
            groupBonus.add(bonusIntelligenz);
            labelBonus = new JTextArea("Verteilen sie einen Attributpunkt");
            labelBonus.setBounds(180,140,150,100);
            labelBonus.setLineWrap(true);
            labelBonus.setWrapStyleWord(true);
            labelBonus.setFocusable(false);
            labelBonus.setOpaque(false);
            labelBeschreibung = new JTextArea();
            labelBeschreibung.setBounds(320,250,200,100);
            labelBeschreibung.setLineWrap(true);
            labelBeschreibung.setWrapStyleWord(true);
            labelBeschreibung.setFocusable(false);
            labelBeschreibung.setOpaque(false);

            buttonAufsteigen = new JButton("Aufsteigen");
            buttonAufsteigen.setBounds(300,280,150,30);
            buttonAufsteigen.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            spieler.setstaerke(staerke);
                        spieler.setausdauer(ausdauer);
                        spieler.setgewandtheit(gewandtheit);
                        spieler.setintelligenz(intelligenz);

            fensterAufstieg.dispose();
          }
        });


            labelWertStaerke.setText(Integer.toString(spieler.getstaerke()));
            labelWertAusdauer.setText(Integer.toString(spieler.getausdauer()));

labelWertGewandtheit.setText(Integer.toString(spieler.getgewandtheit()));
            labelWertIntelligenz.setText(Integer.toString(spieler.getintelligenz()));
            layerAnzeige.add(bildFigur);
            layerAnzeige.add(labelRasse);
            layerAnzeige.add(labelRasseValue);
            layerAnzeige.add(labelKlasse);
            layerAnzeige.add(labelKlasseValue);
            layerAnzeige.add(buttonAufsteigen);
            layerAnzeige.add(labelStaerke);
            layerAnzeige.add(labelAusdauer);
            layerAnzeige.add(labelGewandtheit);
            layerAnzeige.add(labelIntelligenz);
```

```
        layerAnzeige.add(labelWertStaerke);
        layerAnzeige.add(labelWertAusdauer);
        layerAnzeige.add(labelWertGewandtheit);
        layerAnzeige.add(labelWertIntelligenz);
        layerAnzeige.add(bonusStaerke);
        layerAnzeige.add(bonusAusdauer);
        layerAnzeige.add(bonusGewandtheit);
        layerAnzeige.add(bonusIntelligenz);
        layerAnzeige.add(labelBeschreibung);
        layerAnzeige.add(labelBonus);


        layerContainerAufstieg.add(bildHintergrund,new Integer(0));
        layerContainerAufstieg.add(layerAnzeige, new Integer(100));
        radioButtonsChanged();
}

private void radioButtonsChanged() {
        staerke=spieler.getstaerke();
        ausdauer=spieler.getausdauer();
        gewandtheit=spieler.getgewandtheit();
        intelligenz=spieler.getintelligenz();

        if (bonusStaerke.isSelected()) {
                staerke += 1;
                labelWertStaerke.setText(Integer.toString(staerke));
                ausdauer=spieler.getausdauer();
                labelWertAusdauer.setText(Integer.toString(ausdauer));
                gewandtheit=spieler.getgewandtheit();
                labelWertGewandtheit.setText(Integer.toString(gewandtheit));
                intelligenz=spieler.getintelligenz();
                labelWertIntelligenz.setText(Integer.toString(intelligenz));
        }
        else if (bonusAusdauer.isSelected()) {
                staerke=spieler.getstaerke();
                labelWertStaerke.setText(Integer.toString(staerke));
                ausdauer += 1;
                labelWertAusdauer.setText(Integer.toString(ausdauer));
                gewandtheit=spieler.getgewandtheit();
                labelWertGewandtheit.setText(Integer.toString(gewandtheit));
                intelligenz=spieler.getintelligenz();
                labelWertIntelligenz.setText(Integer.toString(intelligenz));
        }
        else if (bonusIntelligenz.isSelected()) {
                staerke=spieler.getstaerke();
                labelWertStaerke.setText(Integer.toString(staerke));
                ausdauer=spieler.getausdauer();
                labelWertAusdauer.setText(Integer.toString(ausdauer));
                gewandtheit=spieler.getgewandtheit();
                labelWertGewandtheit.setText(Integer.toString(gewandtheit));
                intelligenz += 1;
```

```
                    labelWertIntelligenz.setText(Integer.toString(intelligenz));
                }
                else if (bonusGewandtheit.isSelected()) {
                    staerke=spieler.getstaerke();
                    labelWertStaerke.setText(Integer.toString(staerke));
                    ausdauer=spieler.getausdauer();
                    labelWertAusdauer.setText(Integer.toString(ausdauer));
                    gewandtheit += 1;
                    labelWertGewandtheit.setText(Integer.toString(gewandtheit));
                    intelligenz=spieler.getintelligenz();
                    labelWertIntelligenz.setText(Integer.toString(intelligenz));
                }

                if(spieler.getRasse().equalsIgnoreCase("Engel")) {
                    if(spieler.getKlasse().equalsIgnoreCase("Magier"))
                        bildFigur.setIcon(new
ImageIcon("img/engelmagier.png"));
                    else
                        bildFigur.setIcon(new
ImageIcon("img/engelkrieger.png"));
                }
                else {
                    if(spieler.getKlasse().equalsIgnoreCase("Magier"))
                        bildFigur.setIcon(new
ImageIcon("img/daemonmagier.png"));
                    else
                        bildFigur.setIcon(new
ImageIcon("img/daemonkrieger.png"));
                }
            }


        }
}
```

5.2 Grafik (beispielhaft)

5.2.1 Figuren

Engel (Magier)

Engel (Krieger)

Teufel (Magier)

Teufel (Krieger)

Gegner

Händler

Questgeber

5.2.2 Gegenstände

Geschlossene Kiste

Offene Kiste

Gold

Rüstung 1

Rüstung 2

Rüstung 3

Schwert 1

Schwert 2

Schwert 3

Großer Trank

Kleiner Trank

Zauberstab 1

Zauberstab 2

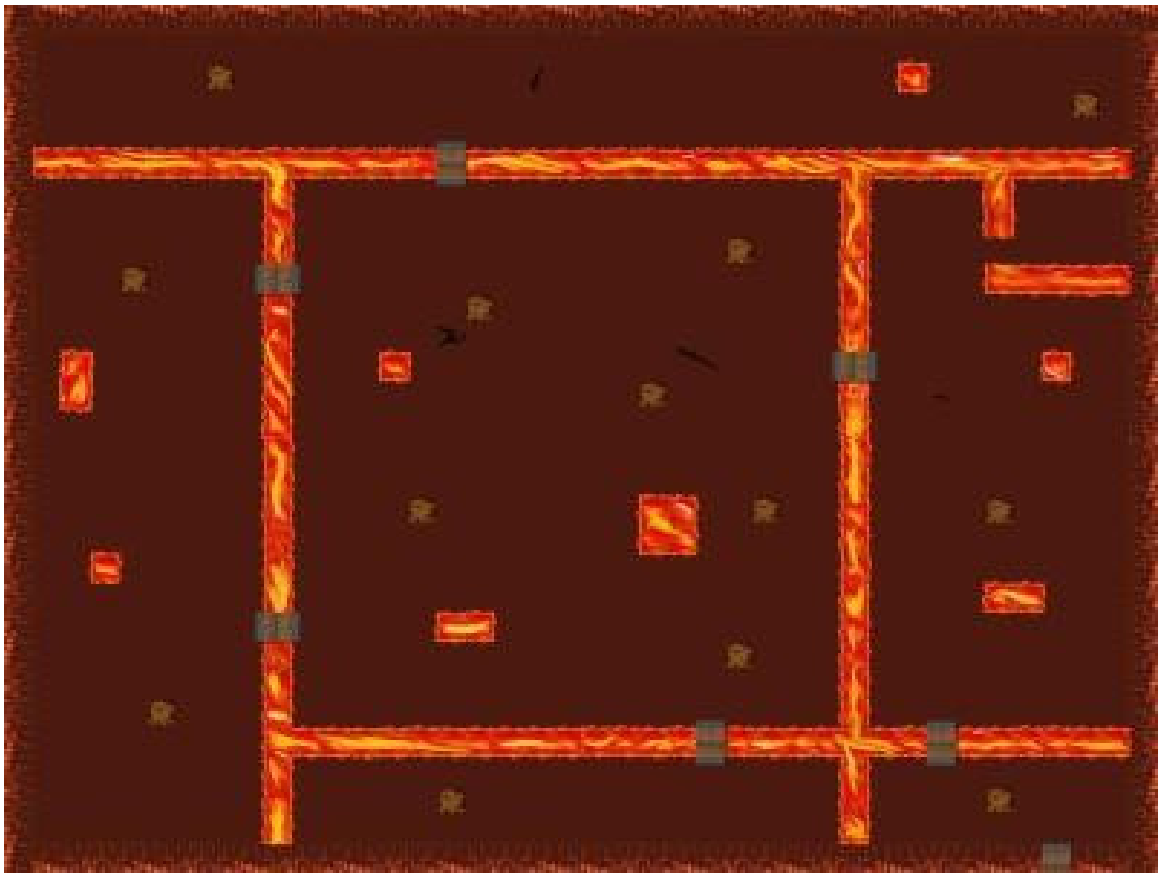
Kind

Schlüssel

5.2.3 Spielwelten

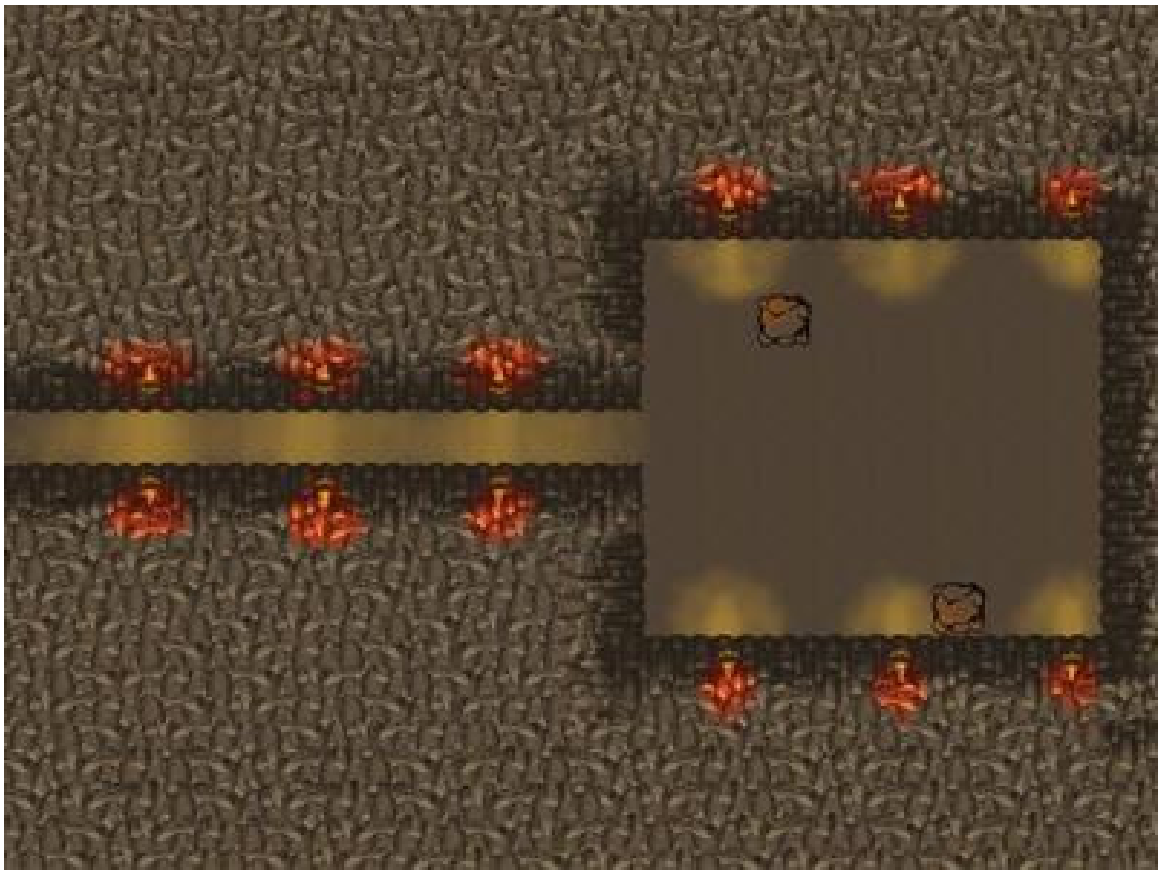
5.2.3.1 Himmel

5.2.3.2 Hölle



5.2.3.3 Erde

5.2.3.4 Geheimhöhle



6.0 Testfälle

6.1 Modultests

6.1.1 Modultest „Sound“

Testziel	Test der Soundkomponente auf allgemeine Funktionsfähigkeit
Testumgebung:	Windows XP SP2, JRE 1.5, Sound.class, Sound$Soundloop.class, Sound$sample.class, /Sounds Testklasse Soundtester.class
Vorbedingung:	Sound.java kompiliert, Soundtester.java kompiliert
Testfallszenario 1	Der Methode playSound() wird der String „kampf1“ übergeben.
Testfallszenario 2	Der Methode playSoundloop() wird der String „schrei1“ übergeben, nach 5 Sekunden wird stopLoop() aufgerufen.
Testfallszenario 3	Der Methode playSound() wird der String „notafile“ übergeben.
Testfallszenario 4	Der Rückgabewert der Methode getSound() wird ausgegeben.
Testfallszenario 5	Die Methode setSound(false) wird aufgerufen, danach wird playSoundloop() mit dem Parameter „kampf1“ aufgerufen.
Nachbedingung:	Soundtester gestartet
Kriterien:	Sounds werden korrekt abgespielt

Dokumentation der Tests:

	Soll-Verhalten	Ist-Verhalten
Testfallszenario 1	kampf1.wav wird abgespielt	Entspricht dem Soll-Verhalten
Testfallszenario 2	schrei1.wav wird 5 Sekunden lang abgespielt	Entspricht dem Soll-Verhalten
Testfallszenario 3	Komponente gibt die Meldung „notafile.wav wurde nicht gefunden!“ aus	Entspricht dem Soll-Verhalten
Testfallszenario 4	Komponente gibt „false“ aus	Entspricht dem Soll-Verhalten
Testfallszenario 5	Es passiert nichts	Entspricht dem Soll-Verhalten

Bewertung des Tests:

Die Komponente hat alle Szenarien bestanden. Deshalb kann der Test als bestanden angesehen werden.

6.1.2 Modultest „Server“

Testziel
- Test der Klasse Server

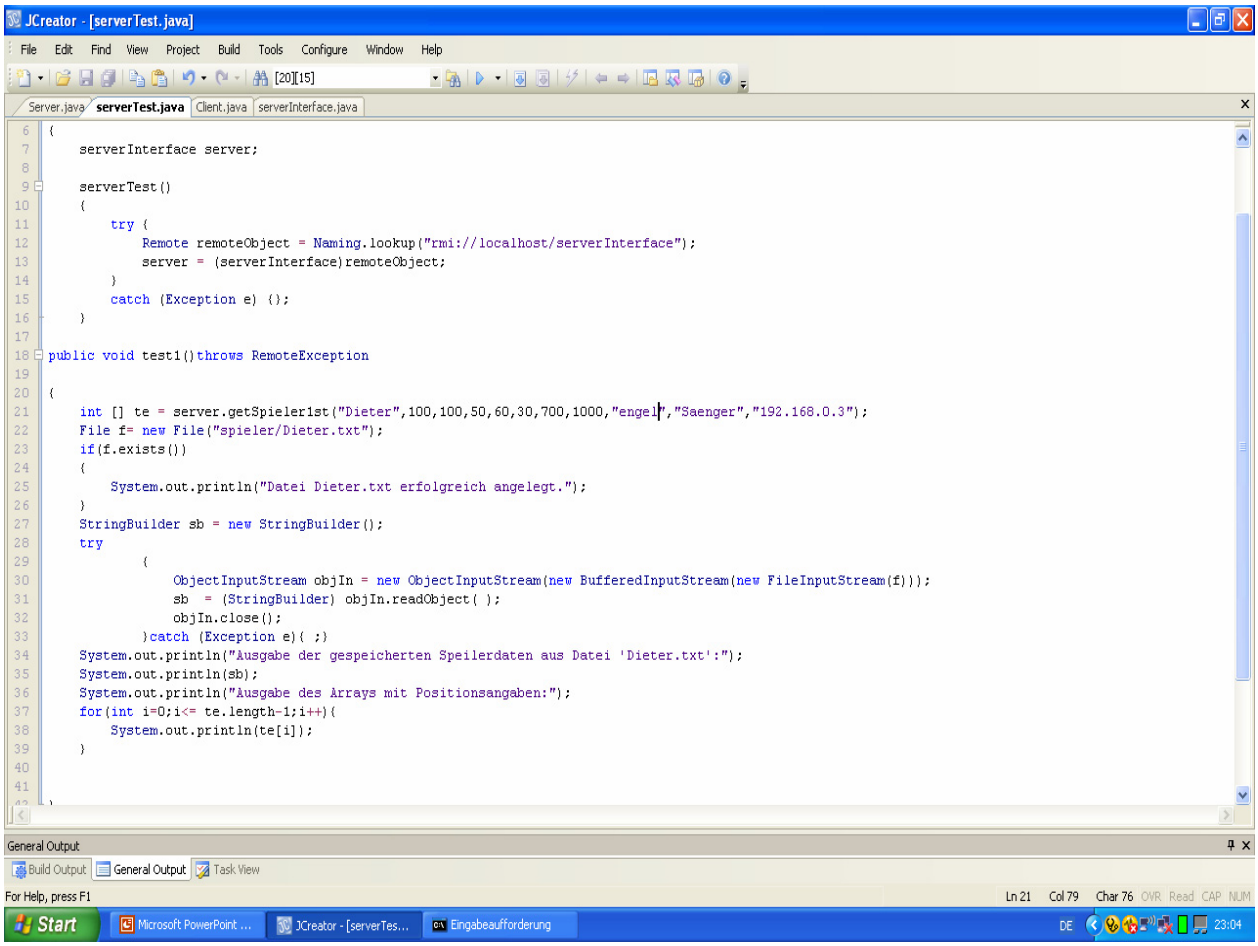
Testumgebung
- Windows XP, JRE 1.5, Server-Klasse, Servertest-Klasse

Vorbedingung
- Server initialisiert (RMI Registry gestartet, Karten geladen), noch keine Ereignisse empfangen

Szenario 1:
- Server empfängt das Ereignis „Client XY möchte neuen Charakter erstellen“
- Prüfe, ob die getSpielerfirst()-Methode
  1. die Charakterdaten richtig speichert
  2. ein neues Spieler-Objekt angelegt wird
  3. ein Array mit den richtigen Positionsangaben zurückgegeben wird

Szenario 2:
- Server empfängt das Ereignis „Client XY fordert aktuellen Spielfeldausschnitt“
- Prüfe, ob die getAktuelleKarte()-Methode die richtige Karte zurück gibt

Vorgehen:
-Programmieren einer Testklasse, die die notwendigen Initialisierungen und Methodenaufrufe durchführt.



Szenario 1:

- Tester erstellt einen neuen Charakter

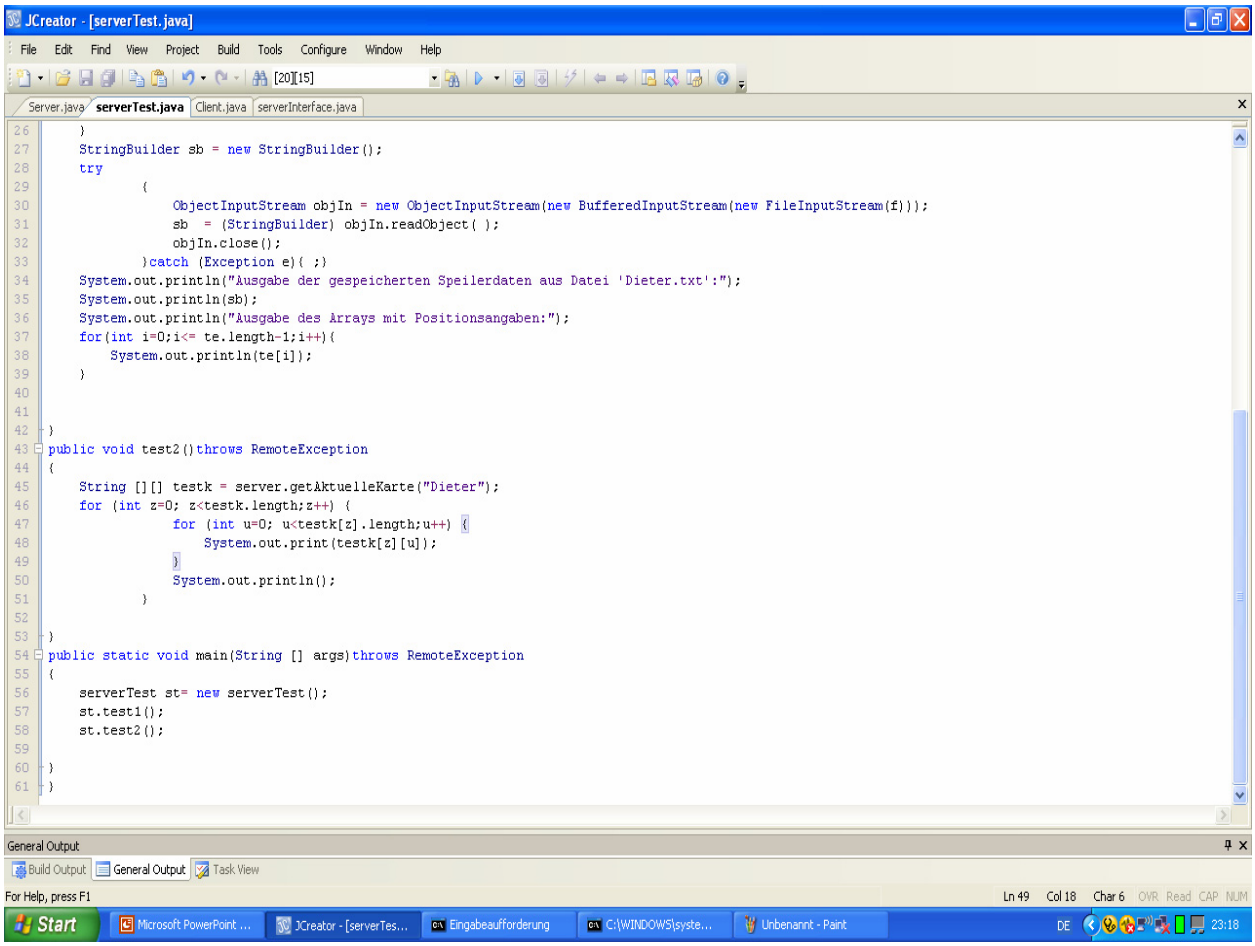
```
D:\prog_test\frank>start

D:\prog_test\frank>java serverTest
Datei Dieter.txt erfolgreich angelegt.
Ausgabe der gespeicherten Speilerdaten aus Datei 'Dieter.txt':
Dieter/100/100/50/60/30/700/1000/0/5/5/engel/Saenger/192.168.0.3/HP.Dieter.engel
.Saenger./true/false/false/true/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/
0/0/0/0/0/0/0/0/0/0/0/0/0/
Ausgabe des Arrays mit Positionsangaben:
5
5
0

D:\prog_test\frank>
```

```
D:\prog_test\frank>java server
RMI registry ready.
methoden bound
Karten geladen
hier
Dieter: in Liste der aktiven Spieler eingefuegt.
5
```

→ Test bestanden, da keine Fehler aufgetreten sind.



Szenario 2:
-Tester fordert aktuelle Karte an

→ Test bestanden, da keine Fehler aufgetreten sind.

6.2 Integrationstests

6.2.1 Integrationstest „Handel“

Testziel

- Test des Bereiches „Charakter kauft Gegenstand“

Testumgebung

- Windows XP Home SP2, JRE 1.6.0_01
- Java-Klassen Spieler.java, Haendler.java, Spielelement.java, Spielfigur.java, Preisliste.java, Gegenstand.java, Graphics.java
- Testtreiber HandelTest.java

Vorbedingungen

- Bilder kltrank.png, grtank.png, auswahl.gif und handel.jpg im Ordner /img vorhanden
- Benötigte Klassen sind kompiliert

1. Schritt: Start des Programms HandelTest

Erwartung:

- Initialisierung der benötigten Klassen Spieler, Händler und Gegenstände
- Preisliste des Händlers wird gesetzt
- Inventare des Händlers wird gefüllt
- Methode zeigeHandel der Klasse Graphics wird aufgerufen

Ergebnis:

```
1183396434218ms: Start des Programms HandelTest.java
1183396435937ms: Spieler wurde angelegt.
1183396435953ms: Haendler wurde angelegt.
1183396435953ms: Kleiner Heiltrank wurde angelegt.
1183396435968ms: Grosser Heiltrank wurde angelegt.
1183396435968ms: Preisliste des Haendlers wurde angelegt.
1183396435968ms: Kleiner Heiltrank wurde dem Haendler hinzugefuegt.
1183396435968ms: Grosser Heiltrank wurde dem Haendler hinzugefuegt.
1183396435968ms: Methode zeigeHandel der Klasse Graphics aufgerufen.
1183396436296ms: Ausgewaehlter Gegenstand wird aktualisiert.
```

→ Test bestanden

2. Schritt: Auswahl eines Gegenstandes über die Tastatur

Erwartung:

- KeyListener der Klasse Graphics wird aufgerufen
- gedrückte Taste wird überprüft
- Anzeige aktualisiert

Ergebnis:

```
1183396439796ms: KeyListener der Klasse Graphics aufgerufen.
1183396439796ms: Taste 40 gedrueckt.
1183396439796ms: Auswahlmarkierung wird versetzt.
1183396439796ms: Ausgewaehlter Gegenstand wird aktualisiert.
```

→ Test bestanden

3. Schritt: Kauf eines Gegenstandes durch Klick der Kaufen-Schaltfläche

Erwartung:

- ActionListener der Schaltfläche Kaufen wird aufgerufen
- Methode spielerWillKaufen der Klasse HandelTest wird aufgerufen
- Preis des ausgewählten Gegenstandes wird vom Händler aufgerufen
- Gold des Spielers wird aufgerufen
- Überprüfung, ob Spieler genügend Gold besitzt
- bei Erfolg Hinzufügen des Gegenstandes in das Inventar des Spielers,
- setzen des Goldes vom Spieler
- Anzeige-Fenster wird aktualisiert

Ergebnis:

```
1183396441765ms: Methode spielerWillKaufen aufgerufen.
1183396441765ms: Preis des Gegenstandes vom Haendler ausgelesen.
1183396441765ms: Gold des Spielers ausgelesen.
1183396441765ms: Ueberpruefung, ob Gold ausreichend ist.
1183396441765ms: Spieler hat genuegend Gold, Inventar ist noch nicht voll.
1183396441765ms: Gold des Spielers aktualisiert
1183396441765ms: Gegenstand dem Spieler hinzugefuegt.
1183396441765ms: Methode zeigeHandel der Klasse Graphics aufgerufen.
1183396441781ms: Ausgewaehlter Gegenstand wird aktualisiert.
```

→ Test bestanden

Testergebnis:

Der Integrationstest ist erfolgreich verlaufen. Es konnte ein Gegenstand vom Händler gekauft werden, der Gegenstand wurde dem Spieler hinzugefügt, und das Gold des Spielers wurde aktualisiert. Somit funktioniert das Zusammenspiel der getesteten Klassen.

6.2.2 Integrationstest Kampf

Testziel
- Teste Ablauf des Kampfes

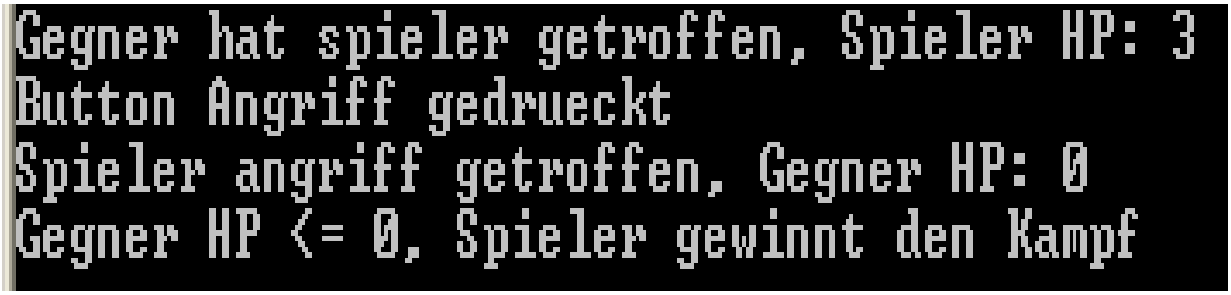
Testumgebung
- Komponente Kampf, Komponente Kampfbildschirm, Komponente Spieler und Gegner.

Vorbedingung
- Spiel läuft, Spieler steht neben einem Gegner, Spieler drückt auf die Enter-Taste

Szenario 1: Tester startet Kampf

- Tester überprüft logs ob der Gegner angreift.
- Tester überprüft logs ob die HP abgezogen werden
- Methode: Gegnerangriff wird aufgerufen, Schaden wird bei jedem Aufprall des Gegnerangriffes berechnet, Spielerlebenspunkte werden dementsprechend abgezogen.

→ Test bestanden
Szenario 2: Tester greift an
- Tester greift an
- Tester beobachtet logs, ob Angriffsbutton gedrückt wurde und ob der Angriff erfolgreich abgeschlossen wurde
- Angriffsbutton gedrückt, Methode spielerGreiftAn wird aufgerufen, Schaden wird berechnet, Schaden wird von den GegnerHP abgezogen, Abfrage ob der Kampf beendet ist.



→ Test bestanden

Testergebnis:
- Der Gegner greift nach dem Start des Kampfes immer wieder an.
Bei der Aktion Angriffsbutton wird gedrückt, wird ein Angriff gestartet, der Schaden wird berechnet und von den gegnerischen Lebenspunkten abgezogen. Abfrage ob ein Kampf beendet wurde wird gestartet, wenn die Lebenspunkte auf 0 sinken ist der Kampf beendet.

6.3 Systemtests

6.3.1 Systemtest “Handel”

Testziel

- •Test des Bereichs HandelTestumgebung
- •Windows XP Home SP2, JRE 1.6.0_01
- •Java-Klassen Spieler.java, Haendler.java, Spielelement.java, Spielfigur.java, Preisliste.java, Gegenstand.java, Graphics.java
- •Testtreiber HandelTest.javaVorbedingungen
- •Bilder kltrank.png, grtank.png, schluessel.gif, auswahl.gif und handel.jpg im Ordner /img vorhanden
- •Benötigte Klassen sind kompiliertVorgehen
- •Start der Klasse HandelTest, Kaufen und Verkaufen von Gegenständen

Testbeschreibung:

Der Charakter soll in dem Spiel die Möglichkeit besitzen, bei einem Händler Gegenstände zu kaufen und zu verkaufen. Gekaufte Gegenstände sollen dem Inventar des Charakters hinzugefügt werden, außerdem muss das Gold des Charakters um den Wert des Gegenstandes reduziert werden.
Ebenfalls soll es möglich sein, vorhandene Gegenstände im Inventar des Charakters zu verkaufen. Der Gegenstand wird dann aus dem Inventar entnommen und der Charakter erhält Gold im Gegenwert des Gegenstandes. Questgegenstände sollen nicht verkauft werden können.

Aktion: Start des Java-Programms HandelsTest.java
Erwartung: Anzeigen des Handelsfensters, Gegenstände des Charakters sowie sein Goldvorrat sollten zu sehen sein
Ergebnis: Handelsfenster wird angezeigt, auf der linken Seite das Inventar sowie das Gold des Spielers, und auf der rechten Seite das Inventar des Händlers. → Test bestanden

Aktion: Auswahl eines Gegenstandes
Erwartung: Per Maus oder Tastatur soll ein Gegenstand auswählbar sein
Ergebnis: Gegenstände können direkt per Mausklick ausgewählt werden. Drückt man die Pfeiltasten der Tastatur, bewegt sich die Auswahlmarkierung entsprechend. In der Mitte des Bildschirms werden Infos zum aktuellen Gegenstand angezeigt. → Test bestanden

Aktion: Kauf eines Gegenstandes
Erwartung: Ist ein Gegenstand im Inventar des Händlers ausgewählt und wird die Kaufen-Schaltfläche gedrückt, wird der Gegenstand dem Inventar des Charakters hinzugefügt, und sein Goldvorrat wird um den Wert des Gegenstandes verringert.
Ergebnis: Beim Klick auf Kaufen wird die Anzeige aktualisiert, neuer Gegenstand ist im Inventar des Charakters, auch das Gold wird aktualisiert. → Test bestanden

Aktion: Kauf eines Gegenstandes, aber Charakter hat nicht genügend Gold
Erwartung: Der Gegenstand darf nicht dem Inventar des Charakters hinzugefügt werden. Es sollte eine Warnmeldung für den Spieler erscheinen.
Ergebnis: Beim Klick auf Kaufen passiert nichts, der Gegenstand wird nicht hinzugefügt, es wird dem Charakter auch kein Gold abgezogen. Es wird jedoch keine Warnmeldung angezeigt. → Test teilweise bestanden

Aktion: Wechsel zwischen Kauf- und Verkaufs-Modus
Erwartung: Es soll möglich sein, zwischen dem Kaufen und Verkaufen von Gegenständen zu wechseln, ohne den Bildschirm zu verlassen.
Ergebnis: Mit einem Klick auf Wechseln wird das Inventar des Charakters fokussiert, die Auswahlmarkierung zeigt den ersten Gegenstand im Inventar. Auch ein Mausklick ins Inventar des Charakters bewirkt einen Wechsel des Modus.
→ Test bestanden

Aktion: Verkauf eines Quest-Gegenstandes
Erwartung: Verkauf sollte nicht möglich sein, Verkaufen-Schaltfläche sollte nicht sichtbar sein.
Ergebnis: Wählt man einen unverkäuflichen Gegenstand aus, so wird die Verkaufen-Schaltfläche nicht angezeigt. Es gibt keine Möglichkeit, den Gegenstand zu

verkaufen. Dem Spieler wird angezeigt, dass der aktuell ausgewählte Gegenstand unverkäuflich ist. → Test bestanden

Aktion: Auswahl eines verkäuflichen Gegenstandes
Erwartung: Verkauf des Gegenstandes sollte möglich sein, Anzeige des Preises und der Verkaufen-Schaltfläche.
Ergebnis: Bei Auswahl eines verkäuflichen Gegenstandes werden die aktuellen Informationen (samt Verkaufspreis) in der Bildschirm-Mitte angezeigt. Auch die Verkaufen-Schaltfläche ist sichtbar. → Test bestanden

Aktion: Verkauf eines Gegenstandes
Erwartung: Der ausgewählte Gegenstand wird aus dem Inventar des Spielers entfernt, der Verkaufspreis wird dem Charakter gutgeschrieben.

Ergebnis: Beim Klicken der Verkaufen-Schaltfläche wird die Anzeige direkt aktualisiert. Der Gegenstand verschwindet aus dem Inventar des Charakters, der Verkaufspreis wird dem Spieler hinzugefügt. → Test bestanden

Testergebnis:
Der Bereich des Handelns zwischen einem Charakter und einem Händler funktioniert entsprechend den Anforderungen. Jeder Testfall wurde erfolgreich durchgeführt, es gab nur eine Anmerkung, die jedoch keinen Einfluss auf die Spiellogik sowie den Spielspaß ausübt und somit als unkritisch eingestuft wird.

6.3.2 Systemtest „Kampf“

Testziel
- Test des Bereiches Kampf

Testumgebung
- Java Klassen: Kampfbildschirm.java, Spieler.Java, Gegner.Java
- Windows XP Home SP2, Java 1.5:

Vorbedingung
- Kampf wurde gestartet.
- man sollte darauf achten, dass die eigenen Lebenspunkte nicht auf 0 oder unter 0 sinken, da der Kampf dadurch beendet wird.

Szenario 1: Der Tester betätigt Button: „Angriff“
Szenario 2: Der Tester betätigt Button: „Kl. Heiltrank“
Alle Szenarien: Der Tester überprüft anhand der Grafiken was passiert

Szenario 1: Angriff

Aktion: Tester klickt auf den Button „Angriff“

Erwartung: Spieler greift an, Gegner verliert berechnete Lebenspunkte, falls diese auf 0 oder unter 0 sinken, ist der Kampf vorbei, sonst werden die neuen Lebenspunkte des Gegners unten rechts angezeigt. Alle Buttons werden deaktiviert, ATB Leiste steigt an, wenn sie voll ist, ist der Tester wieder am Zug, und die Buttons sind wieder aktiviert.

Szenario 2: Heiltrank benutzen

Aktion: Tester klickt auf Button „kl. Heiltrank“

Erwartungen: Falls ein kleiner Heiltrank vorhanden ist: Spieler wird geheilt, Lebenspunkte werden 20 HP hochgesetzt, und ein kleiner Heiltrank verschwindet aus dem Inventar. Wenn kein Heiltrank vorhanden ist, erscheint eine Meldung, dass keine Heiltränke vorhanden sind.