Projekt
„404 – Name Not Found“

Team:

Theresa Artus, Torsten Bosecker,

Marika Friedel, Peter Weigand

31. Juli 2015

Projektaufgabe Programmieren 4

In der Angewandten Informatik

Inhalt

1. Projektziel

Unser Ziel war die Erweiterung des Point-And-Click-Adventures um eine grafische Oberfläche und eine Datenbankanbindung. Weiterhin sollte das ganze Projekt noch einmal überarbeitet und dabei auf das MVC-Modell umgestellt werden.

2. Spielprinzip

Das Spielprinzip ist das eines Point-And-Click Adventures, mit der Maus wird ein Objekt in der Spielwelt angesteuert und mit einem Klick die Aktion ausgelöst.

Schwerpunkt des Spiels ist es im Spielverlauf verschiedene Gegenstände zu finden und diverse Kombinationsrätsel zu lösen. Die Spielfigur wird mit den Cursor-Tasten durch das Spielgeschehen gesteuert und per Mausklick werden Anweisungen gegeben. Diese sind „Benutze“, „Gib“, „Info“, „Rede mit“ und „Nimm“.

In der Spielwelt findet man unter anderem verschiedene Gegenstände, Objekte und Charaktere, die man verwenden oder mit welchen man zusammenarbeiten muss um im Spiel voran zu kommen.

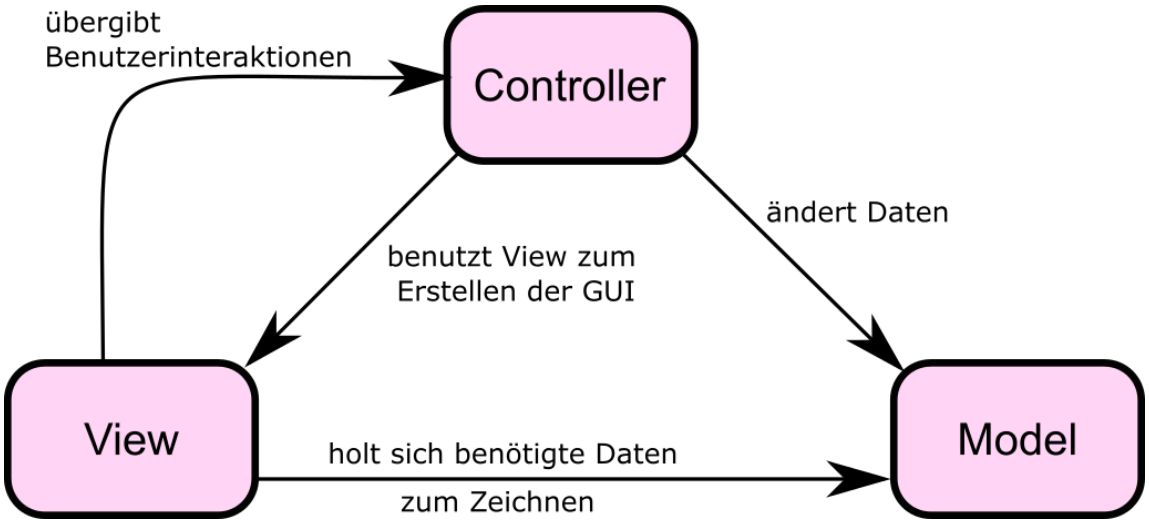
Ziel ist es am Ende alle Rätsel zu lösen und die Handlung damit erfolgreich zu beenden.

Das Spiel besteht aus einer Welt(-karte). In der Spielwelt befinden sich neben dem Spieler weitere Personen mit denen der Spieler reden und von denen er Gegenstände bekommen kann (Klasse „Character“), die Gegenstände die der Spieler aufnehmen und zum Teil miteinander kombinieren kann (Klasse „Item“), und die Vordergrund-Objekte (Klasse „Object“) wie z.B. Zäune, Häuser, Bäume. Gegenstände können zum Teil mit Objekten benutzt werden, z.B. ein Schlüssel (Item) mit einer Tür (Object) um diese zu öffnen, oder ein Schlüssel (Item) und eine Truhe (Object) um sie zu öffnen und ihren Inhalt zu bekommen. Für sämtliche Objekte in der Spielwelt wurde eine Kollissionsabfrage implementiert.

Der Spieler besitzt ein Inventar (Klasse „Inventory“), in diesem werden alle Gegenstände abgelegt die der Spieler in der Spielwelt aufgehoben hat, oder von einer anderen Person erhalten hat. Diese Gegenstände können dann z.B. mit Objekten benutzt werden oder miteinander kombiniert werden.

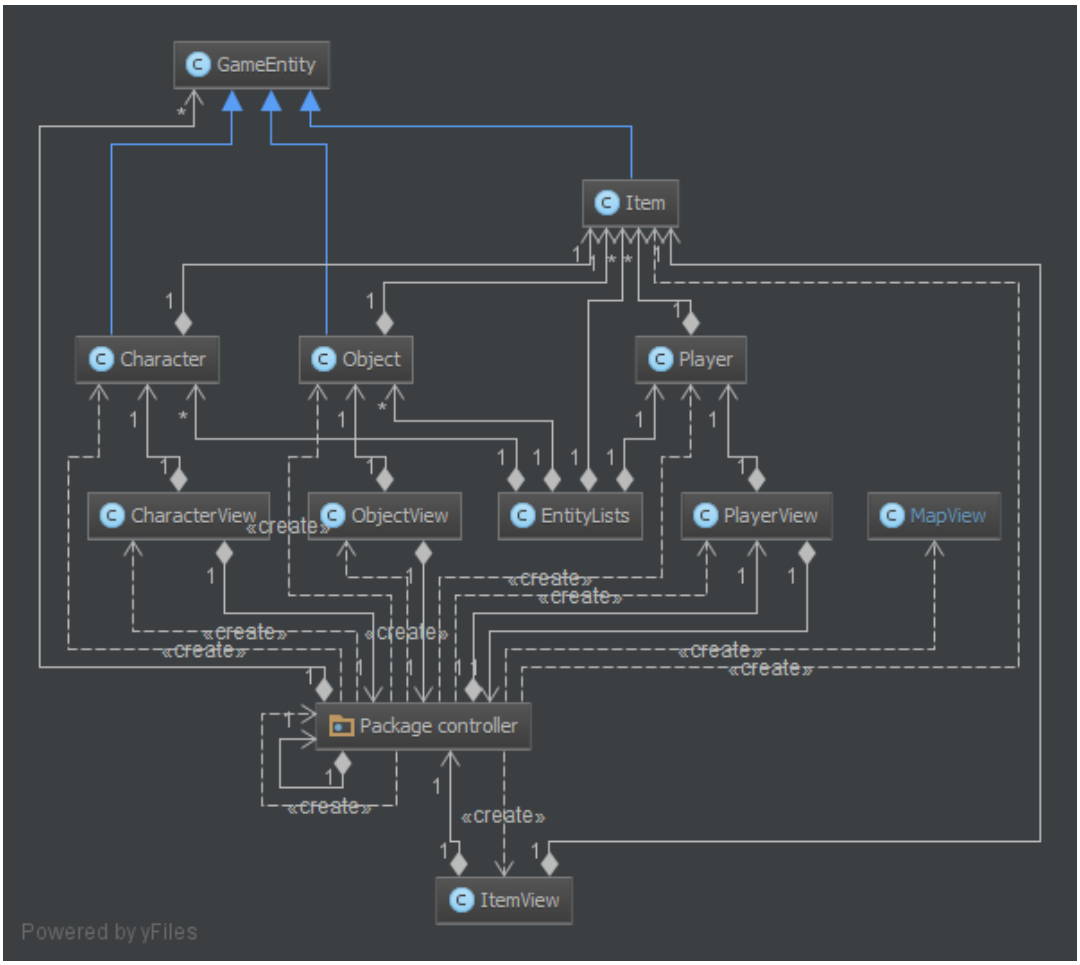
3. Umstellung auf MVC-Modell

Der generelle Aufbau eines MVC-Modells sieht folgendermaßen aus:



Um das zu erreichen, war es notwendig die Klassen des Daten-Modells, die wir in Programmieren 3 erstellt haben, noch einmal zu überarbeiten. Unter anderem war es notwendig alle Methoden die mit der Ausgabe zu tun hatten aus dem Daten-Modell zu entfernen und sie in die Views zu packen.

Am Ende ergab sich folgender Aufbau unseres Programmes (ohne Test-Klassen):



Die Spiele-Entitäten (item, object, character) sind jetzt alle von einer gemeinsamen Basisklasse abgeleitet (GameEntity), dadurch wurde nicht nur die Code-Redundanz verringert, es ist nun auch möglich alle Entitäten in einer gemeinsamen Liste zu speichern.

Die Klasse „EntityLists“ speichert alle Entitäten der Spielwelt (Gegenstände, Objekte, Personen und auch den Spieler selbst) und ermöglicht einen globalen Zugriff auf sie (als Singleton implementiert).

4. Datenbankanbindung

Für die Datenbankanbindung entschieden wir uns die Java Persistence API zu verwenden, anstatt selber die SQL-Befehle an die Datenbank zu schicken. Als Implementierung der JPA wählten wir EclipseLink und als SQL-Datenbank entschieden wir uns für H2.
Die Aufgabe der Datenbankanbindung besteht darin, eine Möglichkeit zu bieten den Spielstand zu speichern und zu einem späteren Zeitpunkt an derselben Stelle wieder einzusteigen.

Dazu werden alle Instanzen aus dem Daten-Modell in die Datenbank gespeichert, und später wieder geladen (um den vorherigen Zustand des Spiels wieder herstellen zu können), also quasi eine Art Quick-Save-Funktion.
Neben einer entsprechenden Controller-Klasse war es dazu weiterhin notwendig, die Klassen des Daten-Modells um entsprechende Annotationen zu erweitern.

5. Grafische Oberfläche

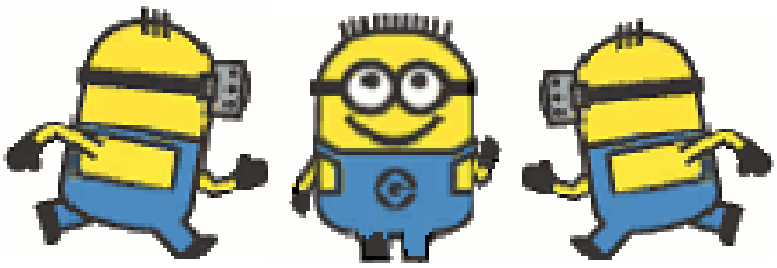
Zur Erstellung der grafischen Oberfläche entschieden wir uns für JavaFX. Die Umsetzung fand vollständig „von Hand“ statt, ohne Zuhilfenahme des JavaFX Scene Builder. Die Implementierung fand hauptsächlich in den Views und zum Teil auch im Controller statt. Im Controller findet die gesamte Initialisierung der JavaFX-Stage statt. Weiterhin werden hier auch sämtliche Keyboard-Events implementiert und an die entsprechenden grafischen Elemente gebunden. Das Gleiche passiert mit den Mouse-Events in den Views.

Unsere grafische Oberfläche sieht im momentanen Stadium so aus:

Das eigentliche Geschehen findet auf dem Spielfeld (1) statt. Die Spielfigur lässt sich mit den Cursor-Tasten steuern. Die Interaktion mit den anderen Entitäten der Spielwelt und den Buttons erfolgt mit der Maus. Darunter (2) sind die Buttons für die typischen Aktivitäten eines Point-and-Click-Adventures („Benutze“, „Gib“, „Info“, „Rede mit“ und „Nimm“). Daneben befinden sich noch die beiden Buttons zum Speichern und Laden eines Spielstandes. Die Textausgabe erfolgt im Bereich darunter (3). Rechts vom Spielfeld (4) befindet sich noch das Inventar, hier werden alle Gegenstände angezeigt, die sich gerade im Inventar des Spielers befinden.

Als Hauptfigur und für die sonstigen vorkommende Figuren wählten wir Minions (aus dem Film „Ich einfach Unverbesserlich“). Wir erstellten für die Hauptfigur eine Animationsabfolge, die nicht unbedingt übertrieben realistisch, aber ziemlich lustig wirkt.

Als Beispiel drei Animationsschritte der Hauptfigur:



Des Weiteren kamen Grafiken für die Welt, Objekte, Items und andere Charaktere mit denen im Spiel interagiert werden kann hinzu.

6. Schwierigkeiten

Neu war für uns alle die Programmierung grafischer Oberflächen, besonders die ereignisbasierte Programmierung per JavaFX. Damit war ein zeitaufwendiges Lernen und Ausprobieren verbunden. Auch die Datenbankanbindung per JPA war für uns neu, es gibt da viele Fallstricke zu überwinden bis die Anbindung letztendlich korrekt funktioniert.

Im Laufe des Projekts stießen wir auf ein paar „Eigenarten“ unserer Programmiersprache, wir wissen nun, dass z.B. die Namen „character“, „object“ und „entity“ von Java vordefiniert sind und nicht für Klassennamen verwendet werden sollten.

Erwähnenswert auch die Koordination der Gruppe, nicht immer konnte man sofort die eine, perfekte Lösung finden, man musste sich zunächst einig werden und evtl. auch Kompromisse finden.

7. Fazit

Alles in Allem würden wir behaupten, dass wir bei diesem Gruppenprojekt nicht nur das eine oder andere, sondern wirklich eine Menge dazugelernt haben. Dies betrifft nicht nur die Programmiersprache Java, wir sind der Meinung, dass gerade die Gruppenarbeit uns gut auf das Berufsleben vorbereitet hat. Besonders toll fanden wir, dass uns auch so viele Freiheiten bezüglich des Themas gelassen wurden, so entfiel ein Großteil des „Zwangs“, den man sonst oft bei Prüfungsleistungen verspürt, aber indem wir so viel selbst entscheiden durften, haben wir viel mehr „Liebe“ in das Projekt gesteckt.