

Game of Zuul

Gruppe 10

Entwicklerdokumentation

Savchuk Dmytro, Maksat Rysbekov

Inhaltsverzeichnis

Vorwort	3
Struktur des Projekts, Game	3
Room	4
Player	5
Parser	5
Command	6
ConcreteCommand	6
Alertbox	7
View	7
Model	7
JUnit Test	8

Vorwort

Game of Zuul wurde im Rahmen der Objektorientierte Softwareentwicklung entwickelt. Dieses Dokument richtet auf diejenigen, die das Projekt weiter entwickeln werden.

Struktur des Projekts

Game

Diese Klasse erstellt Räume mit Items, setzt Ausgänge ein, bearbeitet Ereignisse, macht Veränderungen in der Karte, also wird je nach dem wohin der Player geht, außerdem erneuert die Liste von Item in „Your Bag“ und „Items in Room“, sichert und loadet das Spiel. Game ist Controller zwischen View und Model.

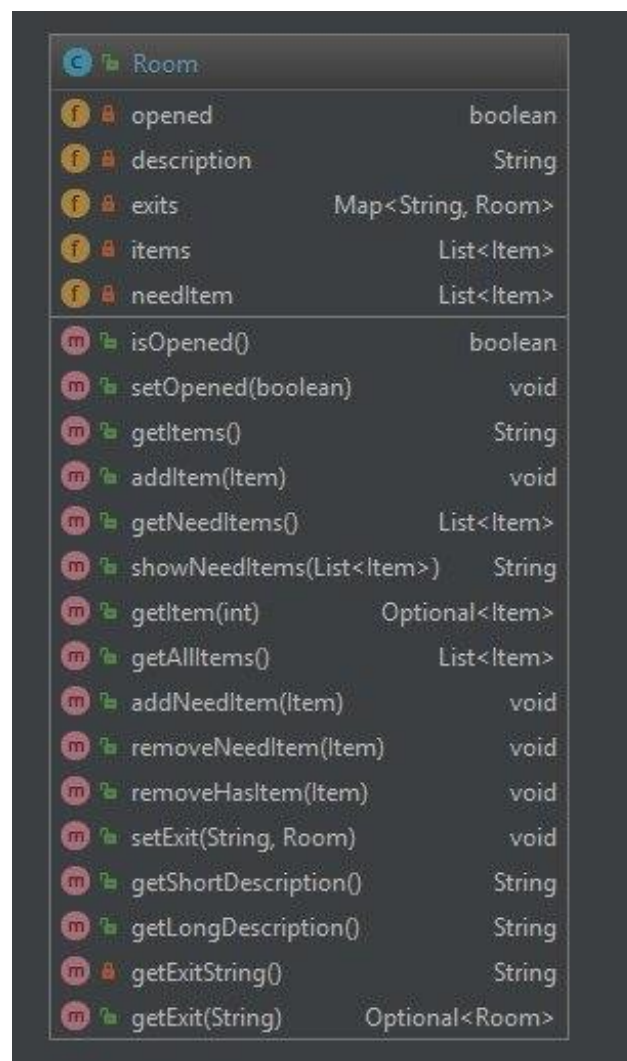


Game	
parser	Parser
player	Player
dragonRoom	Room
corridor	Room
castleGate	Room
kitchen	Room
library	Room
hallway	Room
hall	Room
princess	Room
buttonRooms	List<JButton>
view	View
createRooms()	void
play()	void
printWelcome()	String
actionPerformed(ActionEvent)	void
load()	void
save()	void
updateBagList()	void
updateItemsList()	void
setRoomsColor(Room)	void
valueChanged(ListSelectionEvent)	void

Room

Klasse Room enthält Item, Ausgänge, liefert Beschreibungen zurück, wie der Raum heißt, wo der Player gerade sich befindet und welche Ausgänge es gibt.

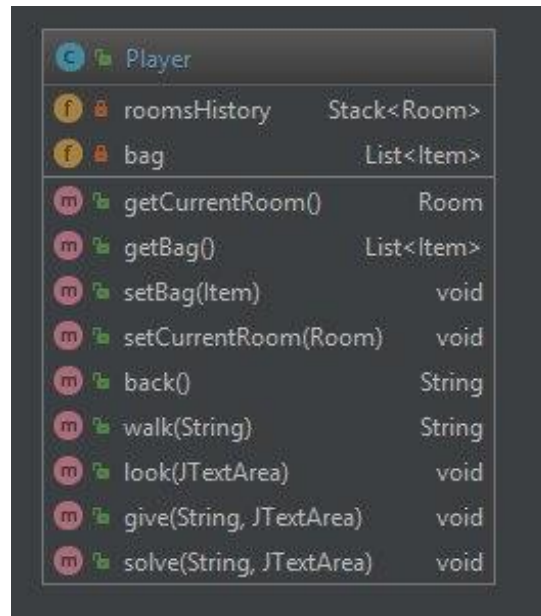
Durch diese Klasse werden Ausgänge eingesetzt.



Room		
f	opened	boolean
f	description	String
f	exits	Map<String, Room>
f	items	List<Item>
f	needItem	List<Item>
<hr/>		
m	isOpen()	boolean
m	setOpened(boolean)	void
m	getItems()	String
m	addItem(Item)	void
m	getNeedItems()	List<Item>
m	showNeedItems(List<Item>)	String
m	getItem(int)	Optional<Item>
m	getAllItems()	List<Item>
m	addNeedItem(Item)	void
m	removeNeedItem(Item)	void
m	removeHasItem(Item)	void
m	setExit(String, Room)	void
m	getShortDescription()	String
m	getLongDescription()	String
m	getExitString()	String
m	getExit(String)	Optional<Room>

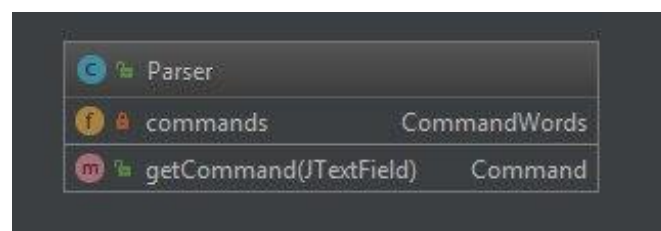
Player

Die Klasse Player enthält ein Stack, wo wird gespeichert, wo der Player war. Außerdem hat eine Liste von Item, also Item die der Player ins Rucksack gelegt hat. Durch diese Klasse kann der Player gehen, durch diese Klasse erfahren wir die Lage von Player, mithilfe der Methode give legt der Player das Item ab. Die Methode unterscheidet, ob der Player das Rätsel gelöst hat.



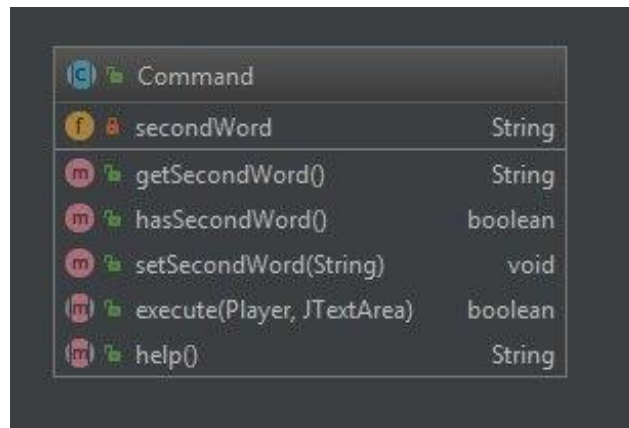
Parser

Diese Klasse ist die einfachste, gibt ein Kommando zurück, wenn der eingegebene Text stimmt, wenn nicht dann Objekt von Optional.



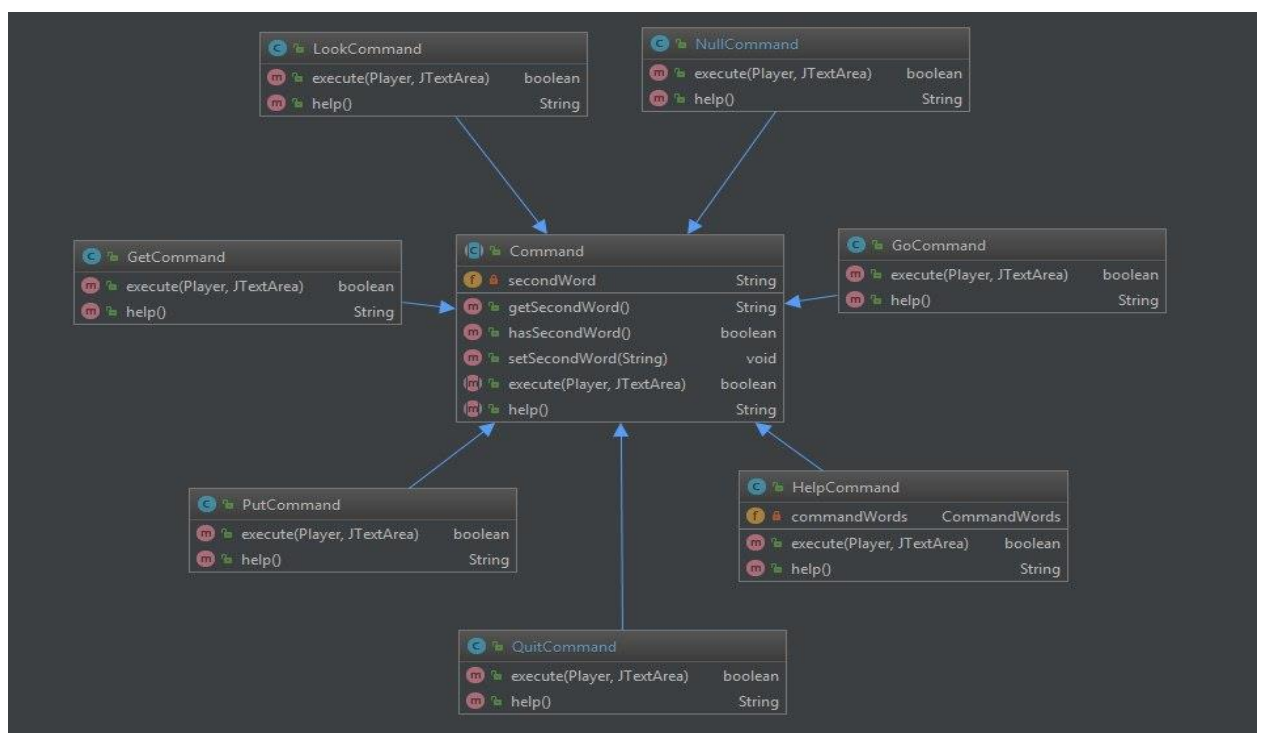
Command

Das ist eine abstrakte Klasse für alle Kommandos wie LookCommand.



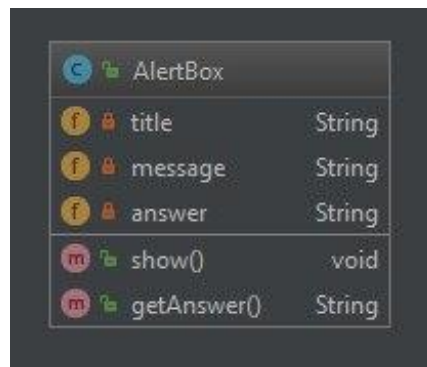
Concrete Command

Hier es wurde gemeint alle Klasse die von Command erben. Folgendes Diagramm zeigt anschaulich, dass alle sind gleich und implementieren die Methode execute.



AlertBox

Diese Klasse macht nur zwei Dingen: erstellt Dialogfeld für Rätsel und gibt eine Antwort zurück, die von Nutzer eingegeben wurde.

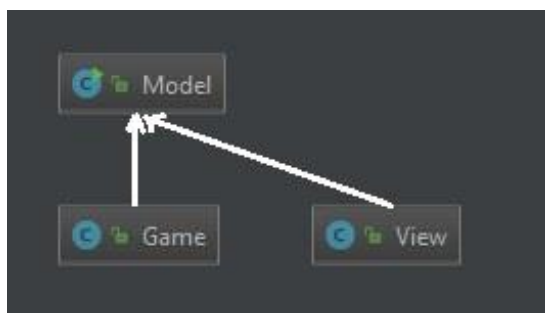


View

Die Klasse View erstellt all Benutzeroberfläche, alles was auf dem GUI steht.

Model

Model verbindet den Controller, das heißt Game mit View.



JUnit Test

In der Klasse Init.java es wurde „simuliert“, dass das Spiel gestartet wurde und man steuert, um zu prüfen, ob man im richtigen Ort sich befindet. Beim anderen ist es gleich, wir testen, ob die Ausgänge übereinstimmen.