Grundlegend liegt die JavaDoc für das Projekt in den Ressourcen im Ordner „doc“. Dort finden sie Informationen zu allen Vorhanden Klassen & Methoden.

Neue Kommandos Hinzufügen:
Um einen neuen Kommando zu erstellen ist folgendes zu beachten. Die Klasse die den Kommando darstellt sollte die Basisklasse „Command“ haben. Des weiteren sollte das Kommando in der Methode „initiale Commands()“ der Klasse „CommandWords“ eingetragen werden.

!!!Achtung!!! Bei der GUI muss beachtet werden das Genauso viele Kommandos wie die dazugehörigen Buttons vorhanden sind.

GameplayRules:
Sind Spezialisierungen des recht sehr Komplexen Nutz-Kommandos und sie werden über der Klasse „UseableCreator“ definiert. Sie greifen bei Ausführung auf GameObjecte zurück → In den GameplayRules benutzte Namen müssen auch als GameObject vorhanden sein damit diese Ausgeführt werden.

Levelerweiterung:
Die Levelerweiterung wird einfach in der Klasse „LevelCreator“ erzeugt. Zuerst erstellt man die Räume die man brauch.

```
raumname = new Room("Raumbezeichnung","Raumbeschreibung",Playersize.größe);
```

Sollte die normale Größe zum betreten des Raumes notwendig sein kann Playersize weggelassen werden.

Dann sollte man die Verknüpfungen mit den anderen Räumen anfangen.

```
raumname.setExit("richtung", angrenzenderRaum);
```

Danach können auf Wunsch Türen abgeschlossen werden.

```
raumname.setClosed(LockedStatus.LOCKED);
```

Gegenstanderweiterung:
Um neue Gegenstände hinzufügen benutzt man auch hier wieder den LevelCreator. Folgendes ist hiermit erreichbar. Die letzten 3 Werte sind Boolean werte.

```
GameObject gegenstandname=
      new GameObject("Bezeichnung","Beschreibung",Aufnehmbar?,Ansprechbar?,Sichtbar?);
```

Dann muss man nur noch die Gegenstände verteilen.

```
raumName.itemStore(gegenstandName);
```

!!!Achtung!!! Bei der Planung der Levels ist zu berücksichtigen das der Player nur 6 Items auf einmal aufnehmen kann aber keines ohne Grund ablegen kann.

!!!Achtung!!! Bei der GUI brauch man das passende 64x64 .png dazu sonst Würde an der Stelle das Spiel Abstürzen. Daher immer dran denken die Bilder zu den Gegenstände mit hochzuladen.

Gesprächsystem:

Das Gespräch setzt sich aus eine Art Baum zusammen. Am Anfang ist nur ein Kurzer Satz mit Begrüßung und je nach dem Welche Antwort man wählt bekommt man eine Gegenantwort. Mit wiederum anderen Antwortmöglichkeiten. Neue Dialoge müssen in den Mutterdialog eingetragen werden. Schleifendialoge sind möglich, damit man zum Ausgangsdialog zurückkehren kann, indem der Wurzeldialog in einen anderen beliebigen eingetragen wird.
Bsp. Man wählt Antwort 1. Und bekommt 2 neue Antworten. Also Antwort 1u1 und 1u2. Und so weiter wird es aufgesplittet.

Observer system:

Die Verbindung des Spiels mit der GUI Läuft über das Observer-Modell ab. Es kann problemlos andere GUI angebunden werden. Die Weitergabe der Daten ist über Strings realisiert.
Es sollte bei Weiterentwicklungen das Obsever-Modell beibehalten werden, da so GUI und Game in unabhängigen Threads problemlos arbeiten.

Testklassen:

Die Testklassenstruktur ist die gleiche wie in den Richtigen Ordnern. Zuerst werden bei den Tests geprüft ob die Rückgabewerte die der erwarteten Werte gleichen. Sollte es so sein, Vertiefen sich die Tests um zu prüfen ob sich die Daten im Programm wirklich geändert haben.
Die momentanen Tests sind an die momentanen Spielinhalte angepasst und müssen bei Veränderung , jedoch nicht bei Erweiterung, erneuert werden.

Weitere Informationen:

Programmiersprache: Java
IDE: Eclipse
Java Version: Java7

Anhang:

UML
Map der Aktuellen Welt.