Projekt: ZorkGame2.0 M320 Java

# Inhaltsverzeichnis:

* Die Idee
* Die Skizze
* Klassenaufteilung
* Dokumentation

# Die Idee

Die Idee des Spiels geht von der Grundlage des abgegebenen Codes aus mit den Klassen:

Game, Room, Parser, Command und Commandwords.

Die Klassen Parser und Commandworkds wurden entfernt und direkt in den Klassen miteinbezogen.

# Ein Bild, das Text, Diagramm, Rechteck, Plan enthält. Automatisch generierte BeschreibungDie Skizze

Die Skizze besteht aus 7 Räumen und man kann sich mit NORTH, EAST, SOUTH, WEST bewegen, je nachdem welche Richtungen im jeweiligen Raum zur Verfügung stehen. Die roten Räume gibt es nicht! Im Büro wird das Spiel am Schluss gewonnen.

# Klassenaufteilung

In diesem Game gibt es 6 Klassen:

Game -> Die Spiellogik inkl. Der Schleife, bis man das Spiel beendet oder Gewonnen hat.

Room -> Die Eigenschaften und Baupläne der Räume.

Item -> Eine eigene Klasse für die Gegenstände (ausgenommen Keys)

OutputHandler -> Für die meisten Textnachrichten zuständig

StartGame -> von hier wird eine Instanz gestartet und die endlosschleife aufgerufen.

ConsoleGameMap -> Eine Simple Karte die nur currentRoom als Variable von anderen Klassen benötigt.

# Dokumentation

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte BeschreibungAls erstes haben wir die Klasse Room erstellt damit wir den Bauplan für die Klassen erstellen können. Anschliessend kommen die Variablen für die Klasse Game. Im Bild, der Konstruktor für die Beschreibung der Räume.

Anschliessend die Methode, um die möglichen Ausgänge festzulegen:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Anschliessend folgen verschiedene Methoden, um die Interaktionen zu ermöglichen. Eine der wichtigsten davon ist die folgende Methode, die überprüft, ob die jeweilige Aufgabe eines Raumes erfüllt ist oder nicht.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungDie Methode «exitString» wird bei jeder Bewegung ausgeführt und zeigt die möglichen Ausgänge an:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungFolgende Variablen gibt es in der Klasse Room:

Die restlichen Methoden wie z.B: goRoom() und goBack() haben wir auch in die Klasse Room implementiert. Insgesamt sind in Room() 20 Methoden implementiert:

1. `public Room(String description)` // Constructor  
2. `public void setMove(Room north, Room east, Room south, Room west)`  
3. `public String longDescription()`  
4. `public String getDescription()`  
5. `private String exitString()`  
6. `public Room nextRoom(String direction)`  
7. `public String performTask(String taskName)`  
8. `public void DropTask()`  
9. `public boolean isTaskCompleted()`  
10. `public String getExitString()`  
11. `public Room goBack()`  
12. `public Room goRoom(String direction)`  
13. `public void addItem(Item item)`  
14. `public void removeItem(String itemName)`  
15. `public void showItems()`  
16. `public Item getItem(String itemName)`  
17. `public Room getNorthExit()`  
18. `public Room getEastExit()`  
19. `public Room getSouthExit()`  
20. `public Room getWestExit()`

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungDie zweite Klasse ist Game. Hier wird das Game initialisiert und unter anderem werden die Räume benannt und die User-Eingabe verarbeitet. Ein Überblick über alle Variablen:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungDanach kommt der Constructor der die Beschreibung und die Ausgänge markiert, sowie ein 2D Array der Karte initialisiert.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungJetzt geht es weiter mit den Methoden. Damit das Spiel läuft braucht es die Methode play():

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungAnschliessend ein BufferedReader um die Eingabe zu ermöglichen und die Methode «getUserInput» um die Eingabe auszuführen:

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungAls nächstes wird die Eingabe verarbeitet mit einer Switch Case Schleife:

Die Klasse OutPutHandler beinhaltet lediglich Textnachrichten also wird diese nicht in der Dokumentation miteinbezogen. Die Klasse Game hat insgesamt 3 Methoden und 1 Konstruktor:

public Game() // Constructor

public void play()

private String getUserInput()

private boolean processCommand(String command)

Ab diesem Zeitpunkt jetzt funktioniert das Game. Jetzt kommt die Klasse ConsoleGameMap,   
die eine kleine Karte implementiert. Als erstes wird das 2D Map-Array und der currentRoom importiert, sowie die 2 Methoden erstellt um sie dann in displayMap() zu verwenden:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte BeschreibungAnschliessend wird die Karte wie folgt erstellt und in der Konsole ausgegeben:

Ein Bild, das Text, Schrift, Screenshot, Design enthält.

Automatisch generierte BeschreibungEin Beispiel der Karte sieht folgendermassen aus:

Aktueller Raum: [ ] // Grün

Aktueller Raum Task erledigt: [ ] // Blau

Leere / Keine Räume: [ ] // Rot

Der User X ist immer Grün: [ X ] // Task nicht erledigt

Im Bild ist also der aktuelle User in einem Raum, bei dem der Task erledigt wurde.

Ein Bild, das Text, Screenshot, Design enthält.

Automatisch generierte BeschreibungAls nächstes folgt die Klasse Item, mit der man in jedem Raum Gegenstände Hinzufügen, Entfernen und Anschauen kann:

Zu guter letzt eine Klasse StartGame() um eine Instanz zu starten und um die Methode play() aufzurufen, um das Spiel zu starten.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Das Spiel ist gegen Fehleingaben geschützt das heisst, man kann das Spiel nicht durch bestimmte Eingaben zum Absturz bringen, was sehr wichtig ist beim Programmieren.