Modul 320 – Übungen Zorg Game Projekt



1 Zork 2 – Game Projekt

Thema: Mit Klassen, Eigenschaften, Objekten, Listen, Vererbung, JavaDoc und JUnit

arbeiten.

Partnerarbeit: maximal 2 Lernende Zeit: ca. 8 Lektionen

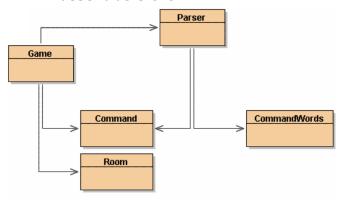
Präsentation: Das eigene Spiel wird der Klasse in ca. 5 Minuten Vorträgen präsentiert

(in unserem Fall werdet ihr ein kurzes ScreenCast der Präsentation erstellen)

1.1 Einleitung

Das Zork 2 ist eine einfache Basis für ein Abenteuer Spiel. Es implementiert Räume, welche man betreten und wieder verlassen kann.

1.2 Klassenübersicht



1.3 Hauptklasse für Zork

Das Spiel stammt ursprünglich aus einem Beispiel der BlueJ Umgebung. Eine Hauptklasse ist die Klasse, welche den Entry Point des Programmes bildet. Hier fängt also alles an. Sie besitzt eine statische Methode *main*. Dank dieser Methode weiss die eclipse Umgebung wo das Programm mit der Ausführung beginnen soll.

Erstellen Sie die Klasse Zork2, welche uns als Hauptklasse dienen soll. Bei der Erstellung dieser Klasse kann die Erstellung der Methode *main* in intellij direkt angegeben werden.

Welche Klasse muss in der main Methode instanziert werden, und welche Methode dieser Klasse muss aufgerufen werden, damit das Programm startet?





1.4 Erzeugung der Räume (aus dem Beispiel)

```
Room outside, lab, tavern, gblock, office;
...

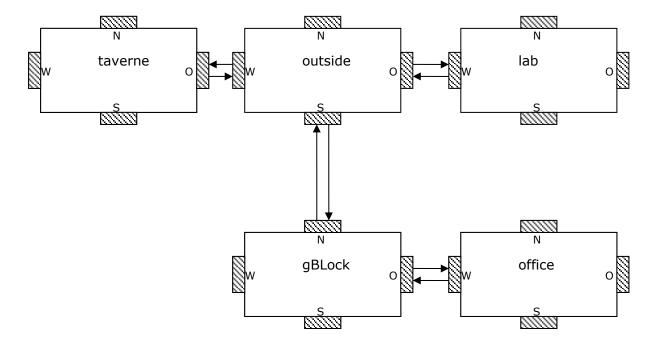
// create the rooms
outside = new Room("outside G block on Peninsula campus");
lab = new Room("a lecture theatre in A block");
tavern = new Room("the Seahorse Tavern (the campus pub)");
gblock = new Room("the G building");
office = new Room("the computing admin office");

// initialise room exits
outside.setExits(null, lab, gblock, tavern);
lab.setExits(null, null, outside);
tavern.setExits(null, outside, null, null);
gblock.setExits(outside, office, null, null);
office.setExits(null, null, null, gblock);

currentRoom = outside; // start game outside
```

1.5 Raumsituation

Dies ist eine Skizze der Räume aus dem vorgegebenen Beispiel.





2 Aufgabe 1 – Spielidee - Dokumentation:

2.1 Eigene Räumlichkeiten definieren.

Definieren Sie eingene Räumlichkeiten eines Gebäudes oder einer Landschaft, ganz nach Ihrer Phantasie. Übertreiben Sie nicht, wählen Sie für den Anfang maximal 10 Räume.

Notiz	zei	n																															
	Ι.	Ţ.,				Ι.			Ι.																		I						
		ка	ur	ne	: S	ele	ct	a,	01	H	се	, li	ab	,	en	tr	у,	οι	uts	sid	е,	ga	arc	er	١,	to	ile	t					
		ка	ur	ne	: S(ele	ct	a,	01	HIC	се	, l	ab	,	en	tr	у,	Οl	ıts	sid	e,	ga	arc	er	١,	to	ile	t					

a) Erstellen Sie eine Skizze und Beschreibung in ihrer Dokumentation und zeigen Sie diese der Lehrperson.

2.2 Gewonnen

Diesem Spiel fehlt eine massgebende Eigenschaft, welche wir jetzt zusätzlich implementieren wollen. Wir wollen implementieren, dass wir auch "gewinnen" können.

a) Beschreiben Sie in ihrer Dokumentation wann Sie gewonnen haben!

Implementieren Sie diese Funktion so, dass man beim Erreichen eines bestimmten Raumes gewinnt. Erweitern Sie die Klasse Raum mit einer Eigenschaft, über welche herausgefunden werden kann ob es sich bei einem bestimmten Raum um den "Gewinner Raum" handelt. Definieren Sie einen solchen Raum (Objekt), bei welchem man gewinnt, sobald man den Raum betritt.



Tip:

Untersuchen Sie wie die Ausgabe der Raumbezeichnung und die möglichen Ausgänge des aktuellen Raumes, programmiert ist. Konkret die Methode *longDescription()* der Klasse Room. Versuchen Sie es zu verstehen und analog zu programmieren.

2.3 Befehl "back"

Implementieren Sie den Befehl back. Dieser Befehl bringt uns zu diesem Raum, von dem wir gekommen sind.

a) Beschreiben Sie in ihrer Dokumentation wie Sie es realisieren möchten!

2.4 Gegenstand im Raum

Im Raum kann ein Gegenstand sein. Erzeugen Sie eine Klasse *Item* mit den Eigenschaften und den dazugehörenden Methoden. Erweitern Sie die Klasse *Room* so, dass man dem Objekt einen Gegenstand hinzufügen kann.

- a) Beschreiben Sie in ihrer Dokumenation wie Sie vorgehen!
- b) Wir müssen auch nach dem Gegenstand im Raum "fragen" können. Dokumentieren Sie wie es realisieren.

2.5 Mehrere Gegenstände

Im Raum können mehrere Gegenstände sein. Erweitern Sie die Klasse *Room* so, dass es eine *ArrayList* mit Gegenständen besitzt. Die Ausgabe soll auch entsprechend abgeändert werden.

Notizen	Neue Klasse 'Item' mit einem ArrayList implementiert.
	Methoden auch in der Klasse 'Item'

Informatik

Modul 320 – Übungen Zorg Game Projekt



2.6 Gewicht

Ein Gegenstand hat ein Gewicht. Erweitern Sie die Klasse *Item* mit einem Attribut *weight*.

a) Erweitern sie die Klasse und beschreiben Sie wie sie vorgehen!

2.7 Befehl "map"

Ergänzen Sie die Klasse *Game* mit einer Methode, welche auf Befehl alle Räume mit allen Gegenständen ausgibt. Erweitern Sie dazu die Klasse *Game* mit einer *Arraylist*, welche die Räume speichert.

a) Aus der Ausgabe soll auch ersichtlich sein, in welchem Raum man sich gerade befindet.

a) Aus der Ausgabe so	on auch ersichtlich sein, in weichem Raum man sich geräde beim	idet.
Notizen		
Die Map wird mit ein	nem 2D Array erstellt und instanziert.	
X ist der aktuelle Ra	aum und hat Feld ist 3x3 grosse Klammern: []	



3 Schwierigere Aufgaben

3.1 Der Spieler kann ein Gegenstand bei sich tragen

a) Dokumentieren Sie und realisieren Sie diese Anforderung!

Notizen																									
Die Sc	hlüs	sel	wer	den	in	der	Ga	me	, K	la	sse	im	pΙε	me	enti	ert	un	d١	vе	rde	en	VΟ	n		
Raum	zu R	aur	n au	uton	า. m	nita	eno	mn	ne	n.	_		-	\perp	_			-		_		-		-	_
	T 1	101 711		. , . , .	- 1	9	7	11																	

3.2 Der Spieler kann mehrere Gegenstände tragen (mit einem Maximalgewicht)

b) Dokumentieren Sie und realisieren Sie diese Anforderung!

b) Dokumentieren Sie und realisieren Sie diese Amorderung:											
Notizen											
Die Schlüssel & Gegenstä	ande können miteinander getragen werden.										
Die Gegenstände sind in e	einer separaten Klasse, die Schlüssel in der Klasse Game.										

3.3 Der Spieler kann Gegenstände auch wieder ablegen

c) Dokumentieren Sie und realisieren Sie diese Anforderung!

e) bokumentieren sie und redisieren sie diese Amorderun	9.
Notizen	
Schlüssel können jederzeit fallengelassen werden	. Implementiert mit der switch
case "drop" der Klasse Game.	
Der Schlüssel jedes Raumes ist eindeutig das heis	sst ich kann nicht Schlüssel
A in Raum B fallenlassen.	951,1911,1411,1111,111

3.4 Das Konzept der Vererbung wird eingesetzt

- d) Dokumentieren Sie und realisieren Sie diese Anforderung!
- e) Z.B. könnt ihr bei den Items noch spezielle Items erstellen , Item -> DiamantItem
- f) Oder bei den Spielern

,	•				
Notizen					
Haben	wir nicht r	mehr gesch	nafft, wegen	zu wenig Zeit,	bzw. eine Vererbung
passt ir	unserem	Fall nirge	ndwo hinein		
puoot ii					