LAB REPORT zum 21.01.2014

Von Adrian Frank, Keoma Trippner, Duc Anh Phi

Wir haben uns von den im Moodle gestellten Aufgaben gelöst und eigene Ziele für diese Woche erarbeitet. Diese Ziele waren:

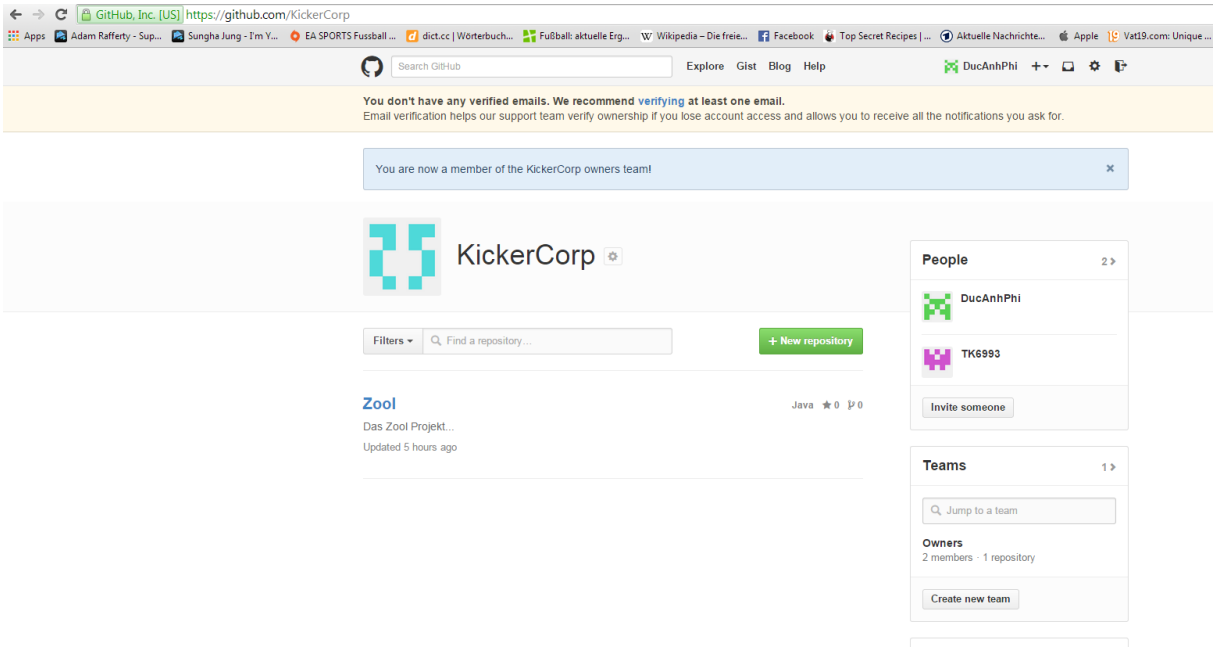
- GitHub für unsere Gruppe und das Zool Projekt anlegen.
- Eine Klasse Personen sowie eine Klasse Inventar in das Projekt implementiert.
- Unser Welt gebaut.
- Einige neue Kommando Wörter in das Projekt implementieren.
-

1. Aufgabe: Mit Github verbinden

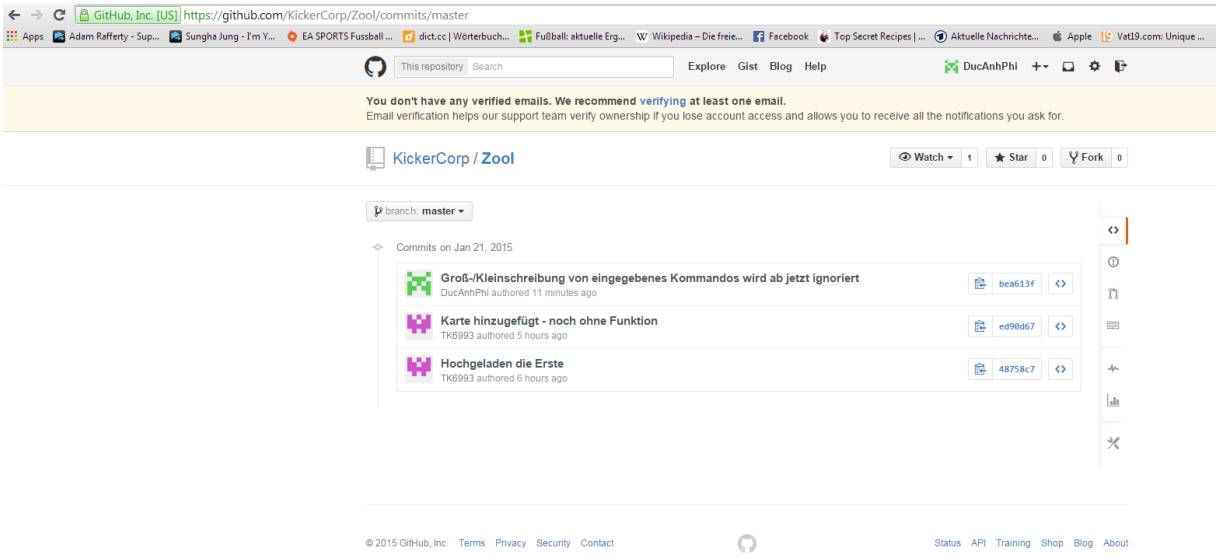
Um miteinander vernünftig an dem Projekt arbeiten zu können, brauchten wir eine geeignete Plattform um die individuellen Änderungen am Code für jeden verständlich darzustellen.

Deshalb haben wir uns alle etwas mit Github beschäftigt und ein eigenes Repository angelegt.

Wir arbeiten mit der mitgelieferten Git Shell, deren Bedienung erstaunlich einfach ist. Man musste sich nur 5 Kommandos merken: git clone, git add, git commit –m, git pull und git push.



Unsere Organisation heißt KickerCorp mit dem Repository Zool.



Hier sind unseren bisherigen commits.

Unsere commits im Detail. Schön sind die farblich hervorgehobenen Änderungen:

Rot – davor

Grün- danach

2. Aufgabe: Neue Klassen einbauen

Wir haben in das Projekt zwei neue Klassen eingebaut. Erstens die „Person“:

Person ist eine Klasse die in das Spiel NPCs implementiert. Diese bestehen aus einem Namen, mehreren Dialog Sätzen und gegebenen falls einem Item welches sie dem Spieler überlassen können.

Bis jetzt haben wir diese Klasse soweit geschrieben dass wir in der Klasse Game in jedem instanziierten Raum ebenfalls eine Person hinzugefügt werden kann.

In der Klasse Person selbst sind die Personen aufgelistet und die damit verbunden Antworten wenn man sie den anspricht. Bis jetzt haben wir nur eine Beispiel Person „Tom“ in unseren Schlossgarten eingefügt die Antworten wenn sie angesprochen wird: „ Hallo ich bin Tom.“

```
public Person(String theName)
{
   name = theName;
   item = null;
   firstAnswer = null ;
   secondAnswer = null;
   hasItem = false;


}

public String getName()
{
    return name;
}

// Getter:
public String getFirstAnswer(){
    return firstAnswer;
}

 public String getSecondAnswer(){
    return secondAnswer;
}

// Methoden:
public void setAnswersAndItem(String PersonName){
    if(PersonName.equals("Tom")){firstAnswer = "Hallo ich bin Tom!";item = "";hasItem = true;}
    if(PersonName.equals("")){firstAnswer = "";item = "";hasItem = true;}
    if(PersonName.equals("")){firstAnswer = "";item = "";hasItem = true;}
    if (PersonName.equals("")){firstAnswer = "";item = "";hasItem = true;}


    if(item == null){hasItem = false;}
```

Die zweite Klasse die wir eingebaut haben ist das Inventar. Das Inventar besteht aus einer ArrayList, einem boolean der prüft ob das Inventar schon voll ist und einem Integer der die Objekte im Inventar zählt. Zurzeit ist die Größe des Inventars auf 3 begrenzt. Des weiteren haben wir die Klasse Room soweit erweitert das sich neben den Exits und den Personen auch noch ein Item in jedem Raum befinden. Dieses wird dann einfach

dort (in „Game“) mit angegeben wo der Raum instanziiert wird. Zurzeit befindet sich im Raum „Küche“ ein Item „Brot“ welches der Spieler aufheben kann aber erst nachdem er im Schlafzimmer eine Rucksack eingesammelt hat. Es gibt auch eine Methode die dem Spieler anzeigt was für Gegenstände er grade in seinem Rucksack hat.

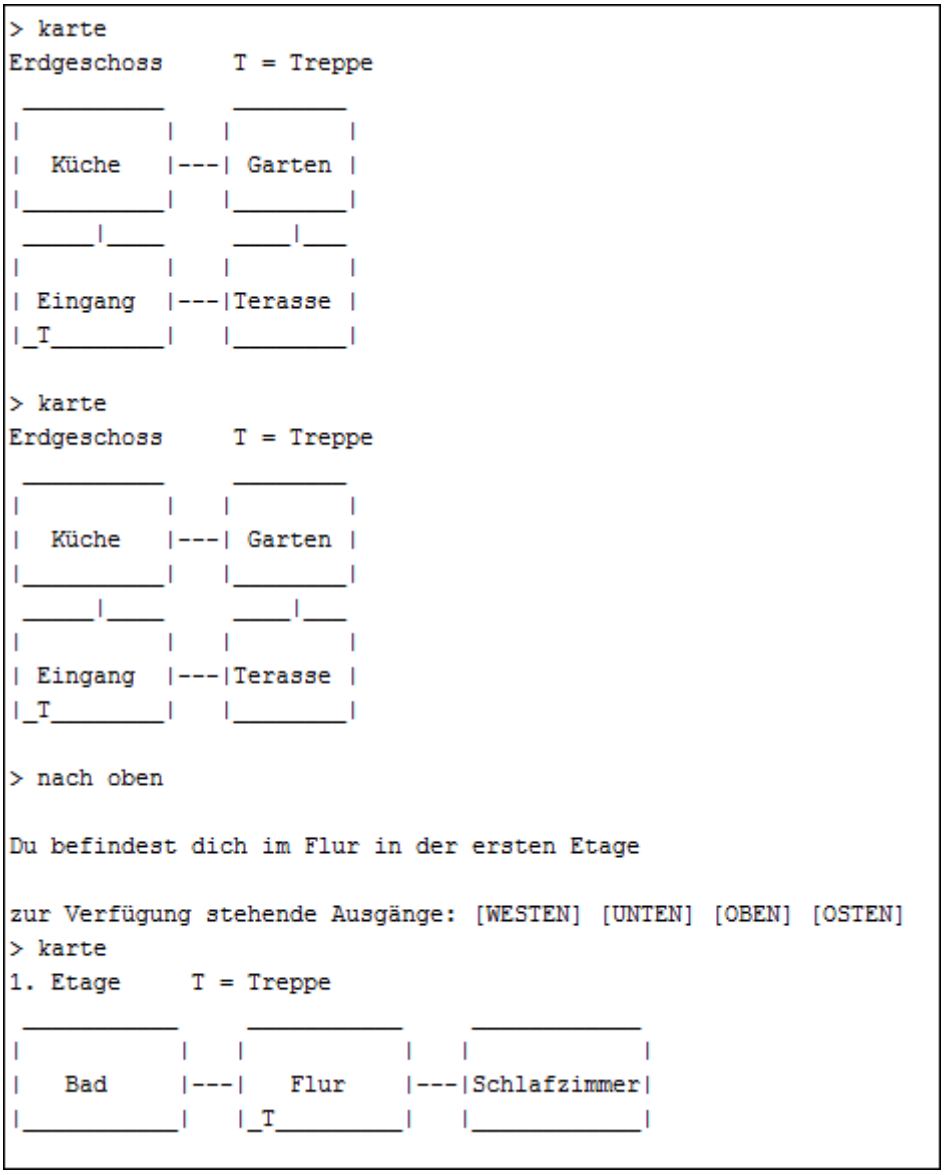
```
private ArrayList<String> Inventar;
private boolean InventarIsFull;
private int ObjectsInInventory;

/**
 * Constructor for objects of class Inventory
 */
public Inventory()
{
    Inventar = new ArrayList<>();
    InventarIsFull = false;
    ObjectsInInventory = 0;
}

public void addToInventory(String item){

    if(!InventarIsFull){
        Inventar.add(item);
        ObjectsInInventory++;
            //Rucksack wird nicht mehr im Inventar angezeigt.
            if(Inventar.contains("Rucksack")){
                Inventar.remove("Rucksack");
            }
    }
    else{System.out.println("Dein Rucksack ist schon voll!");}

    if(ObjectsInInventory == 3){InventarIsFull = true;}
    else{InventarIsFull = false;}

    }

public void useItem(String Use){
```

Wir haben auch schon mit einer dritten Klasse begonnen: Assignments

Diese Klasse kommt bei uns noch nicht groß zum Einsatz aber sie soll später den Quest Status regeln.

3. Aufgabe: Eigene Welt bauen

Als erstes haben wir eine Karte unserer Welt auf Papier gezeichnet und uns ein generelles Spielziel überlegt. Diese Karte haben wir auch in das Spiel integriert. Sie kann mit dem Befehl [Karte] aufgerufen werden und zeigt die Karte des aktuellen Stockwerks an.



Alle Räume werden in Game instanziiert mit ihren jeweiligen Exits und anschließend in einen array rooms vom Typ Room gespeichert. Hierbei ist zu beachten, dass die Indizes stockwerkweise mitlaufen. So liegen beispielsweise Küche, Garten, Eingang und Terrasse in rooms[0], rooms[1], rooms[2] und rooms[3]. Das ist wichtig, um die Karte korrekt für jede Etage wiederzugeben. Außerdem erhält jeder Raum Parameter für seine Beschreibung, ggf. vorhandene Personen, ggf. vorhandene Gegenstände und ob die Tür verschlossen ist oder nicht.

```
Room eingangshalle, schlossgarten, kueche, terrasse, schlafzimmer,flur, badezimmer, keller, vorratskammer, kickerraum,

// create the rooms
eingangshalle = new Room("Du befindest dich in der Eingangshalle",null,null,true);
schlossgarten = new Room("Du befindest dich im Schlossgarten","Tom",null,true);
kueche = new Room("Du befindest dich in der Küche",null,"Brot",false);
schlafzimmer = new Room("Du befindest dich im Schlafzimmer",null,"Rucksack",true); //Rucksack hinzugefügt
badezimmer = new Room("Du befindest dich im Badezimmer",null,null,true);
keller = new Room("Du befindest dich im Keller",null,null,true);
vorratskammer = new Room("Du befindest dich in der Vorratskammer",null,null,true);
kickerraum = new Room("Du befindest dich im Kickerraum",null,null,false);
flur = new Room("Du befindest dich im Flur in der ersten Etage",null,null,true);
zielraum = new Room("Du befindest dich im Zielraum",null,null,false);
werkstatt = new Room("Du befindest dich in der Werkstatt",null,null,true);
terrasse = new Room("Du befindest dich auf der Terrasse",null,null,true);

rooms[0] = eingangshalle;
rooms[1] = terrasse;
rooms[2] = schlossgarten;
rooms[3] = kueche;
rooms[4] = flur;
rooms[5] = badezimmer;
rooms[6] = schlafzimmer;
rooms[7] = keller;
rooms[8] = werkstatt;
rooms[9] = vorratskammer;
```

4. Aufgabe Neue Komandos:

Zuletzt haben wir einige neue Kommandos in das Spiel eingebaut:

Sprich: um eine Peron in einem Raum anzusprechen.

Inventar: Um sich das Inventar anzeigen zu lassen.

Nimm <Item>: Um einen Gegenstand aufzuheben.

Karte: Um die Karte des momentanen Geschosses zu sehen.

Nach <Richtung>: Um in einen anderen raum zu wechseln. (Wir haben „go“ in „nach“ umgeändert)