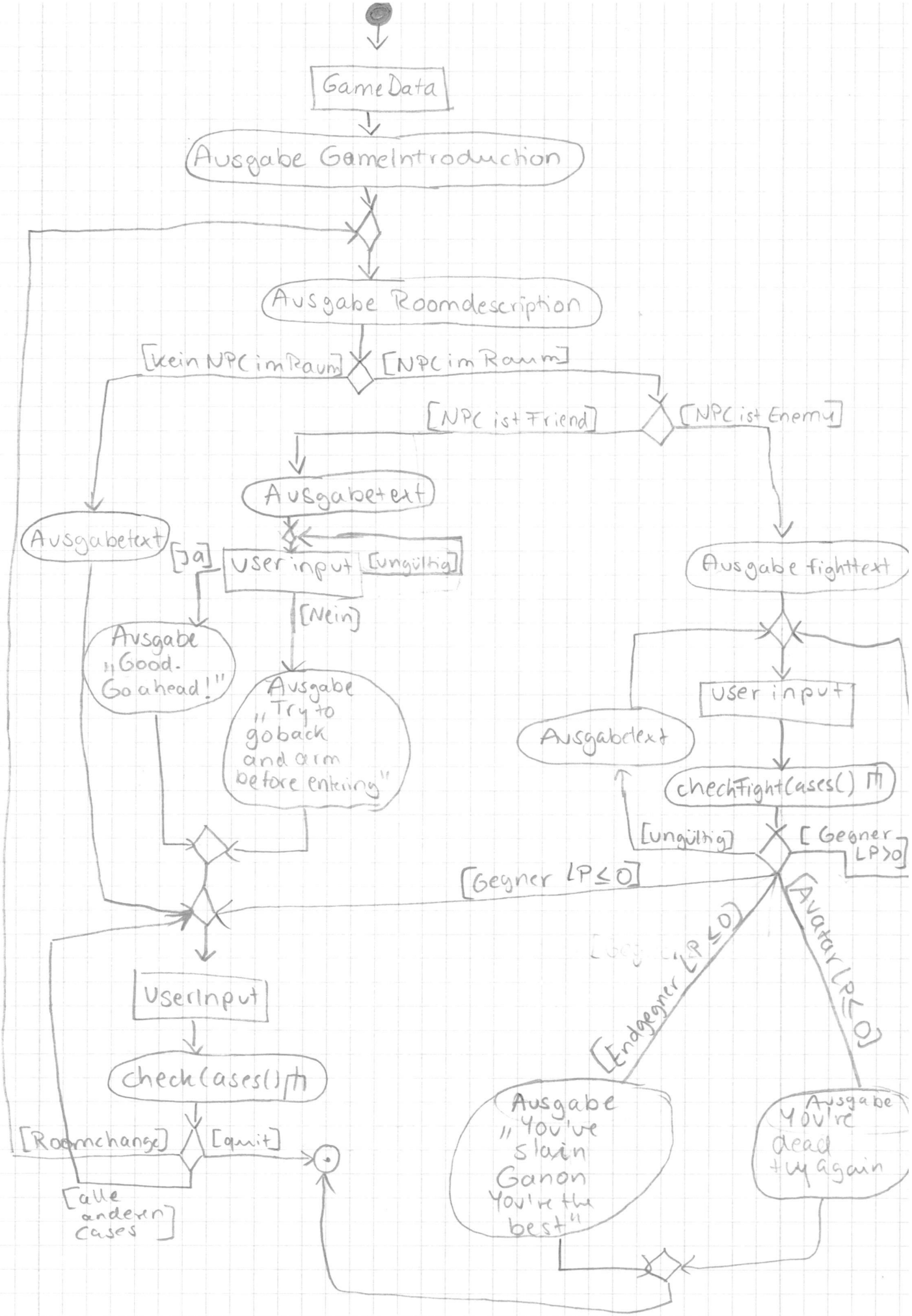
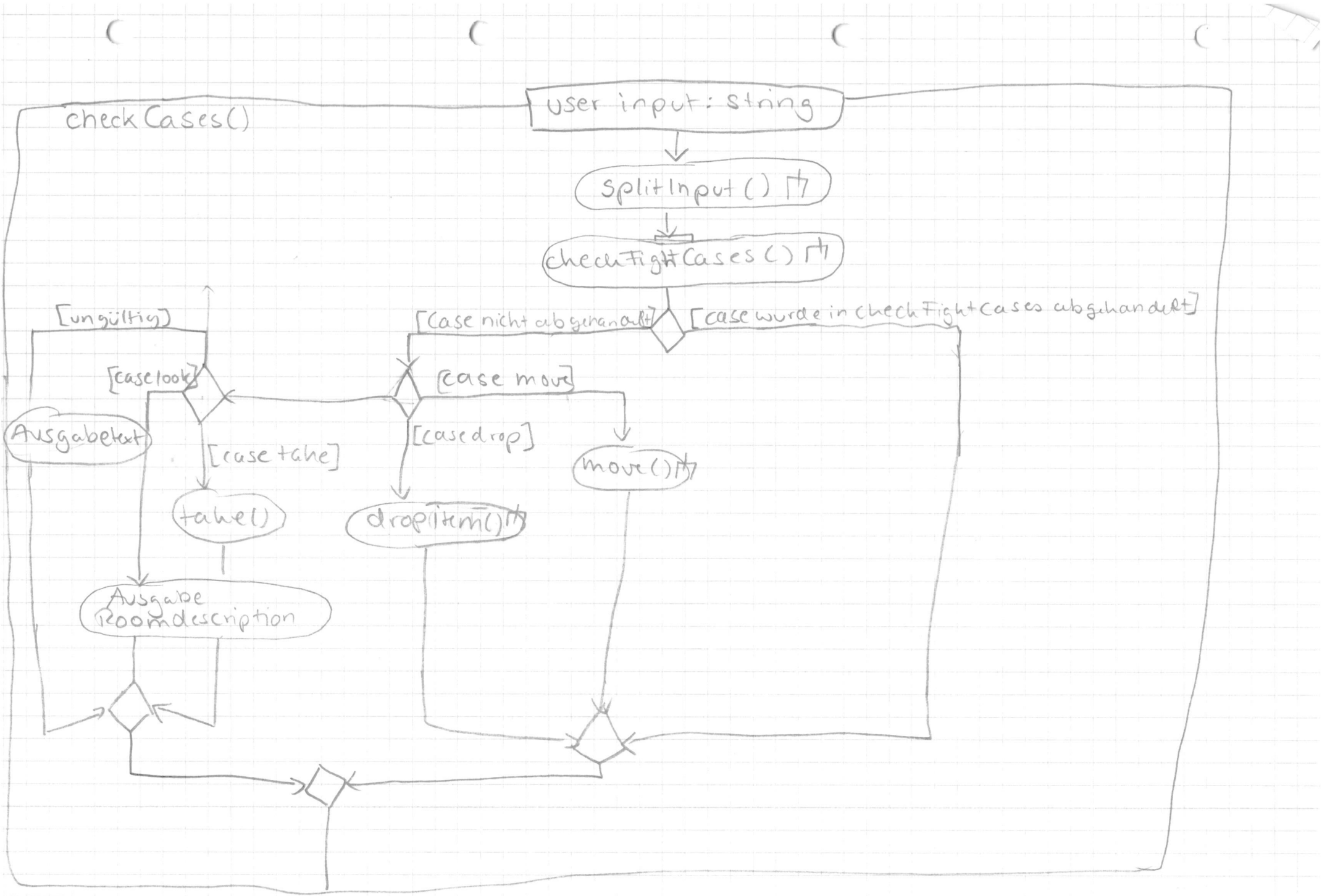
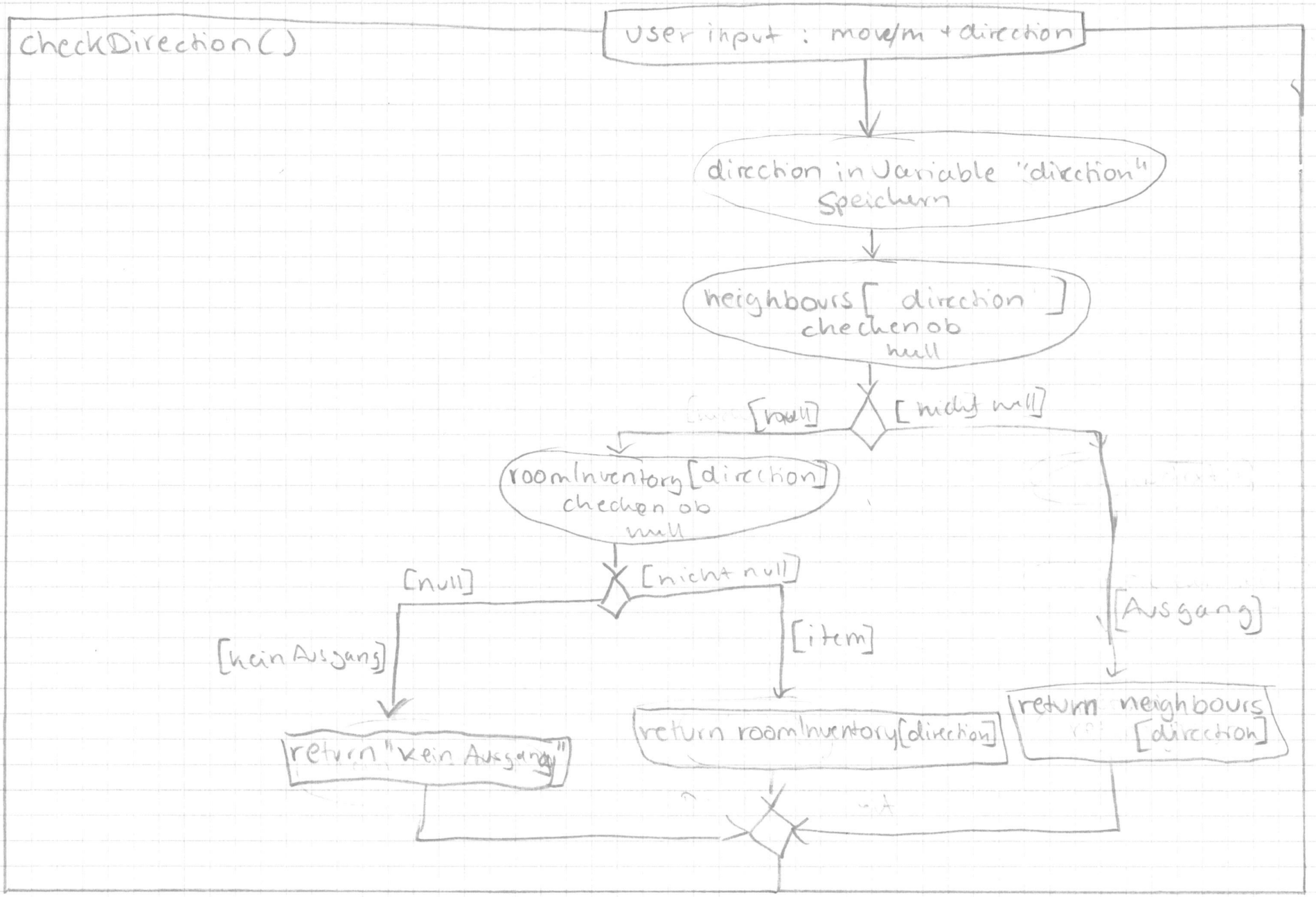
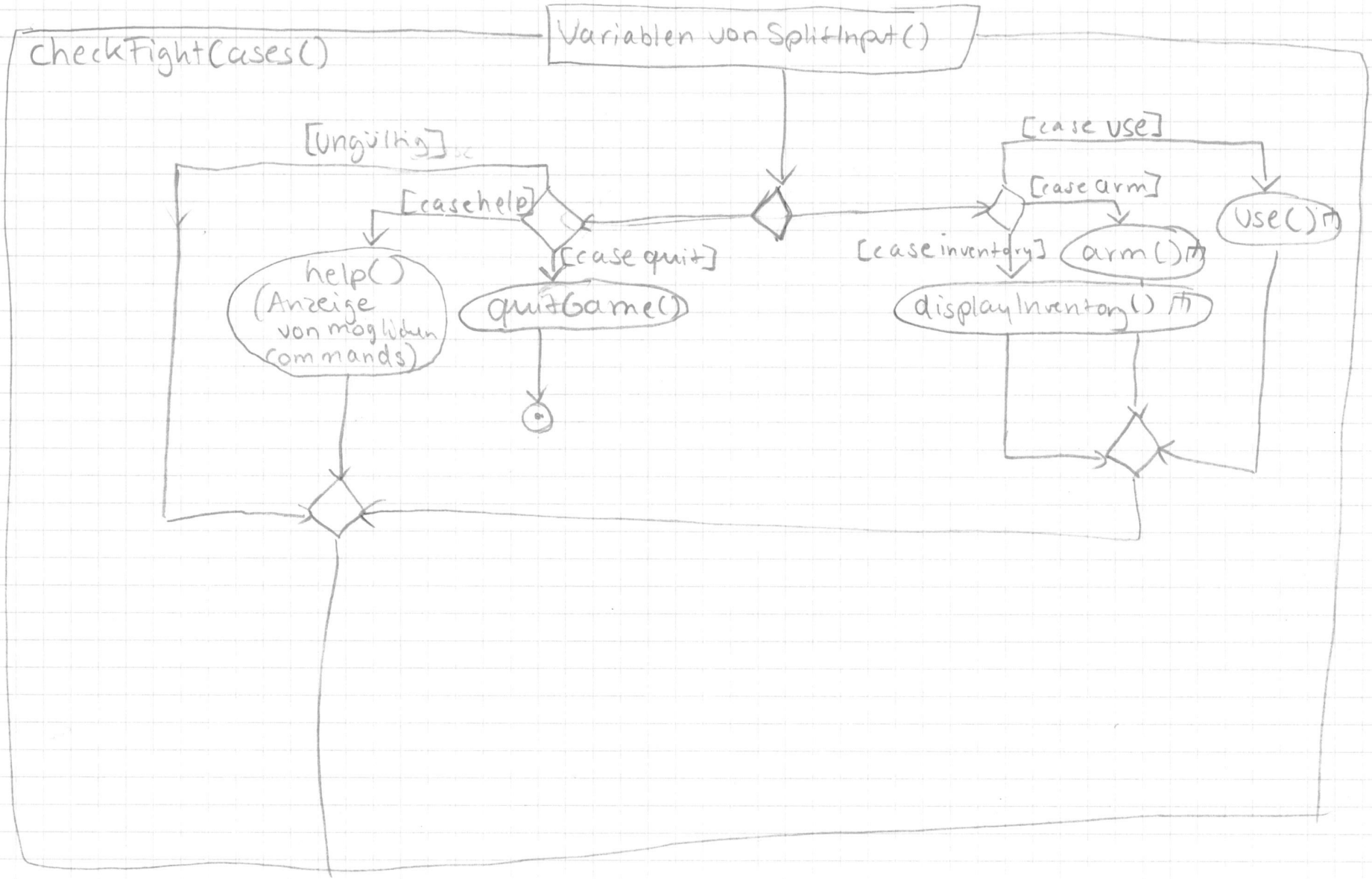

GameData
Ausgabe GameIntroduction
Ausgabe Roomdescription
[kein NPC im Raum]
[NPC im Raum]
[NPC ist Friend]
[NPC ist Enemy]
Ausgabetext
Ausgabetext
[ja]
user input
[ungültig]
[Nein]
Ausgabe „Good. Go ahead!"
Ausgabe Try to „go back and arm before entering"
Ausgabe fighttext
user input
Ausgabetext
checkFight(cases()
[ungültig]
[Gegner LP>0]
[Gegner LP≤0]
[Endgegner LP≤0]
[Avatar LP≤0]
UserInput
check(cases()
[Roomchange]
[quit]
[alle anderen Cases]
Ausgabe „You've slain Ganon You're the best!"
Ausgabe You're dead try again

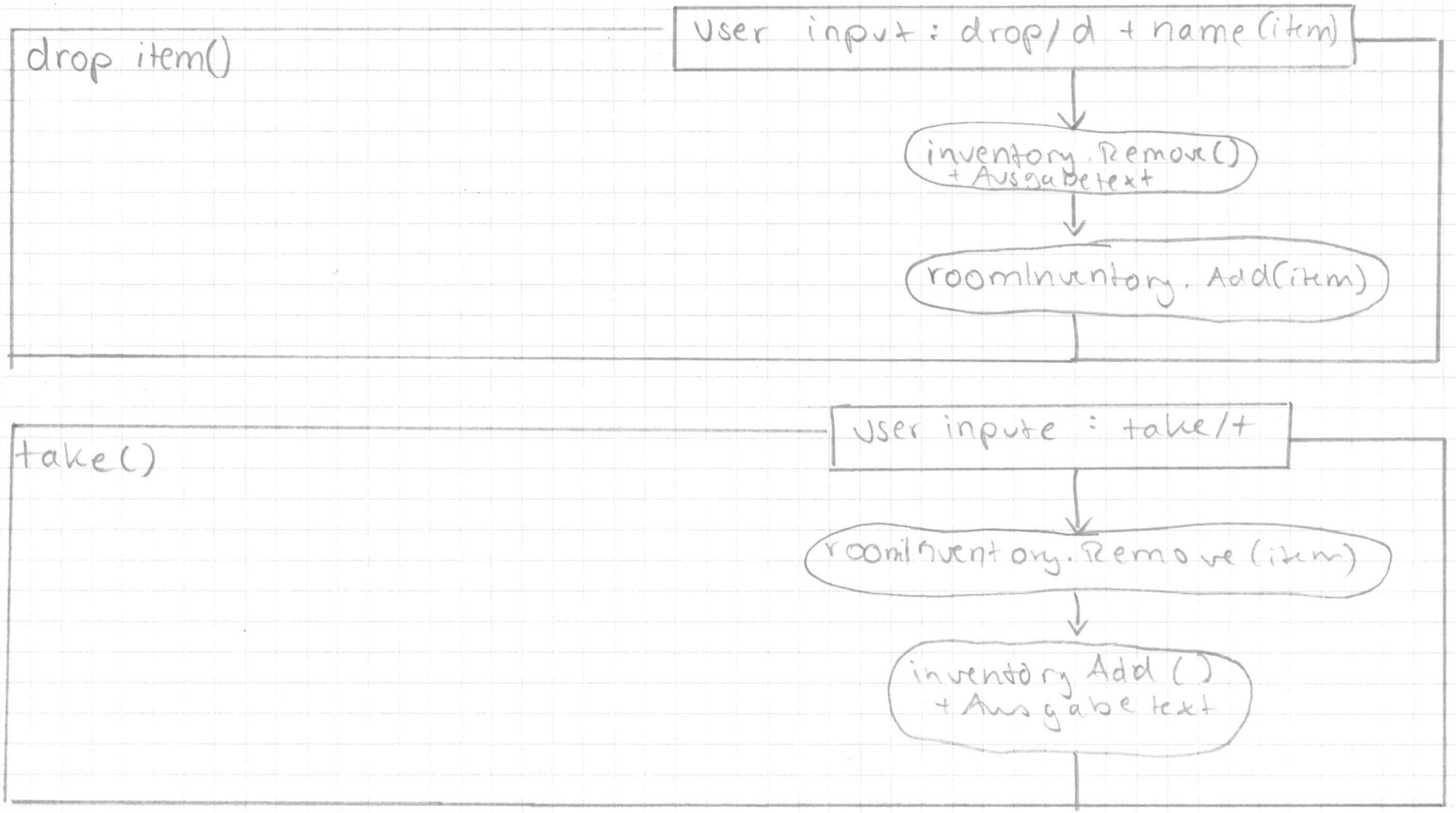

checkCases()
user input: string
splitInput()
checkFightCases()
[case nicht abgehandelt]
[case wurde in checkFightCases abgehandelt]
[ungültig]
[case look]
[case move]
Ausgabetext
[case take]
[case drop]
move()
take()
dropItem()
Ausgabe Roomdescription

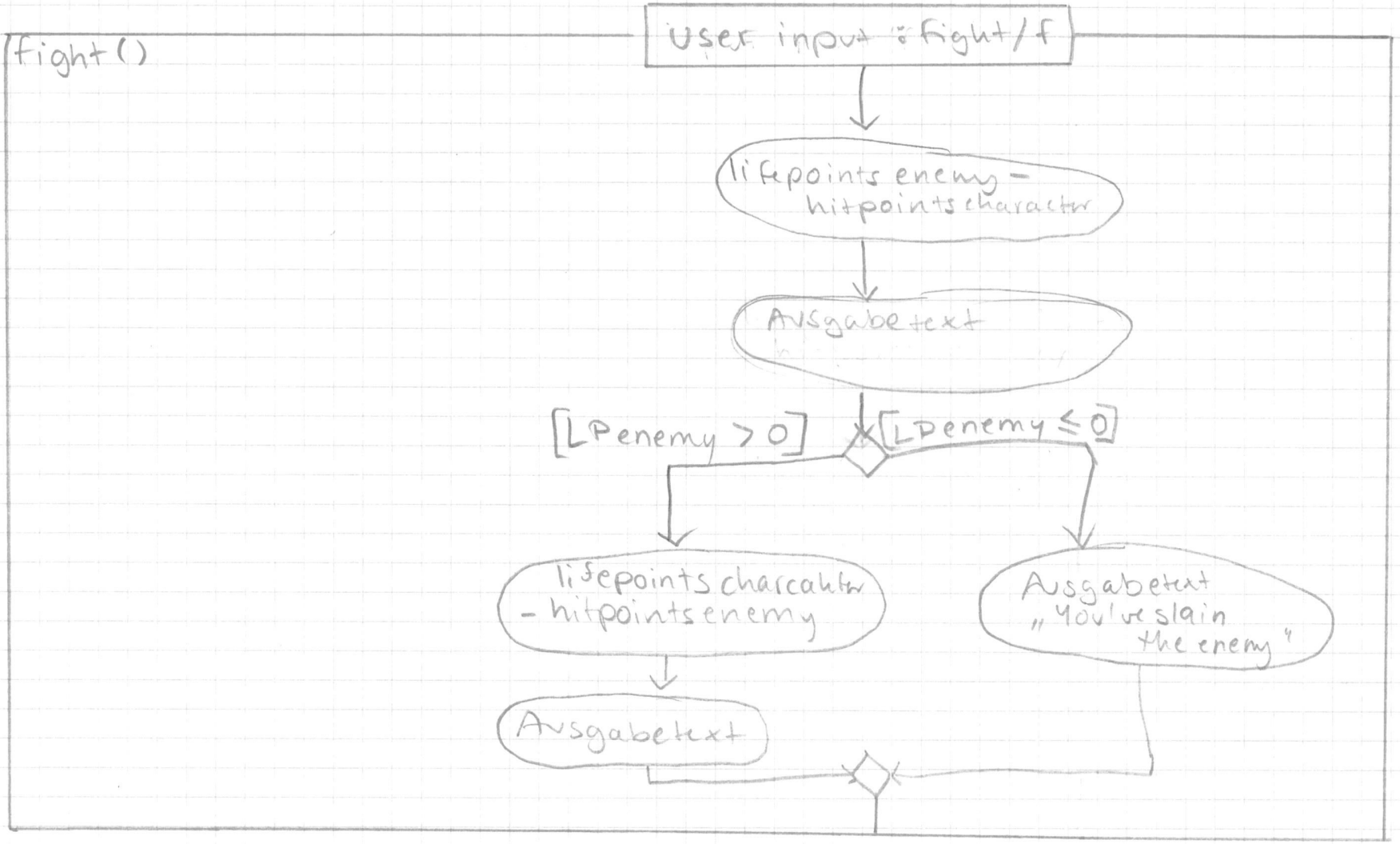
siehe move()

checkDirection()

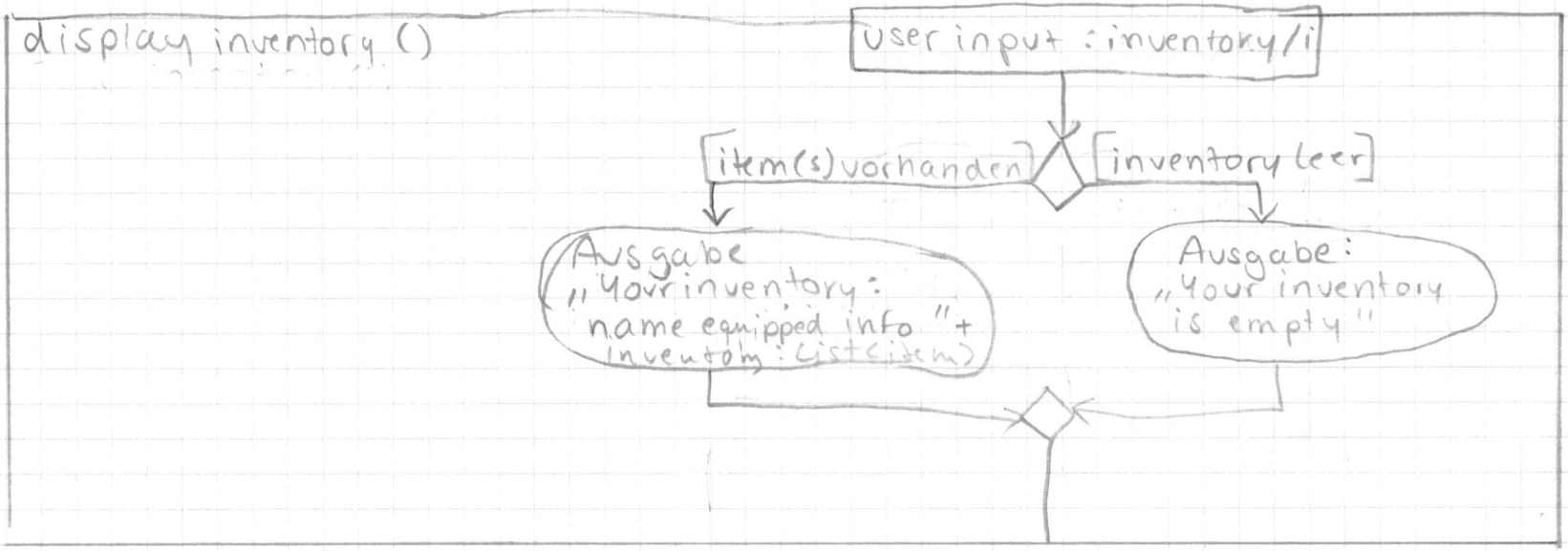



checkFightCases()
Variablen von SplitInput()
[ungültig]
[case help]
[case quit]
help()
(Anzeige von möglichen commands)
quitGame()
[case use]
[case arm]
[case inventory]
use()
arm()
displayInventory()

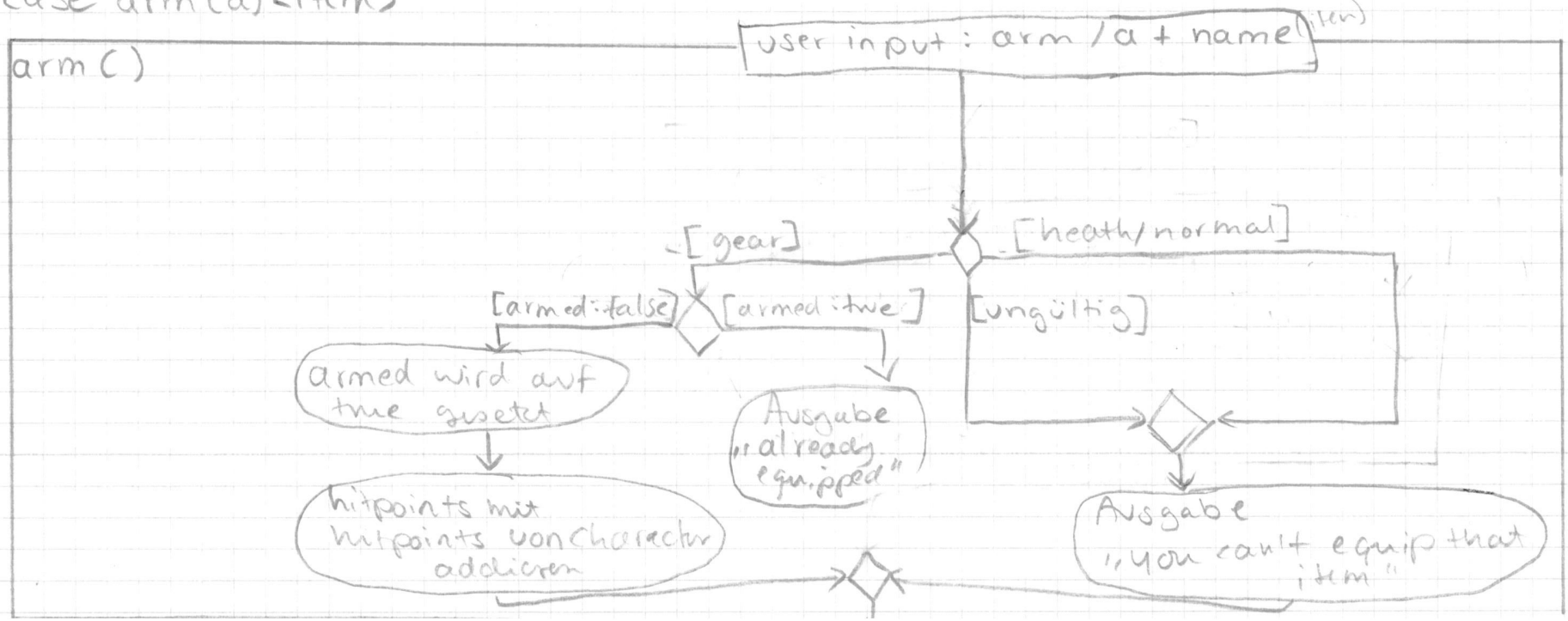

drop item()
User input: drop/d + name(item)
inventory.Remove()
+ Ausgabetext
roomInventory.Add(item)
take()
User inpute: take/t
roomInventory.Remove(item)
inventory.Add()
+ Ausgabetext

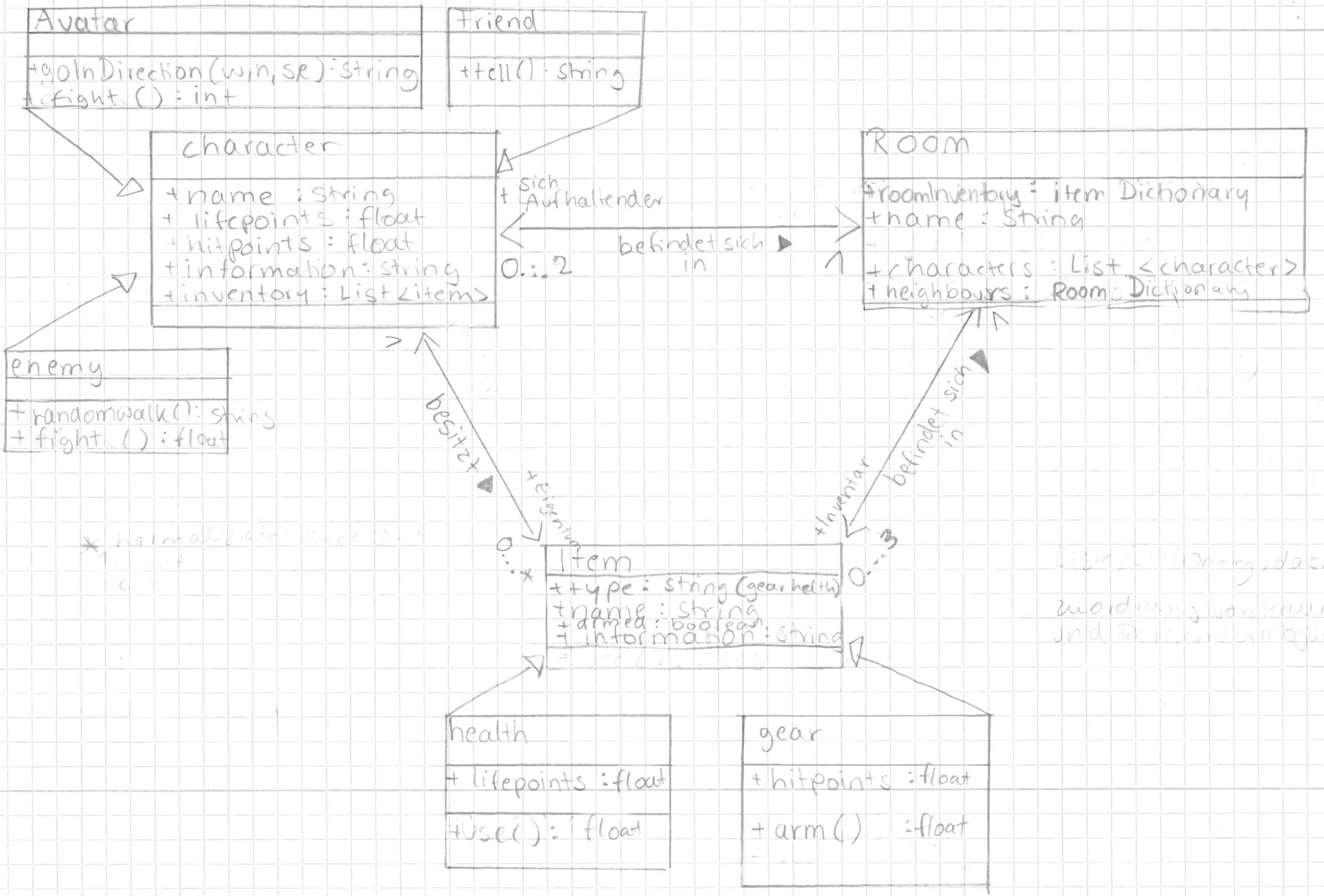

fight()
User input = fight/f
lifepoints enemy -
hitpoints character
Ausgabetext
[LPenemy > 0]
[LPenemy ≤ 0]
lifepoints charcahter
- hitpoints enemy
Ausgabetext
Ausgabetext
„You've slain
the enemy"

Case inventory(i)



case arm(a)<item>



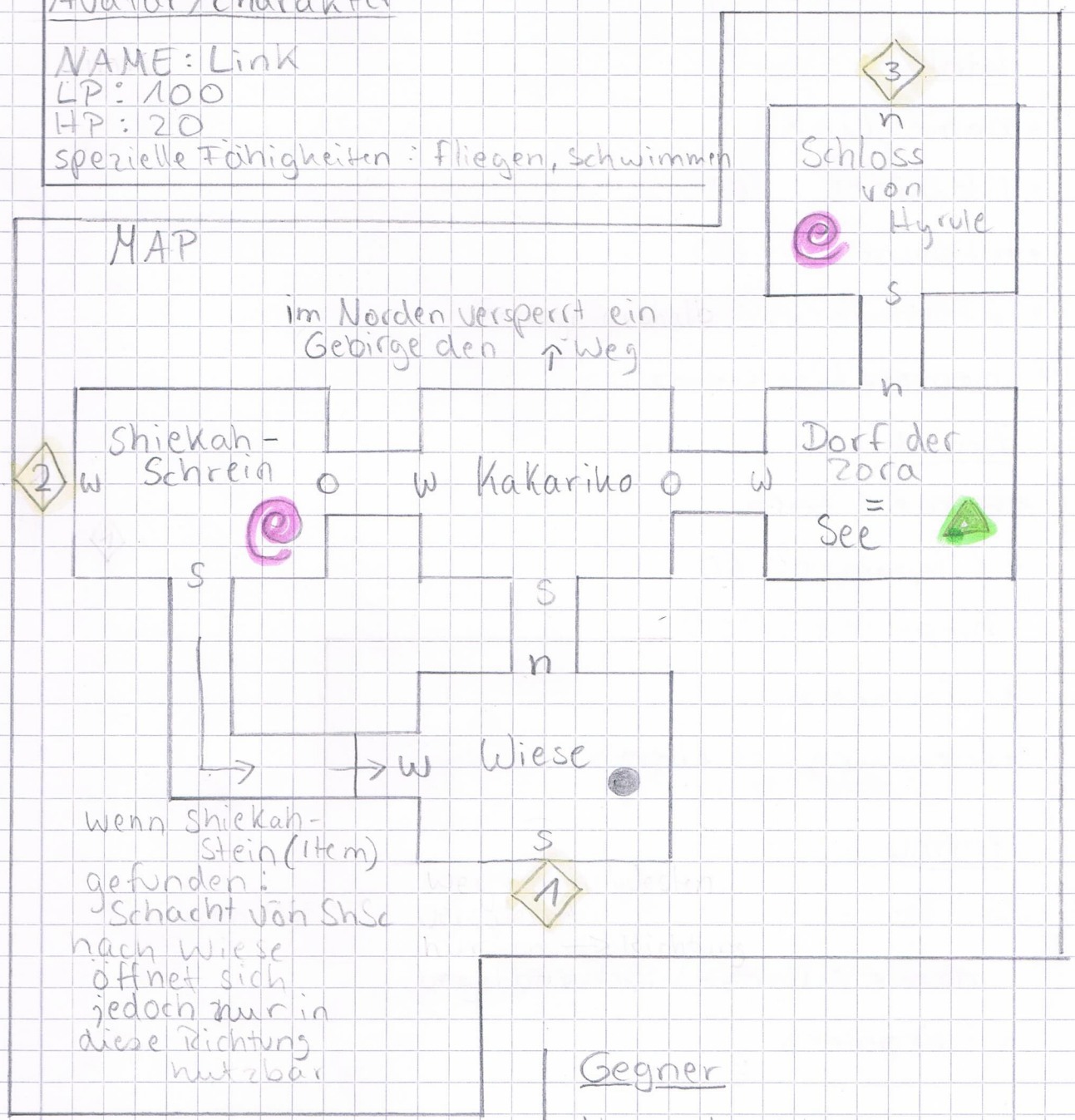

Avatar
+goInDirection(w,n,se):String
+fight():int
Friend
+tell():String
character
+name :String
+lifepoints :float
+hitpoints :float
+information:String
+inventory :List<item>
+sich Aufhaltender
befindet sich in
0..*
0..2
Room
+roomInventory : item Dictionary
+name : String
+characters : List<character>
+neighbours : Room Dictionary
enemy
+randomwalk():String
+fight():float
besitzt
+Eigentümer
0..*
+Inventar
befindet sich in
0..3
item
++type : String (gear health)
+name : String
+armed : boolean
+information : String
health
+ lifepoints :float
+use():float
gear
+hitpoints :float
+arm() :float

The Legend of Zelda: Text-Adventure

Avatar/Charakter

NAME: Link
LP: 100
HP: 20
spezielle Fähigkeiten: fliegen, schwimmen

MAP



Gegner

Nano-Wächter

Die Verheerung Ganon

Legende

n Item

Spieler muss spezielle Fähigkeit nutzen um zu passieren

Gegner

NPC

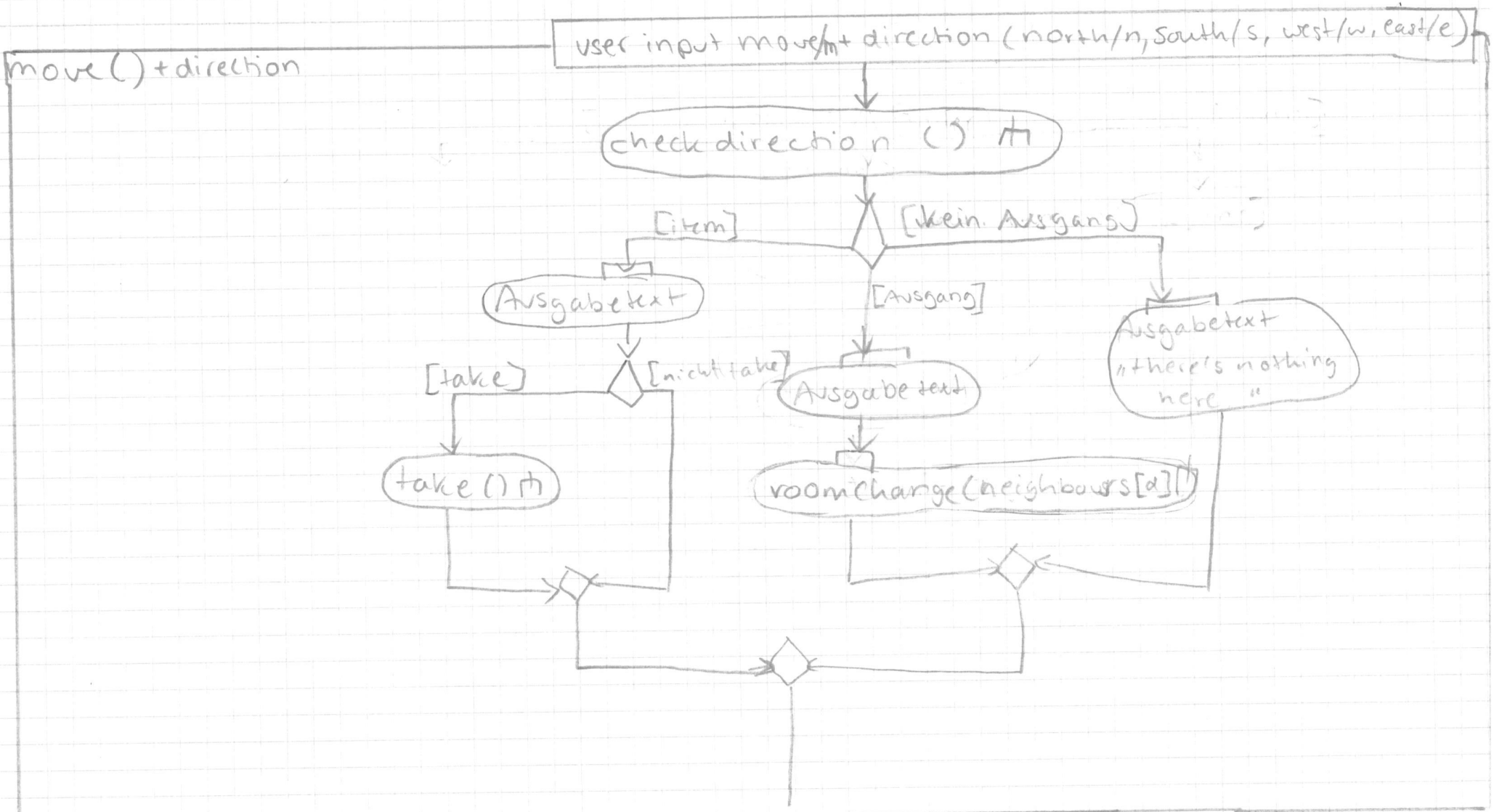
Zora

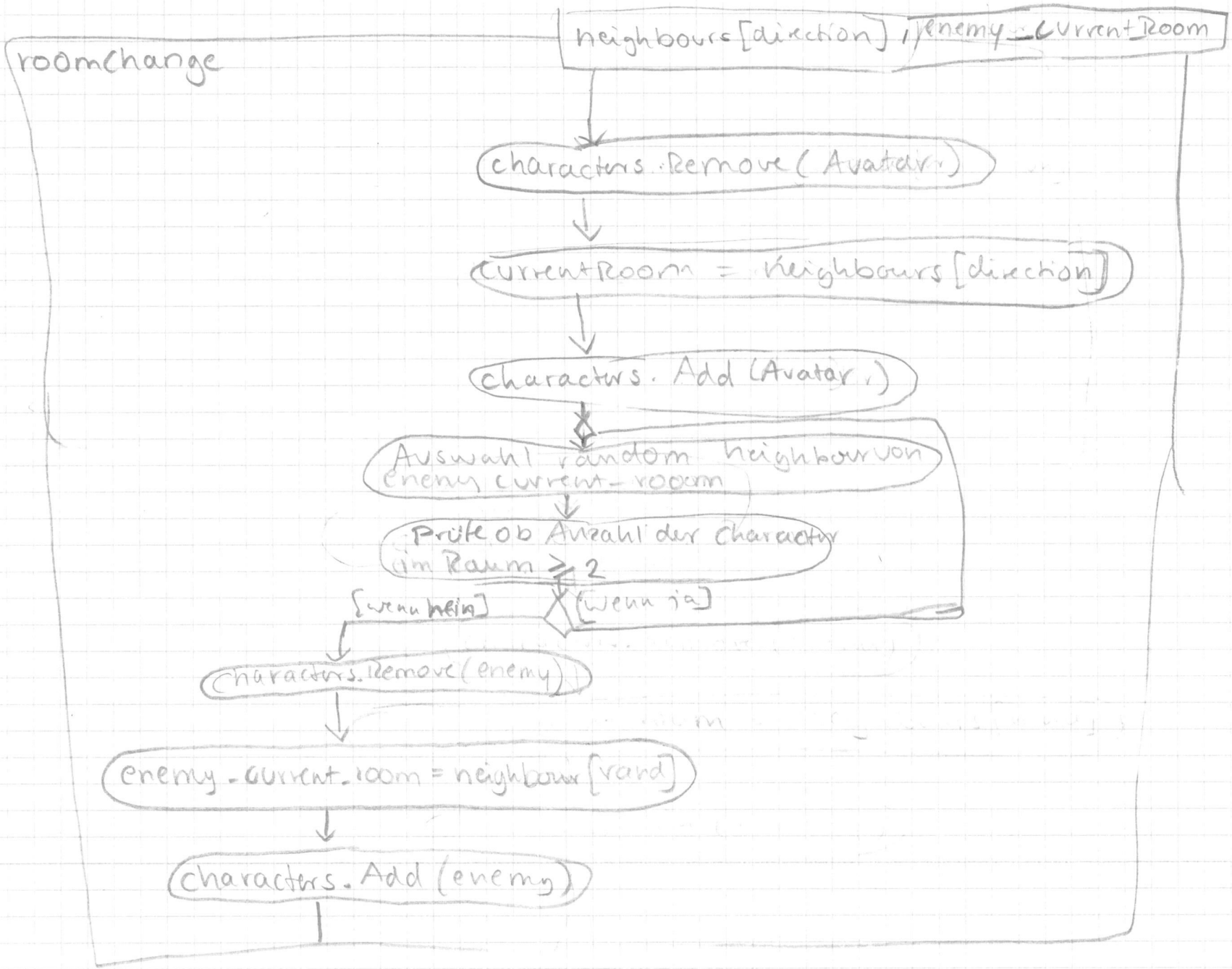
Items

Wiese: 1 → Meisterschwert

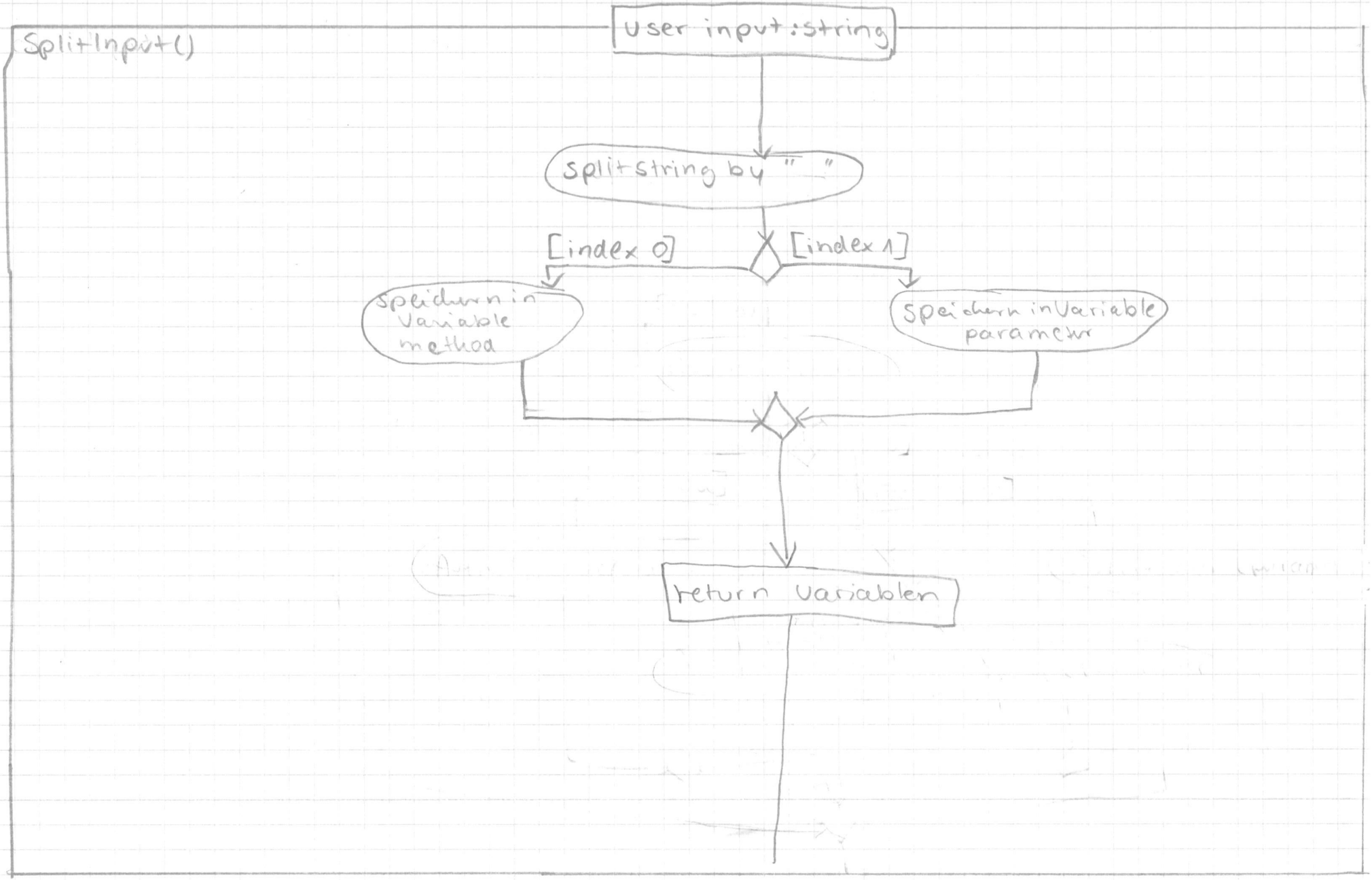
Shiekah-Schrein: 2 → Shiekah-Stein

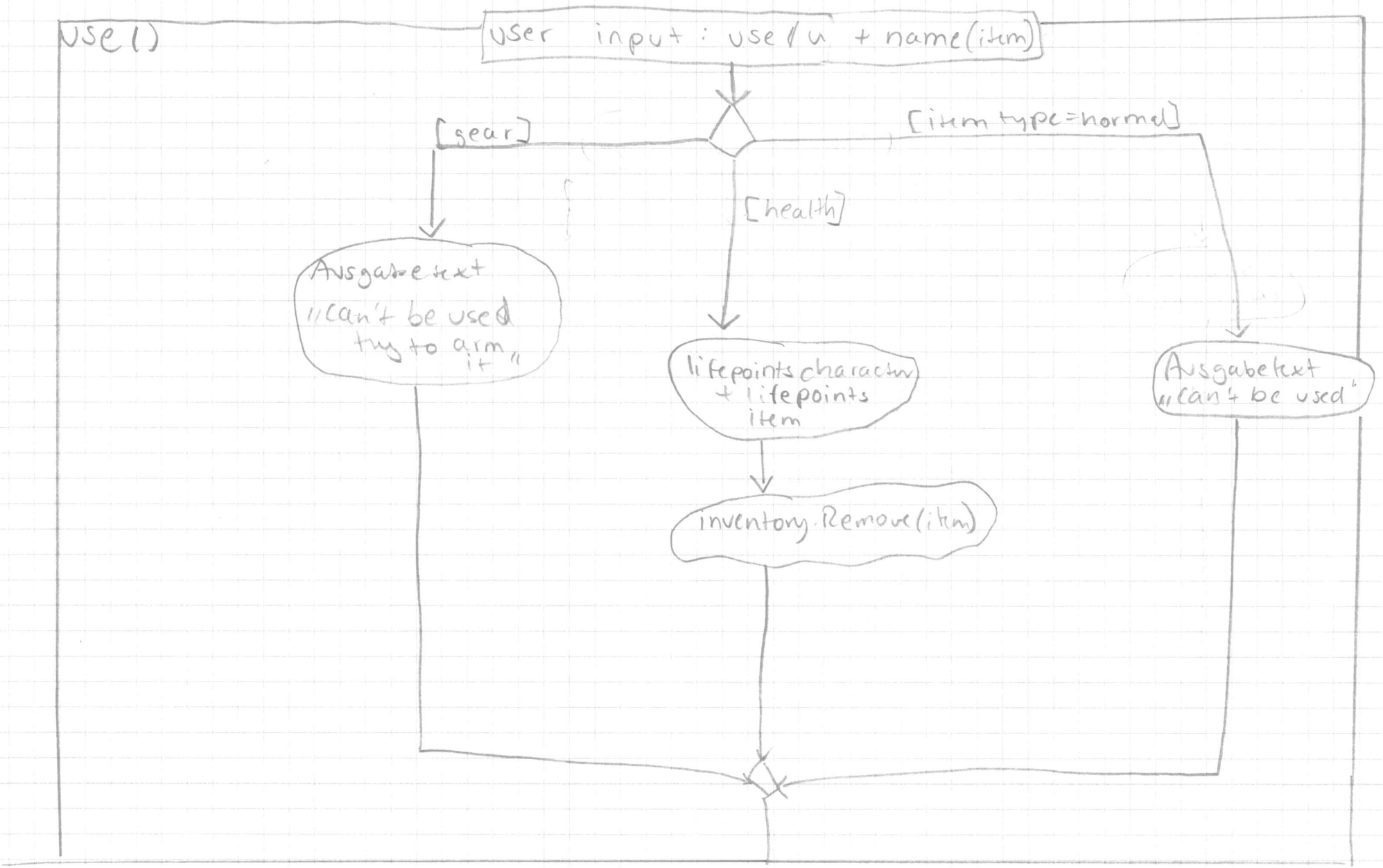
Schloss von Hyrule: 3 → Statue von Prinzessin Zelda


move() + direction
user input movement + direction (north/n, south/s, west/w, east/e)
checkdirection () #
[item]
[kein Ausgang]
[Ausgang]
Ausgabetext
[take]
[nicht take]
take () #
Ausgabetext
Ausgabetext
„there's nothing here"
roomchange(neighbours[a])


roomchange
neighbours[direction], enemy = CurrentRoom
characters.Remove(Avatar)
CurrentRoom = neighbours[direction]
characters.Add(Avatar)
Auswahl random neighbour von enemy current_room
Prüfe ob Anzahl der Charakter im Raum ≥ 2
[wenn nein]
[wenn ja]
characters.Remove(enemy)
enemy.current_room = neighbour[rand]
characters.Add(enemy)

wird genutzt bei arm() und usc(), drop, move




use()
User input : use(u + name(item)
[gear]
[item type=normal]
[health]
Ausgabetext
"can't be used try to arm it"
lifepoints charactw + lifepoints item
Ausgabetext
"can't be used"
inventory.Remove(item)