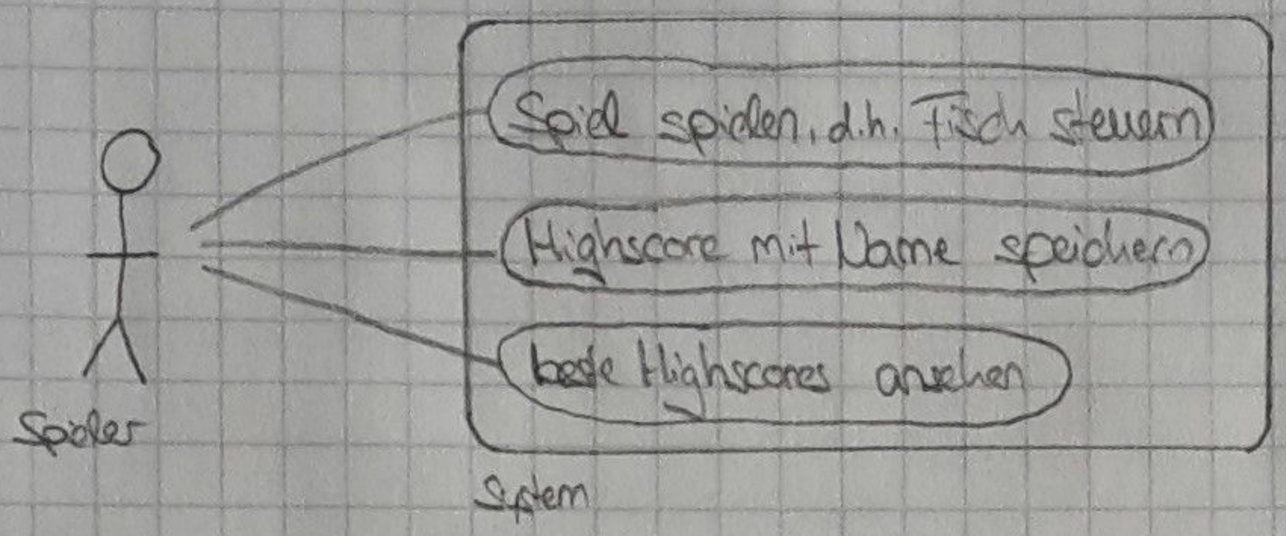
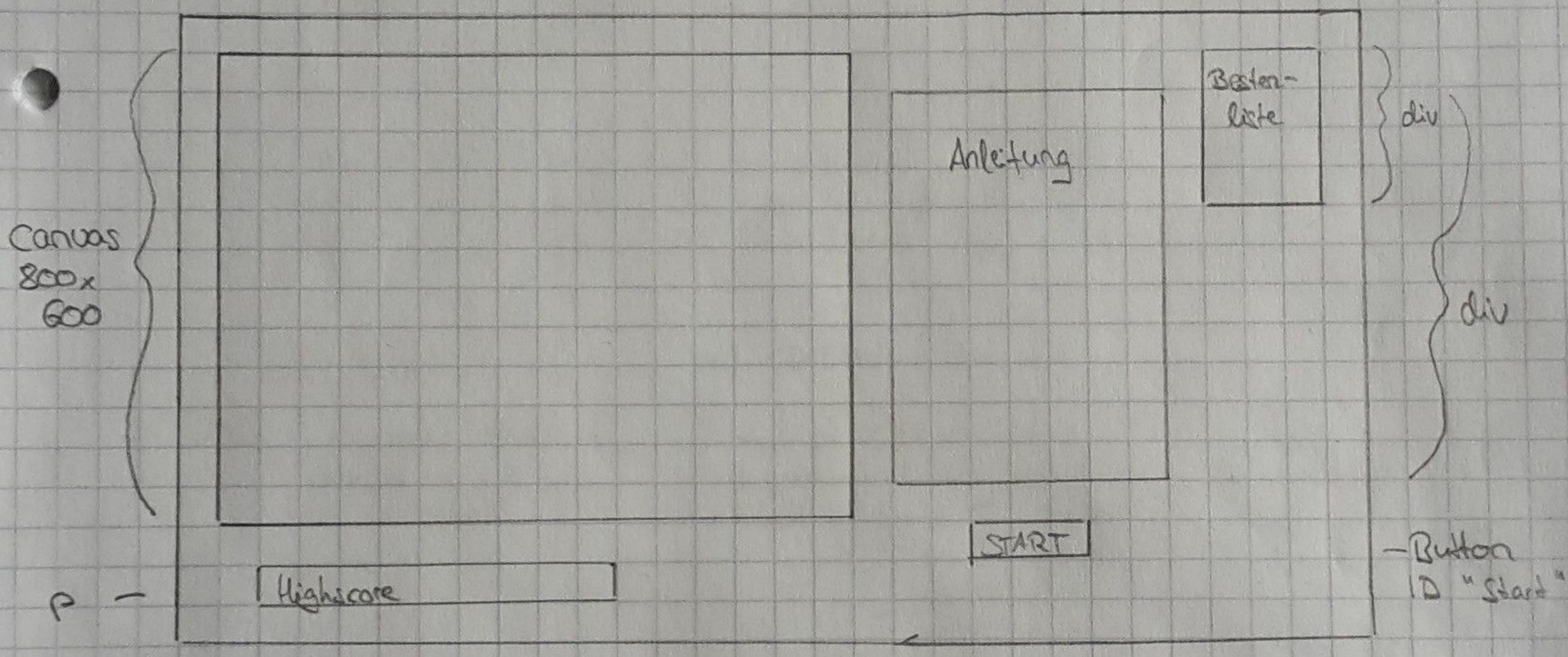
Endabgabe - Aquarium

{ Funktionale Analyse

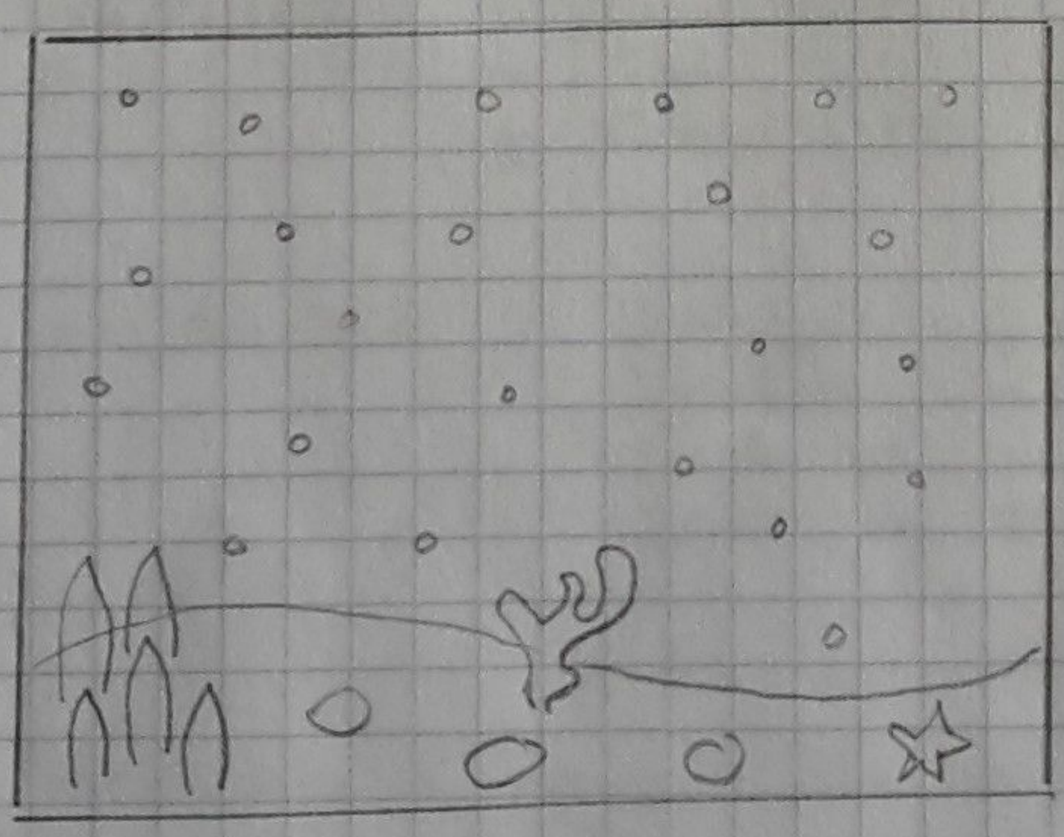
Anwendungsfalldiagramm



Skizze



Skizze - Canvas - Hintergrund



- Boden
- Wasser
- Pflanzen
- Steine
- Koralle
- Seestern

- Blubberblasen → random generiert
  → 50x

→ keine Interaktion mit Hintergrund möglich

(2)

Skizze - interaktive Objekte

Fische

- 3 Größen
  - klein → rot → immer fressbar
  - mittel → orange → ab 200 Punkten fressbar, dann Game Over
  - groß → gelb → ab 900 Punkten fressbar, dann Game Over
- schwimmen nach rechts, an random Stelle von canvas.length

besondere Fische

„Partyfisch"
- schwimmt rückwärts nach links, random Stelle von canvas-length
- so groß wie mittlerer Fisch
- andere Farbe

=> wenn gefressen, wird die Steuerung umgekehrt

„Qualle"
- lila → verlangsamt den Spielfisch deutlich
- schwimmt nach oben, random Stelle canvas.width

„Anglerfisch"
- ca. so groß wie große Fisch => immer Game Over
- schwimmt nach rechts

Futter

rot → verkleinert den Spielerfisch
grau → hebt Wirkung der Qualle auf
grün → hebt Wirkung des Partyfisches auf

Spielerfisch

mit Pfeiltasten steuerbar

- unterschl. Blautöne
- anfangs so groß wie kl. Fisch

→ frisst andere Fische
→ wechselt Farbe, wenn neue Stufe erreicht
- wächst → kl. F +0,2
  mi. F +0,5
  gr. F +1
  rot -1

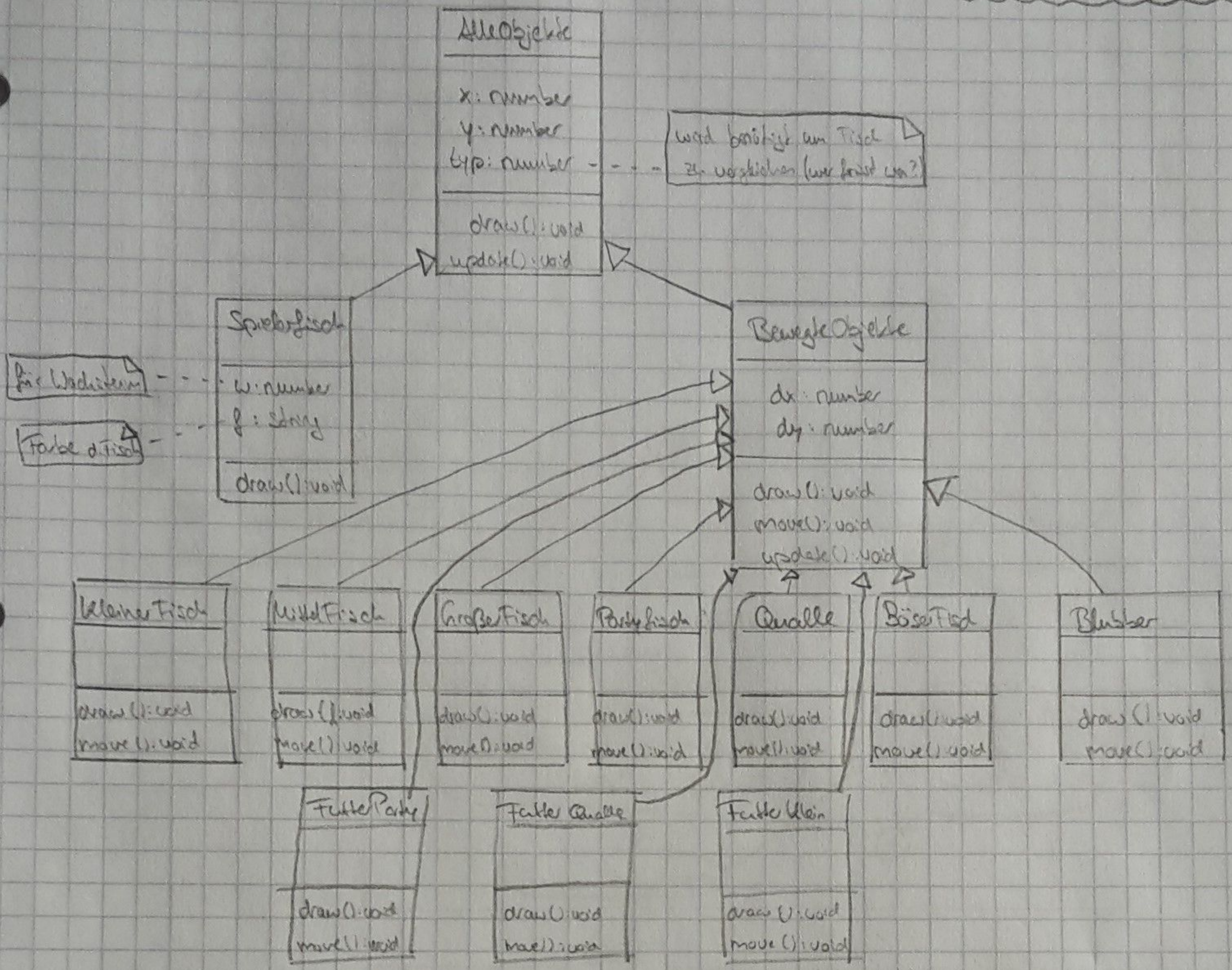
=> Fressen/gefressen werden nur am Mund möglich

=> PUNKTE
- kl. Fisch +10
- mi. Fisch +50
- gr. Fisch +100

=> GAME OVER
- Spielerfisch gefressen
- Anglerfisch berührt
- Spielerfisch zu klein
- Spielerfisch zu groß

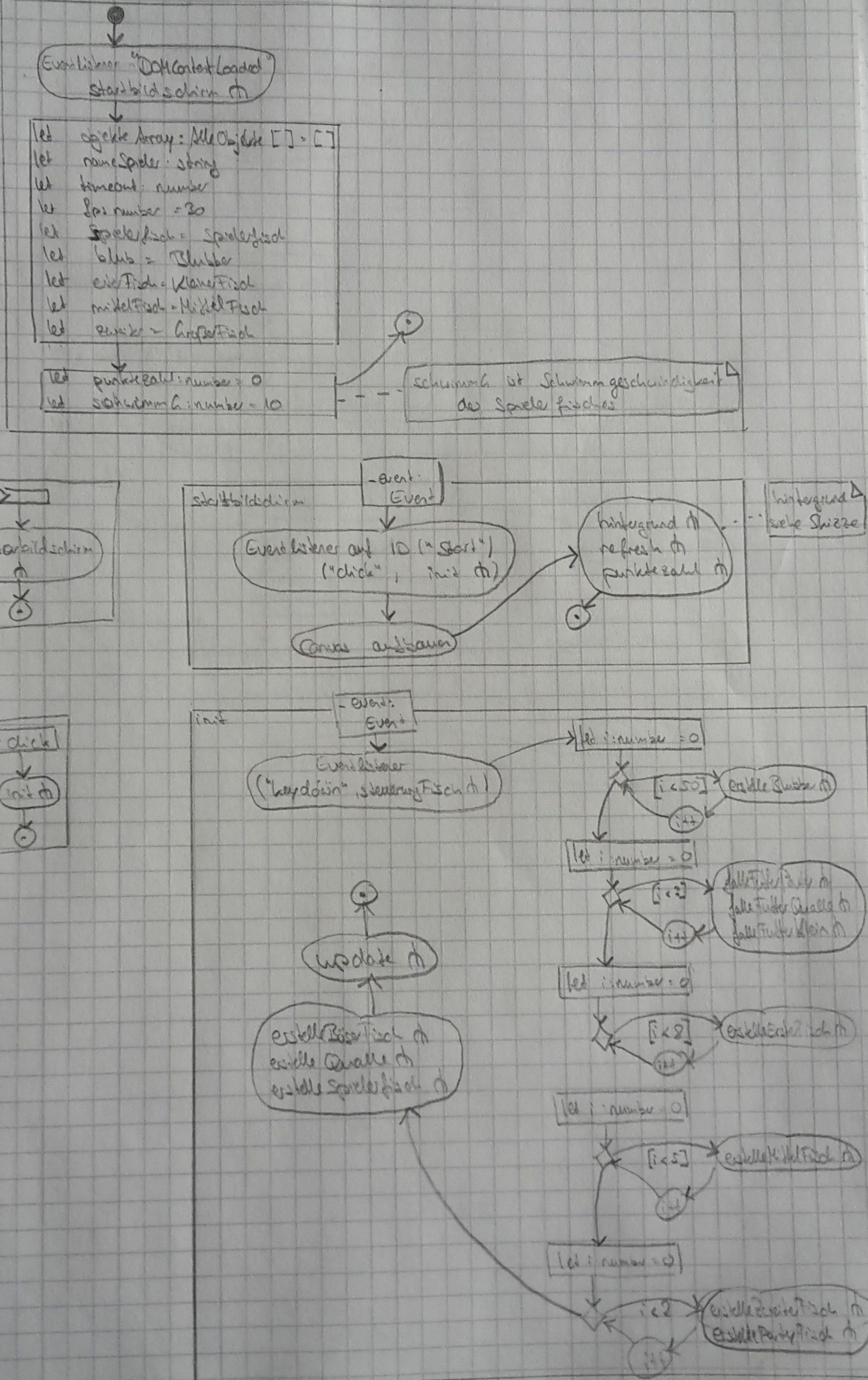
=> Eintragen in Highscore-Liste via „prompt" nach Game Over

Klassendiagramme



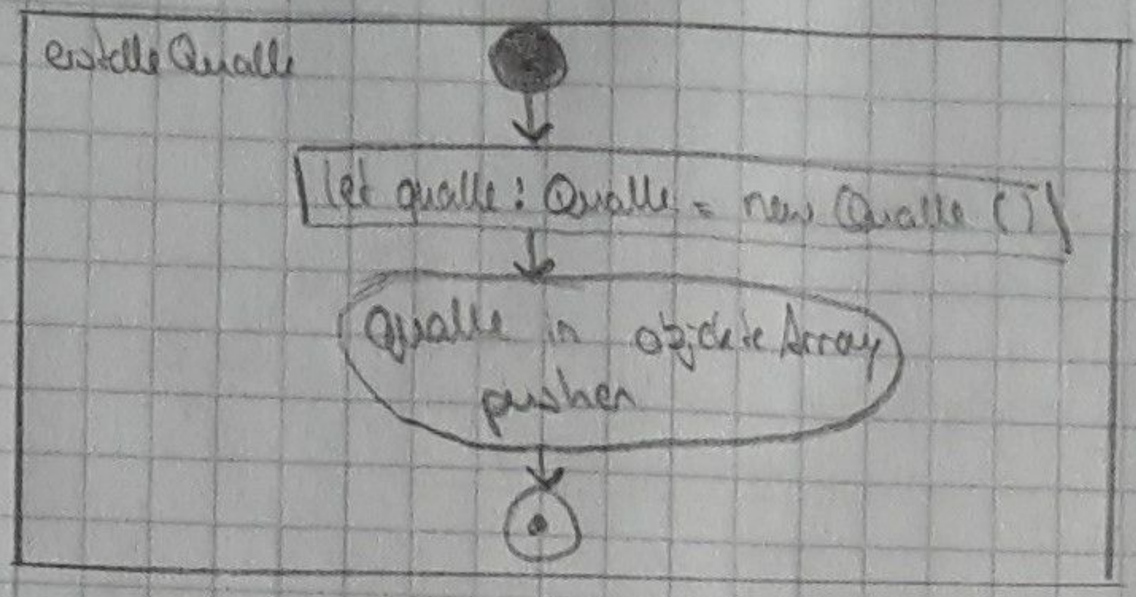
④

Aktivitätsdiagramm

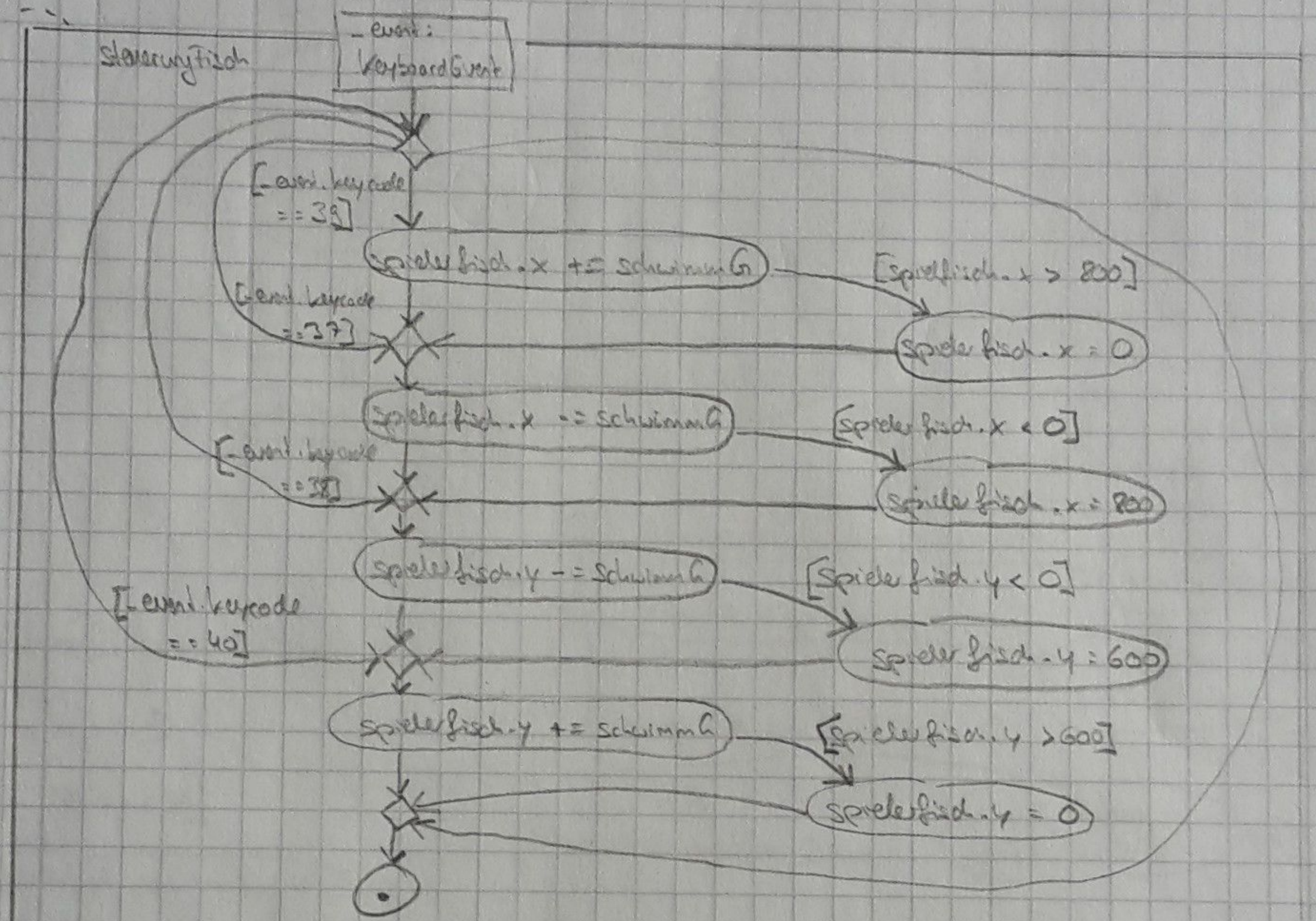


⑤

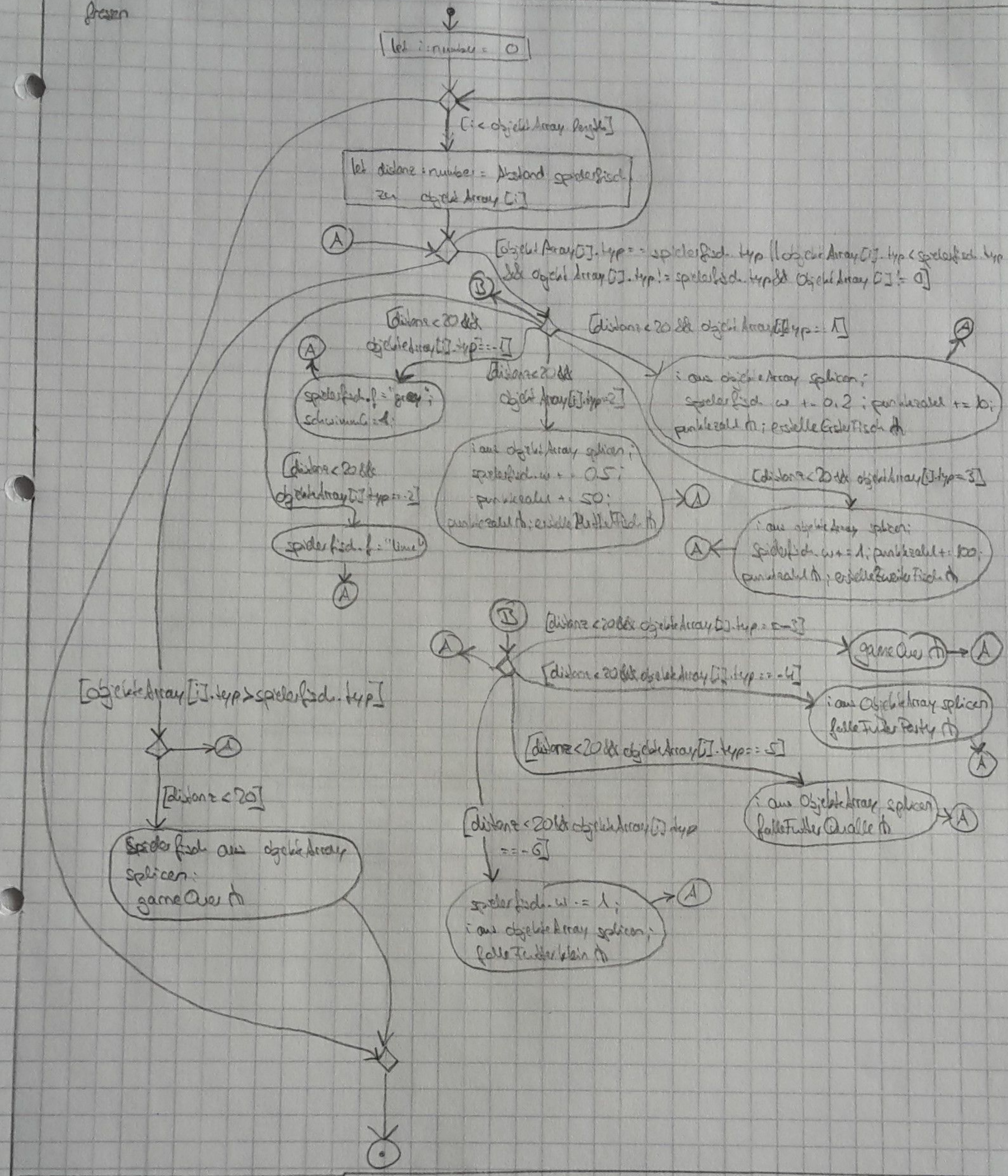
Erstellen der interaktiven Objekte am Beispiel Qualle



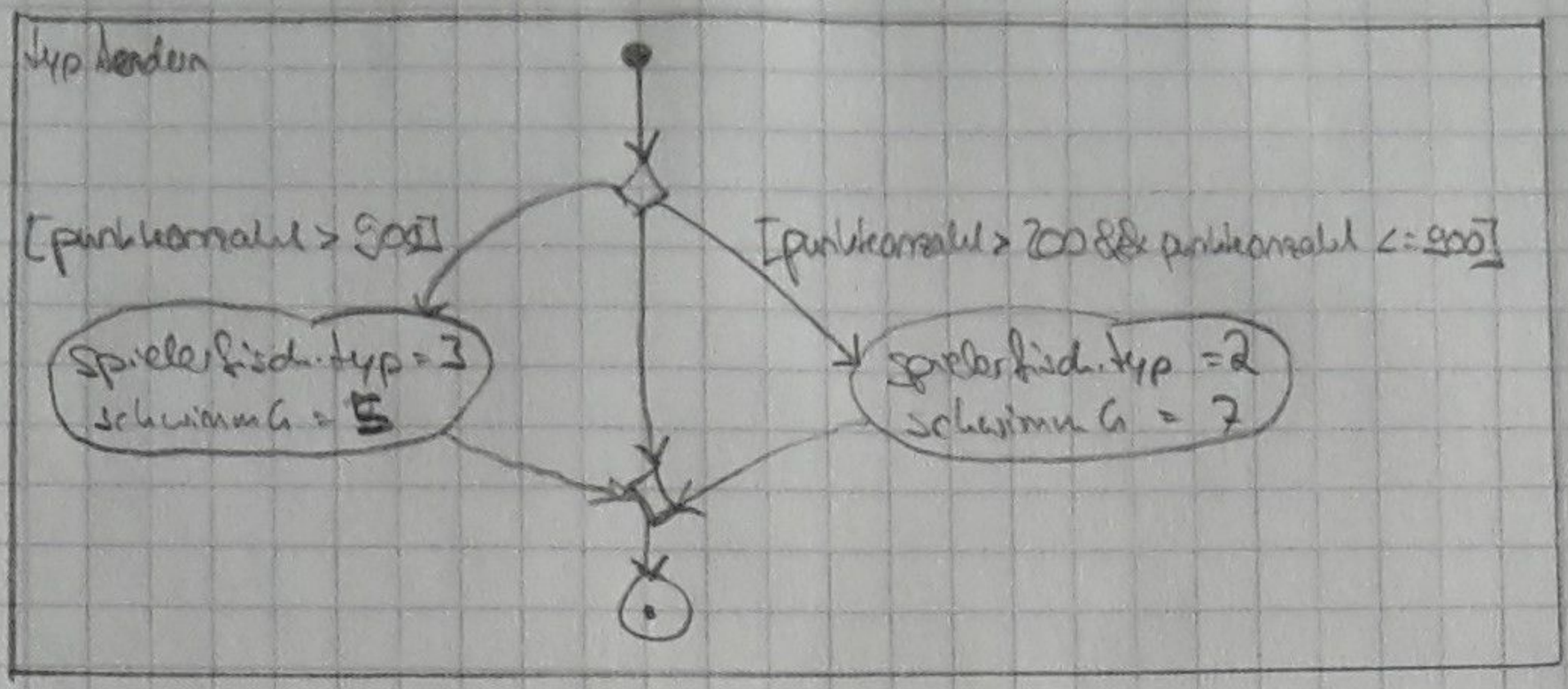
für die Steuerung nach dem der Fisch berührt wurde, werden + und - umgekehrt

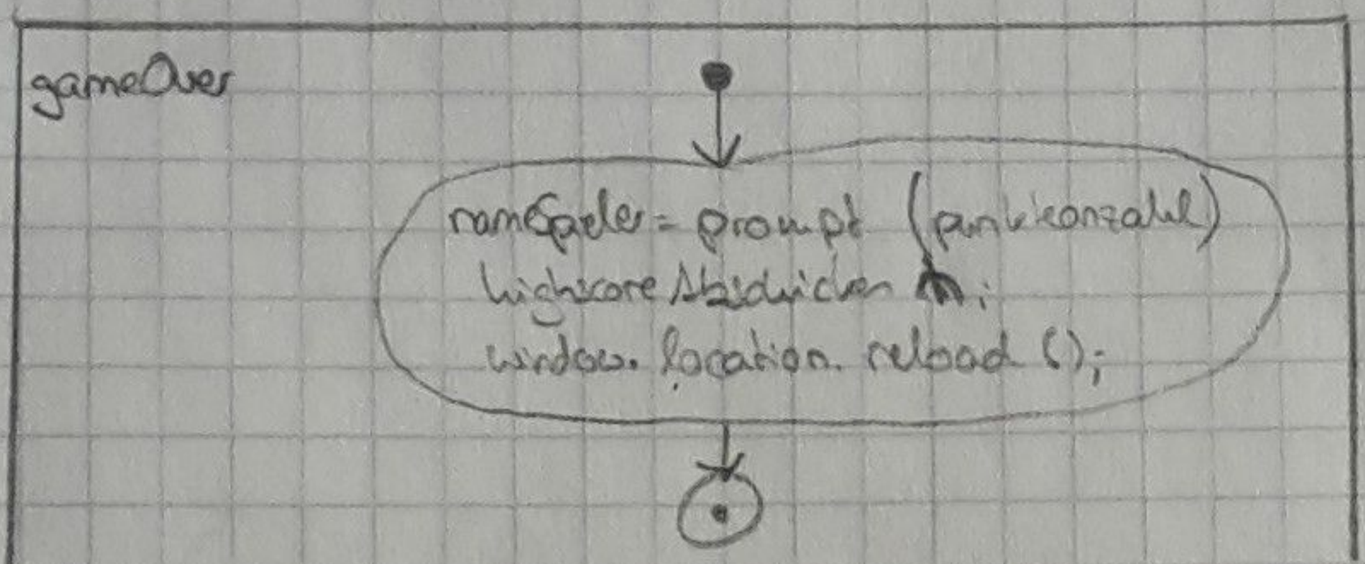


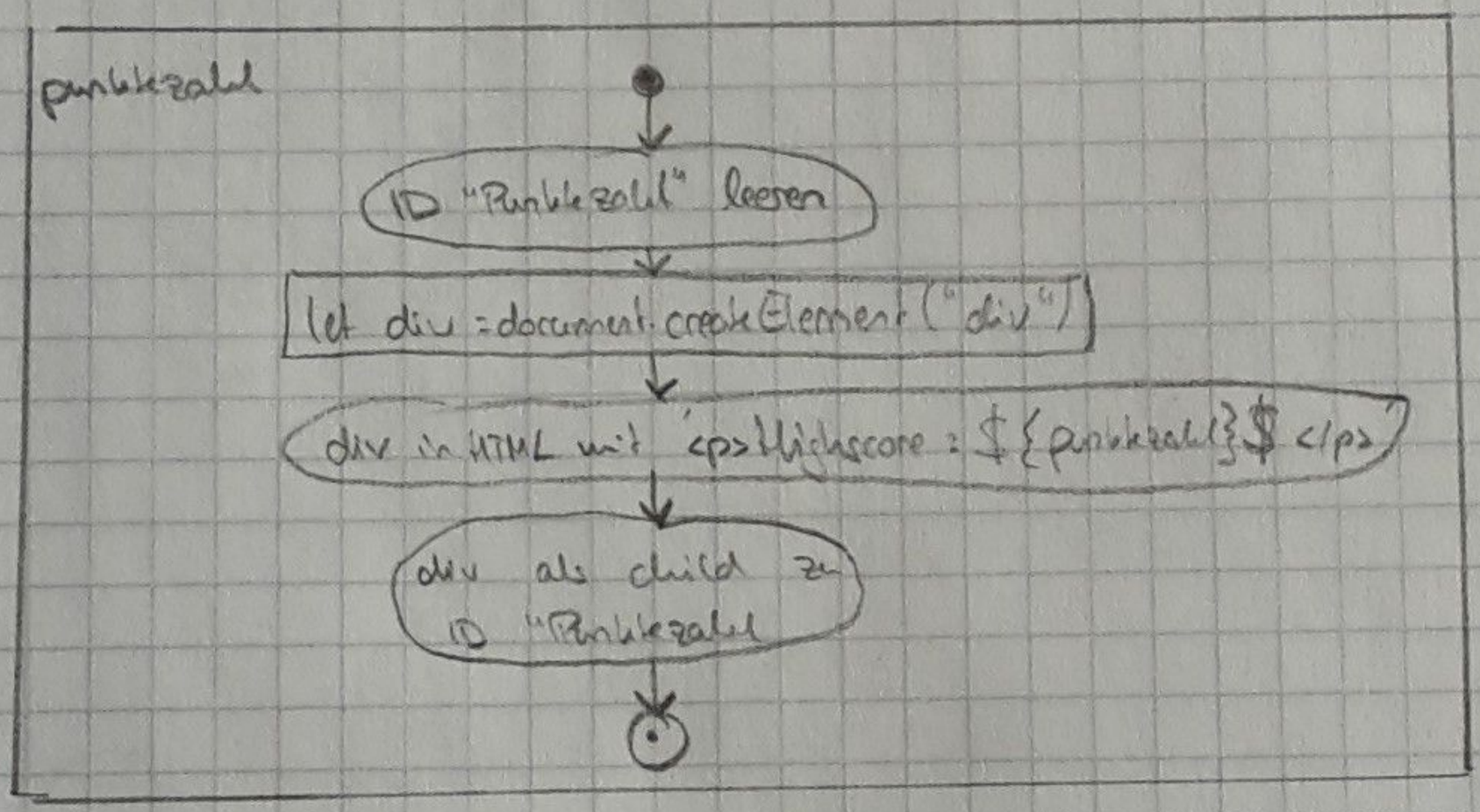
fragen


let i: number = 0
[i < objekteArray.length]
let distanz: number = Abstand spielerfisch zu objekteArray[i]
A
[objekteArray[i].typ == spielerfisch.typ || objekteArray[i].typ < spielerfisch.typ && objekteArray[i].typ != spielerfisch.typ && objekteArray[i] != 0]
B
[distanz < 20 && objekteArray[i].typ == -1]
spielerfisch.f = "grau";
schwierigkeit -= 1;
[distanz < 20 && objekteArray[i].typ == -2]
spielerfisch.f = "lime";
[distanz < 20 && objekteArray[i].typ == 2]
i aus objekteArray splicen;
spielerfisch.w += 0.5;
punktezahl += 50;
punktezahl ausgeben; erstelle Rotte Fische
[distanz < 20 && objekteArray[i].typ == 1]
i aus objekteArray splicen;
spielerfisch.w += 0.2; punktezahl += 10;
punktezahl ausgeben; erstelle Gelbe Fische
[distanz < 20 && objekteArray[i].typ == 3]
i aus objekteArray splicen;
spielerfisch.w += 1; punktezahl += 100;
punktezahl ausgeben; erstelle Zweite Fische
[objekteArray[i].typ > spielerfisch.typ]
[distanz < 20]
Spielerfisch aus objekteArray splicen:
gameOver ausgeben
C
[distanz < 20 && objekteArray[i].typ == -3]
[distanz < 20 && objekteArray[i].typ == -4]
i aus ObjekteArray splicen
falle Futter Position ausgeben
[distanz < 20 && objekteArray[i].typ == 5]
i aus ObjekteArray splicen
falle Futter Quelle ausgeben
[distanz < 20 && objekteArray[i].typ == -6]
spielerfisch.w -= 1;
i aus objekteArray splicen;
falle Futter klein ausgeben

7


typ kiezen
[punkteaantal > 200]
[punkteaantal > 200 && punkteaantal <= 500]
spelerfiscl.typ = 3
schuimen = 5
spelerfiscl.typ = 2
schuimen = 7


gameOver
naamSpeler = prompt (punkteaantal)
highscore Afdrukken;
window.location.reload();


punkteZaal
ID "PunkteZaal" lezen
let div = document.createElement("div")
div in HTML met Highscore : ${punkteZaal}$
div als child aan ID "PunkteZaal

User

Client

Server

Datenbank

Name eingeben
Highscore absenden

Inputdaten annehmen

url aus Inputdaten formen

url als request an Server senden

request als request annehmen

Daten aus URL in request übernehmen + an Datenbank übergeben

request empfangen + speichern

alle Daten an Server senden

Daten empfangen + response erstellen

response an Client schicken

response empfangen

aus empfangenen Daten Array erstellen

Array nach höchsten Highscore sortieren

5 höchsten Highscores in ID "Bestenliste" in HTML darstellen

Bestenliste ansehen

Domänen übergreifender AJAX