

FUNKTIONALE ANALYSE

- Nutzer soll mit Pfeiltasten beweglichen Fisch steuern.
- Nutzer soll Spaß dabei haben, Fische zu fressen.

Plattform

- Soll PC sein, da mehr Platz zur Verfügung steht.
- Man sieht mehr vom Canvas und steuert ohne das man Canvas verdecken muss. (Pfeiltasten VS. Finger)

Interaktion Objekte untereinander

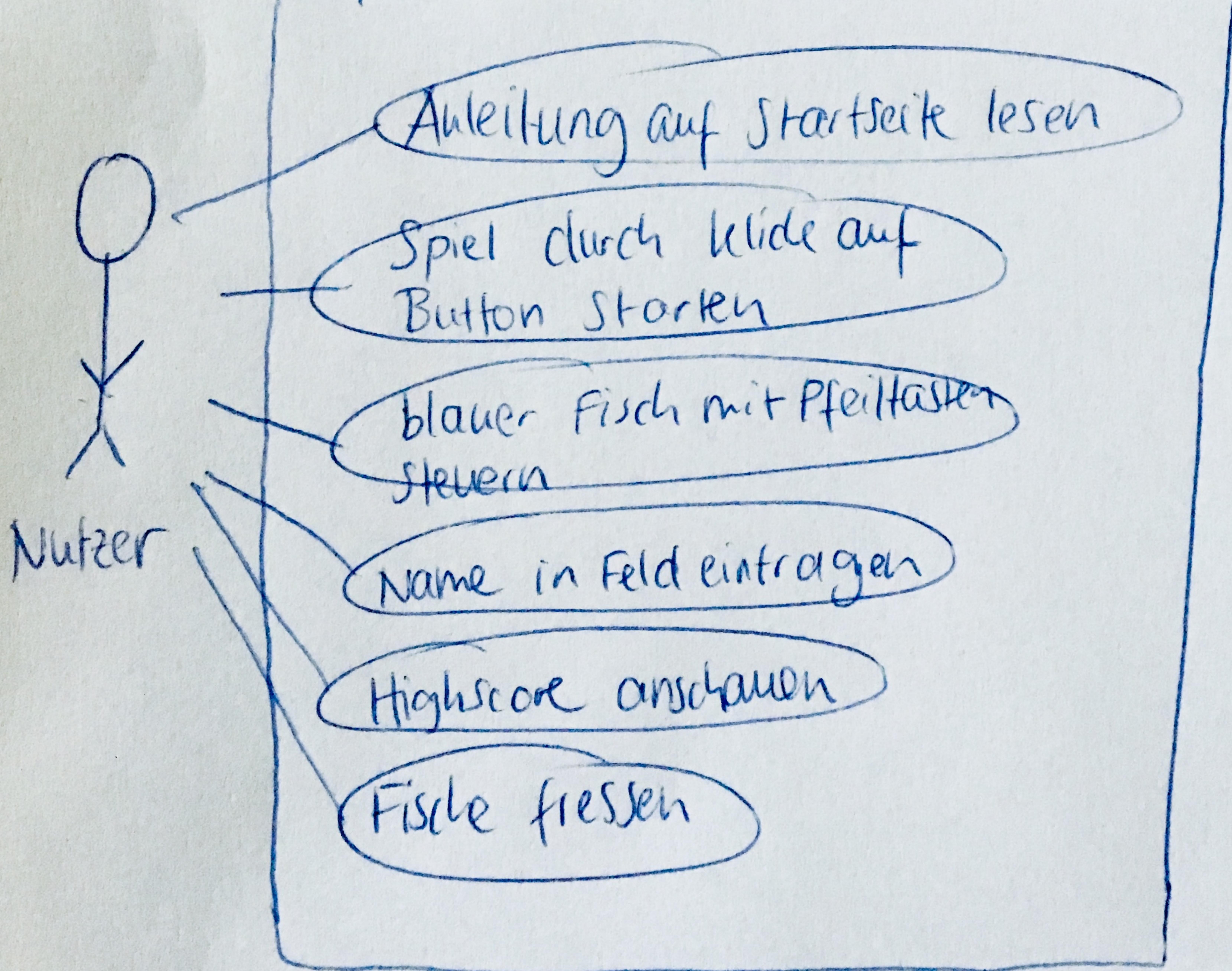
- Blauer steuerbarer Fisch muss erst alle 7 kleineren grüne Fische fressen.
- Danach kann er die 7 großen, lila-gelben Fische fressen.
- Bei jedem gefressenen ~~Fisch~~ Fisch, wächst der spielbare blaue Fisch.
- Große, lila-gelbe Fische, können auch blauen Fisch töten/fressen, wenn er noch nicht alle 7 grünen Fische gefressen hat.
- Werden alle Fische gefressen, hat man gewonnen & Highscore von 80.
- Werden Fische berührt/getragen verschwinden diese.

Aufbau/Ablauf aus Sicht des Nutzers

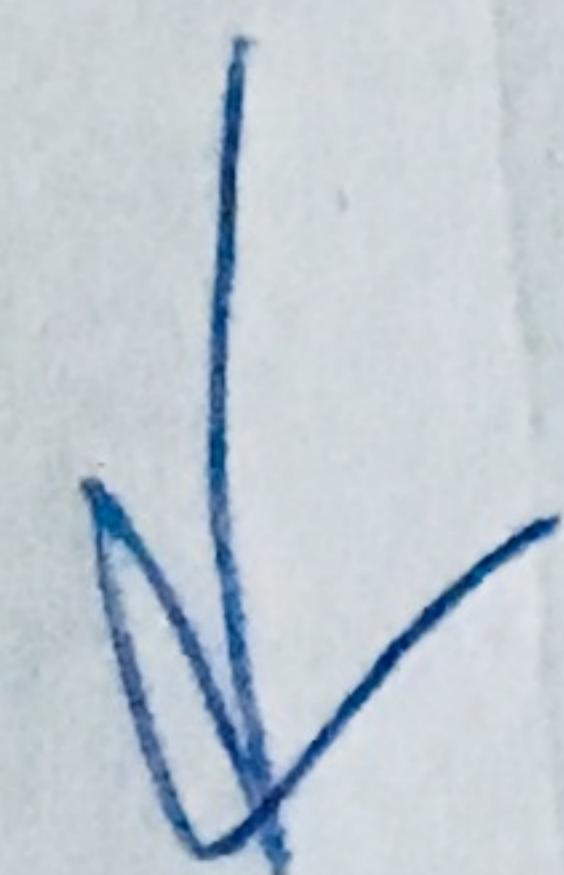
- Nutzer liest erst Spielanleitung auf Startseite. Er kann dann über Klick auf den "Spiel starten"-Button das Spiel beginnen. Nun kann der Nutzer mit den Pfeiltasten den blauen Fisch ~~vor~~ nach oben, rechts, links und unten steuern. Erst müssen die 7 kleinen grünen Fische gefressen werden, damit die großen lila-gelben Fische gefressen werden können. Diese kann man einfach fangen bzw. dann mit Berührung vom blauen Fisch, an den jeweiligen fangenden Fischen auslösen. Berührt man vorher, bevor man alle 7 grünen Fische gefangen/gefressen hat, einen großen lila-gelben Fisch stirbt man. Man hat die Möglichkeit seinen Namen dann in ein Prompte eintragen, ~~diesen~~ Den erreichten Score, zeigt es dann mit Namen rechts in der Highscoreliste neben dem Canvas an. Werden alle Fische gefressen hat man gewonnen, bzw. den höchsten Punktestand erreicht, 80 Punkte.

Anwendungsfalldiagramm

System Canasunterwassergame



Technische
Analyse



Startseite.html

```
<html>
<header>
<h1> WILLKOMMEN IM INTERAKTIVEN UNTERWASSER-GAME </h1>
</header>
<main>
<p>
```




```
</p>
```

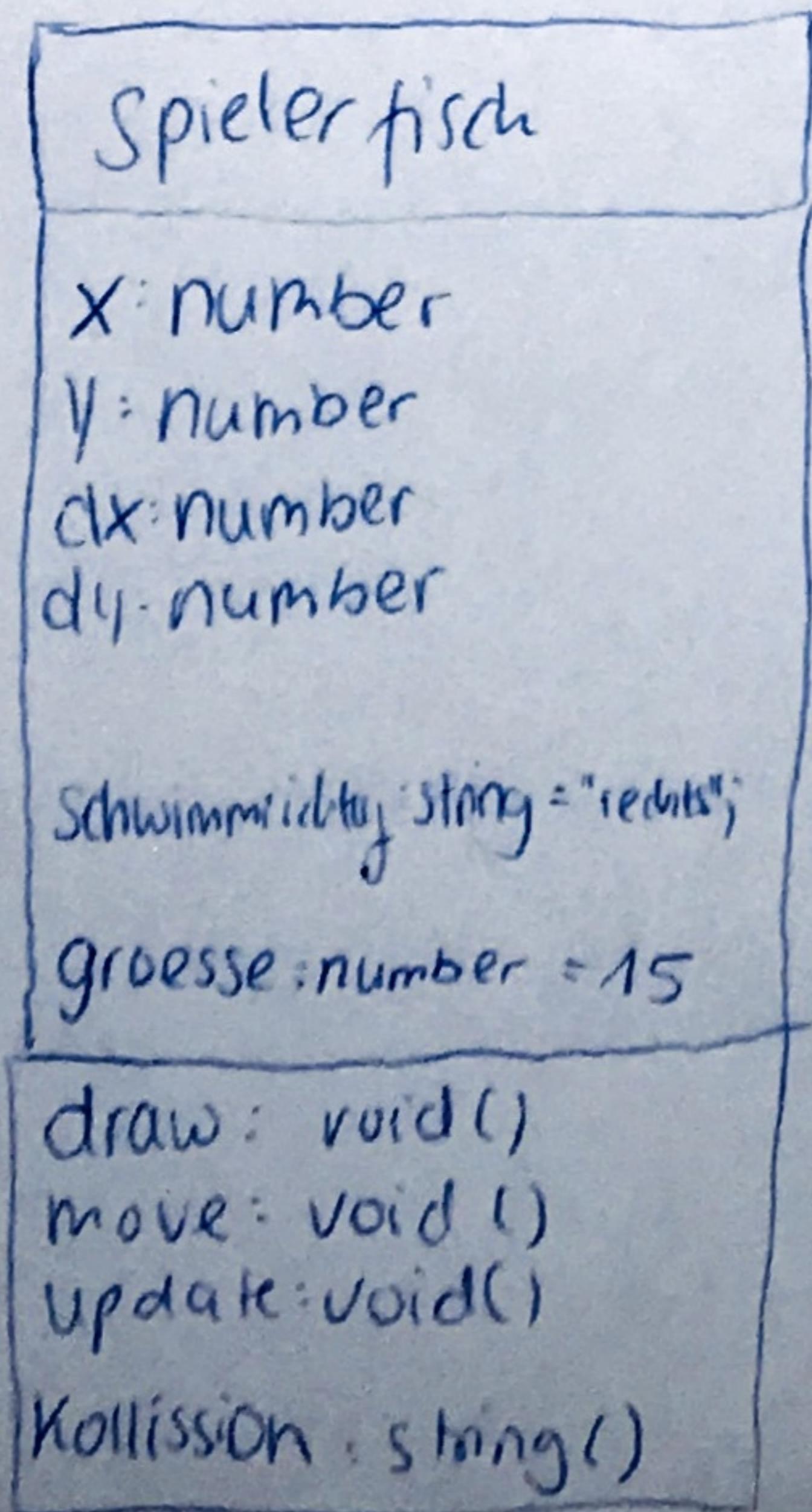
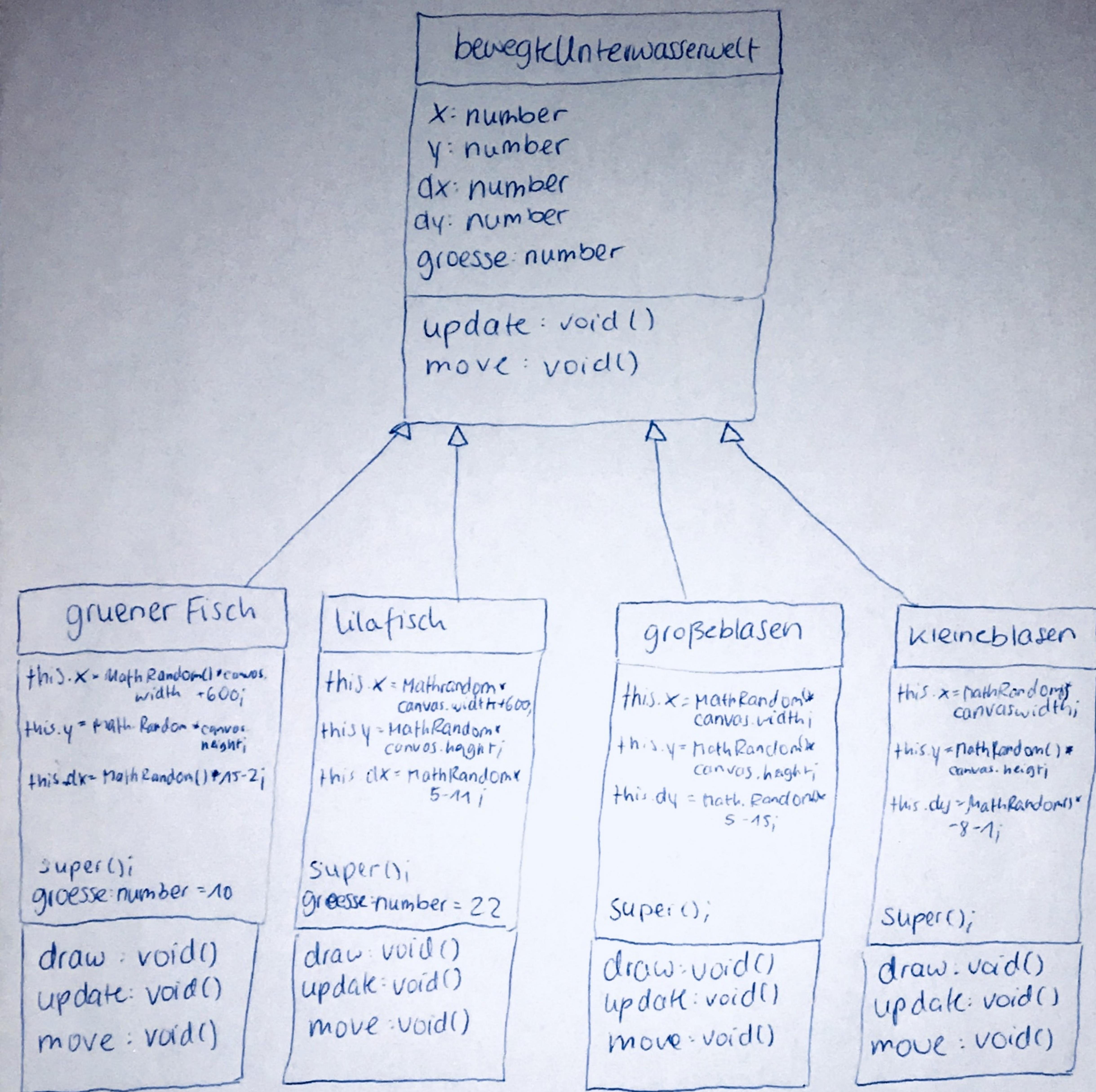
LET THE GAME BEGIN

```
</main>
</html>
```

<button>
mit Link zur
canvas.html</button>



Klassendiagramme



Hauptprogramm

```
document.addEventListener("DOMContentLoaded", init) {
```

```
    let fps: number = 25;  
    let spielerfisch: Spielerfisch;  
    let imageData: ImageData;  
  
    export let crc: CanvasRenderingContext2D;  
    export let canvas: HTMLCanvasElement;  
    export let bewegtUnterwasserweltArray: BewegtUnterwasserwelt[];  
    export let Punktestand: number = 0;  
    export let Spielername: String;  
    export let timer: number;
```

```
Σ keydown
```

```
→ moveSpielerFisch()
```



init

```
Canvas = document.getElementById("canvas")[0];
cgc = canvas.getContext("2D");
```

refresh()

Zeichne Hintergrund()

```
imageData = cgc.getImageData(0,0,canvas.width,canvas.height);
Spielerfisch = new SpielerFisch();
```

Spielerfisch.draw()

document.addEventListener("keydown", moveSpielerFisch)

```
let i:number = 0;
```

[$i \leq 7$]

```
let gruen: GruenerFisch
```

```
gruen = new GruenerFisch();
```

bewegteUnterwasserweltArray.push(gruen)

gruen.draw()

```
let i:number = 0;
```

[$i < 7$]

```
let lila: LilaFisch
```

```
lila = new LilaFisch();
```

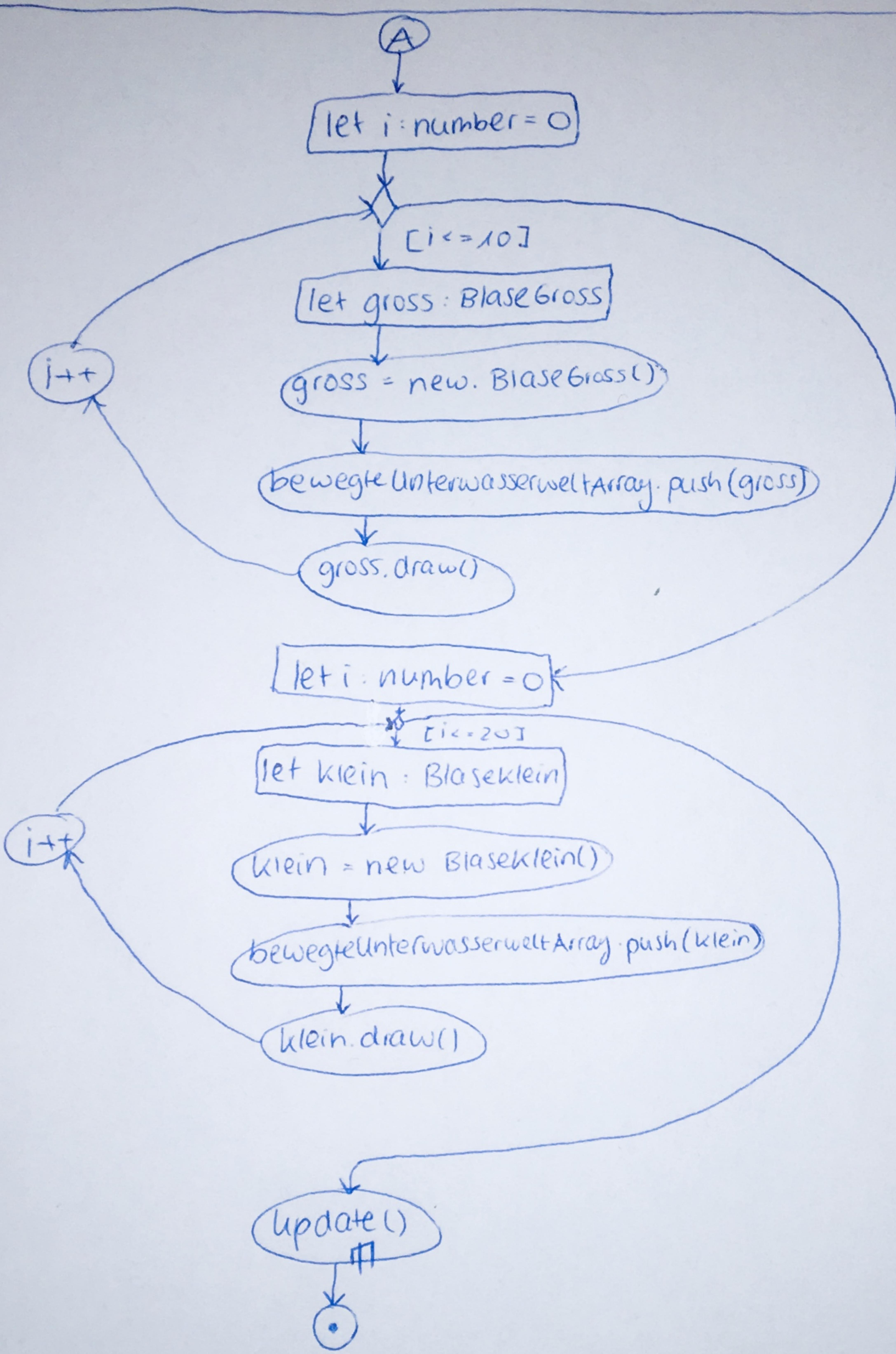
bewegteUnterwasserweltArray.push(lila)

lila.draw()


j++

A


init

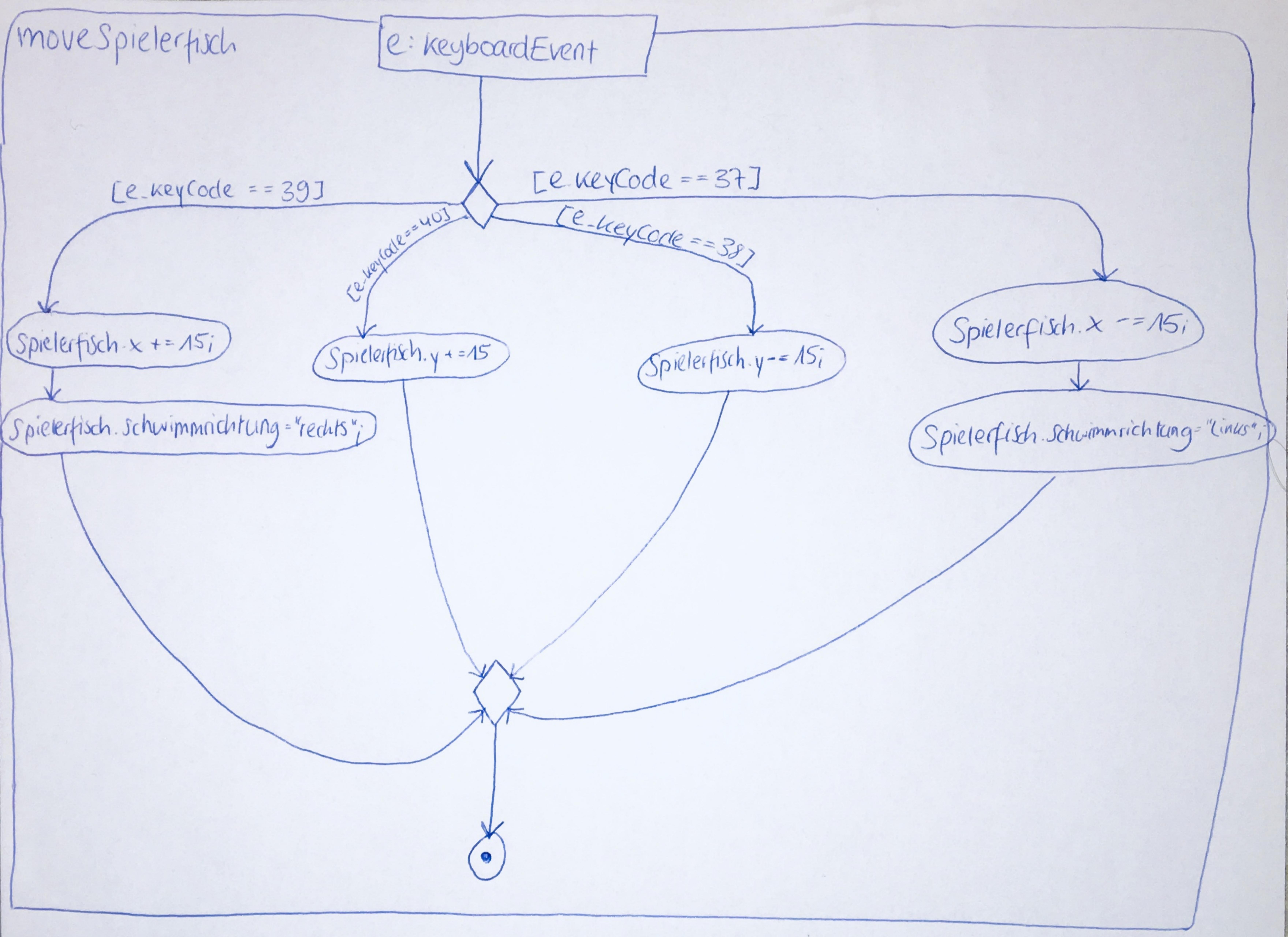


update

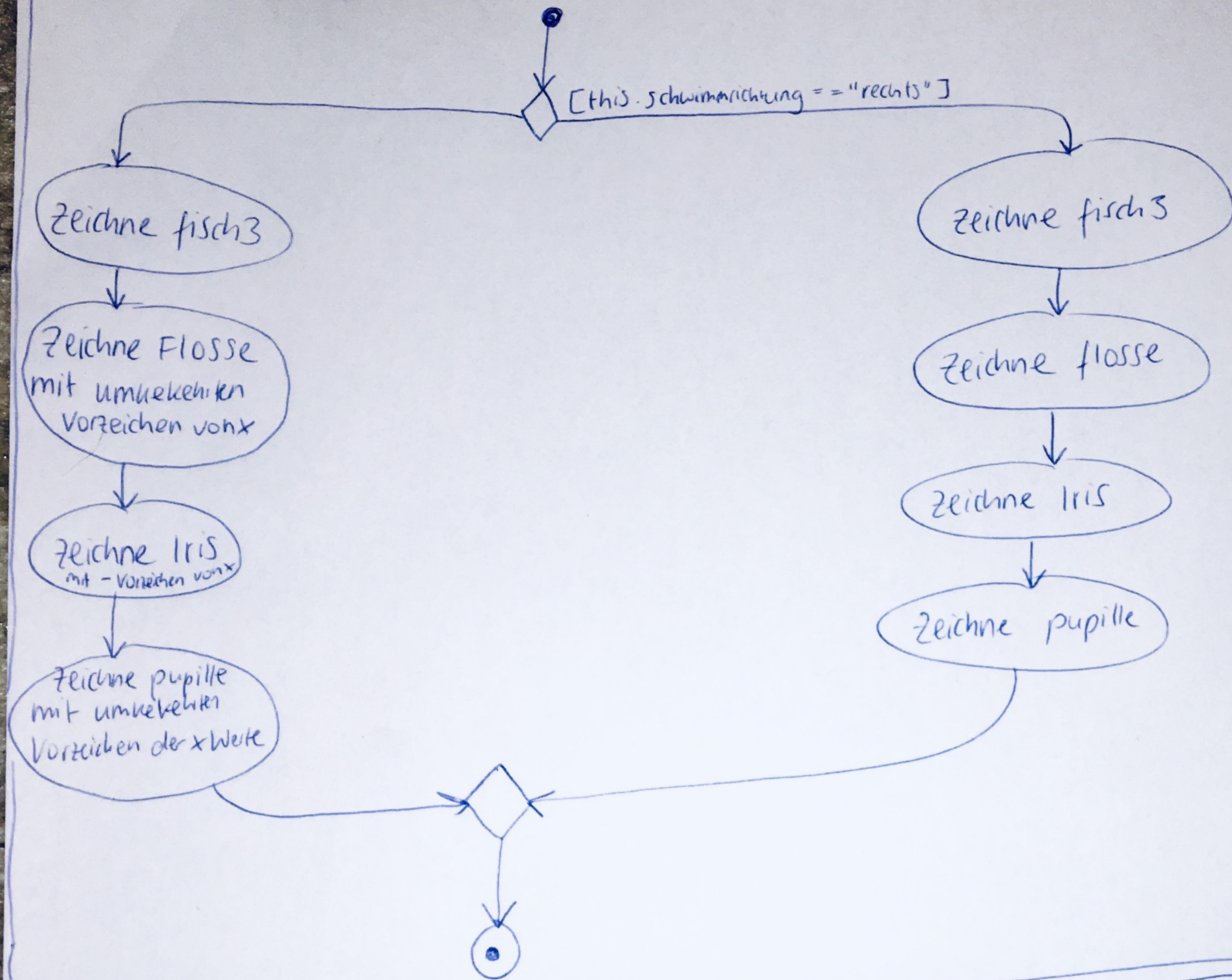


Zeichne Hintergrund

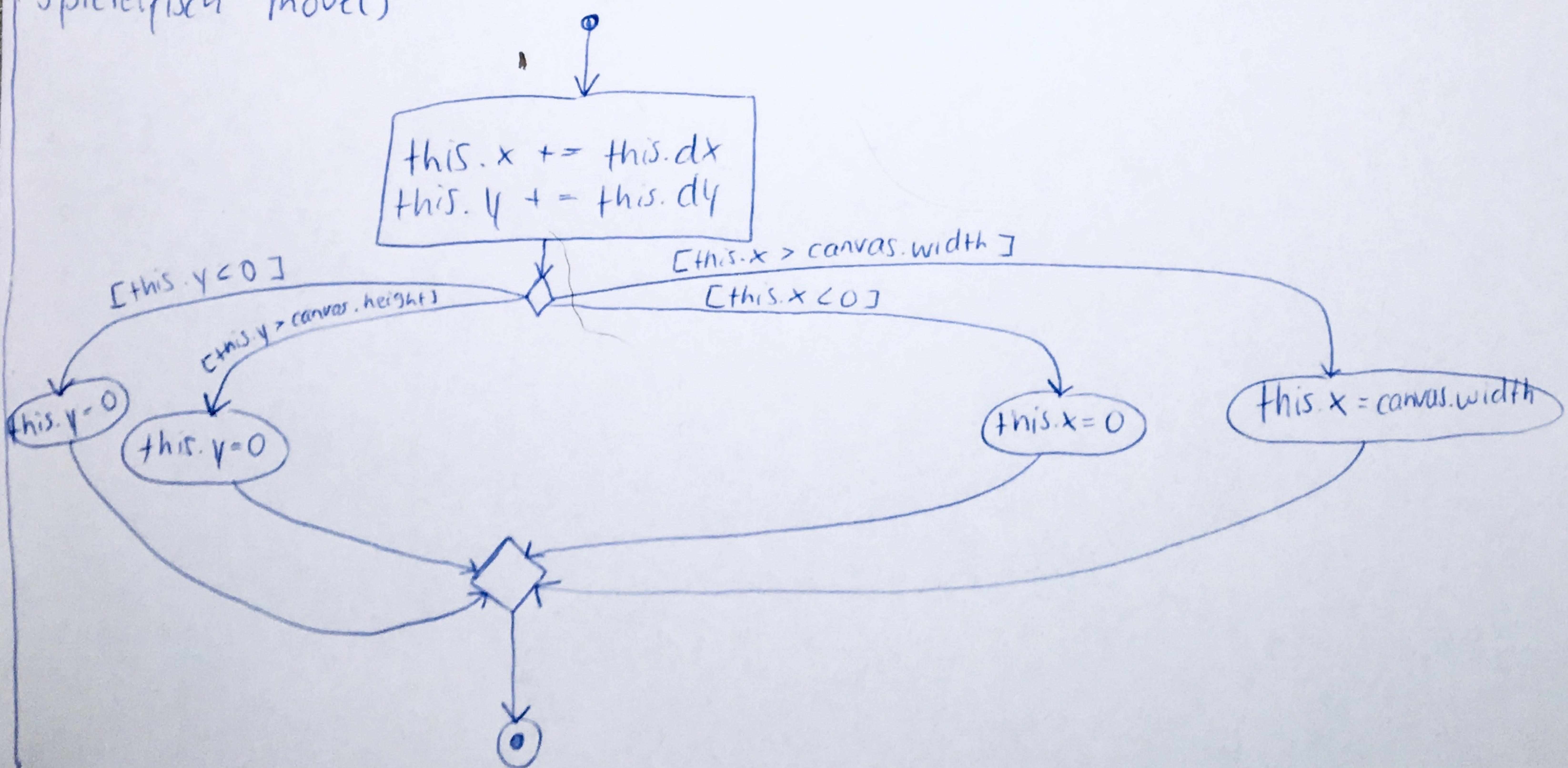




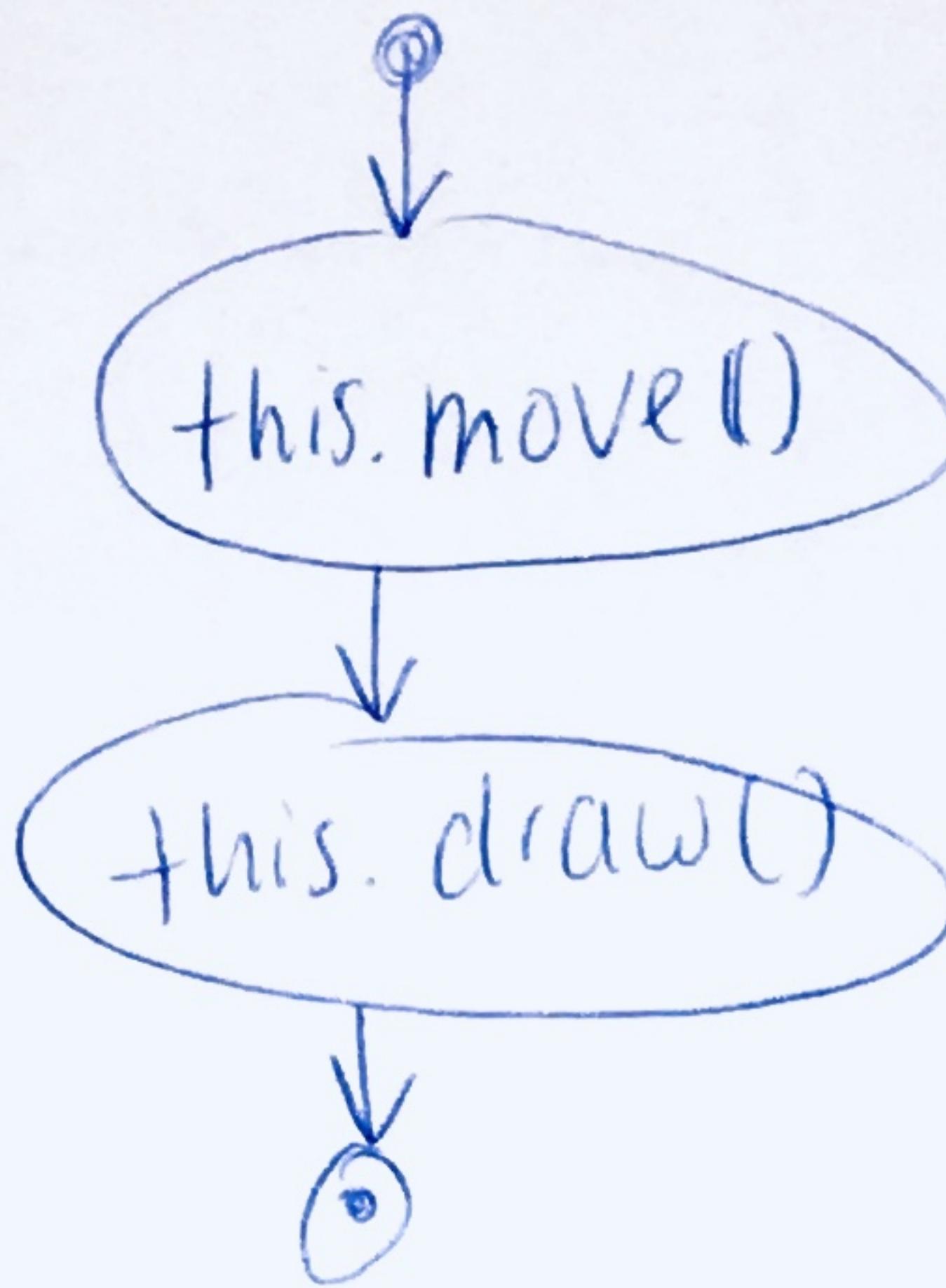
Spieldfisch draw()




Spieldfisch move()

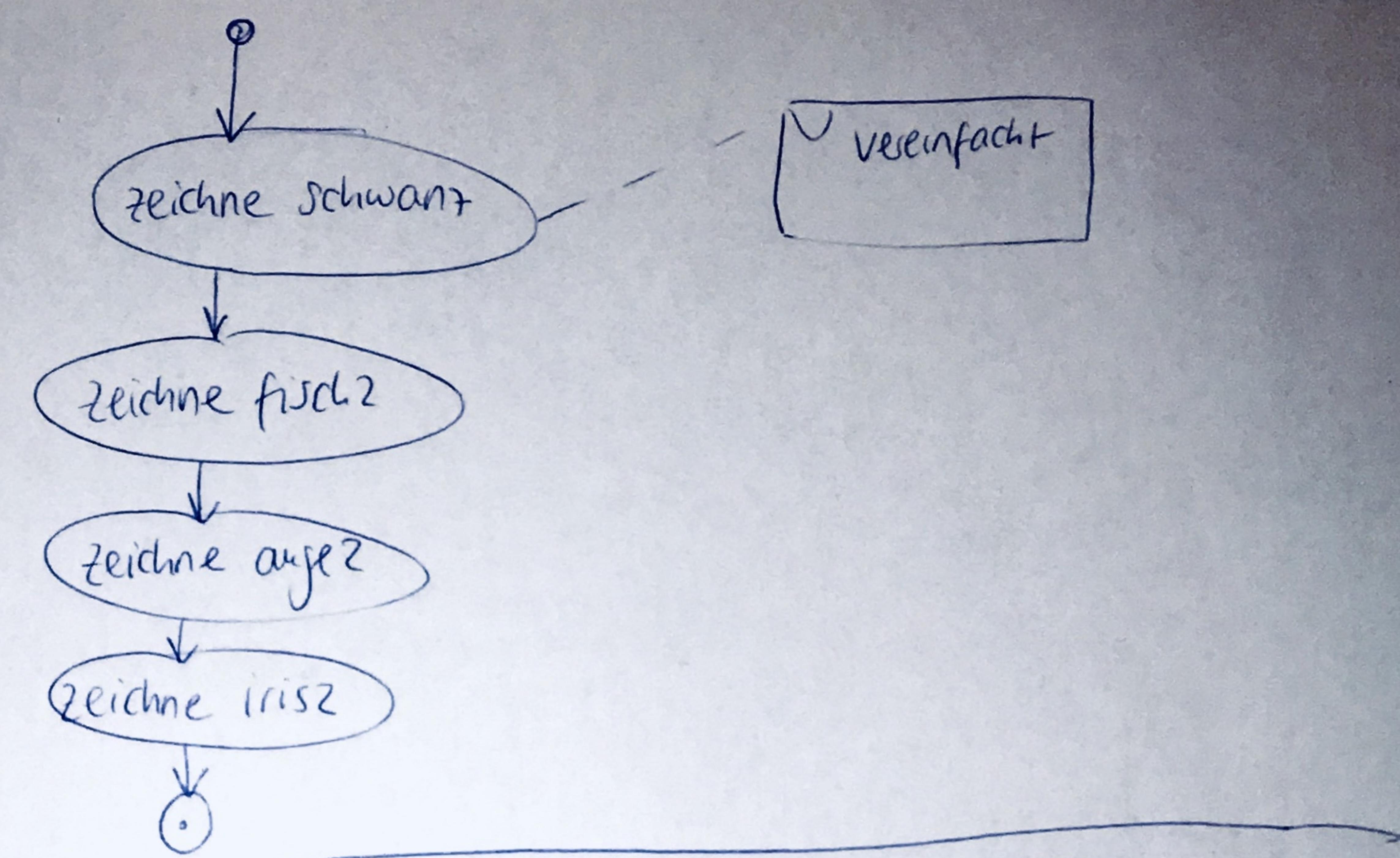


Spielefisch update()

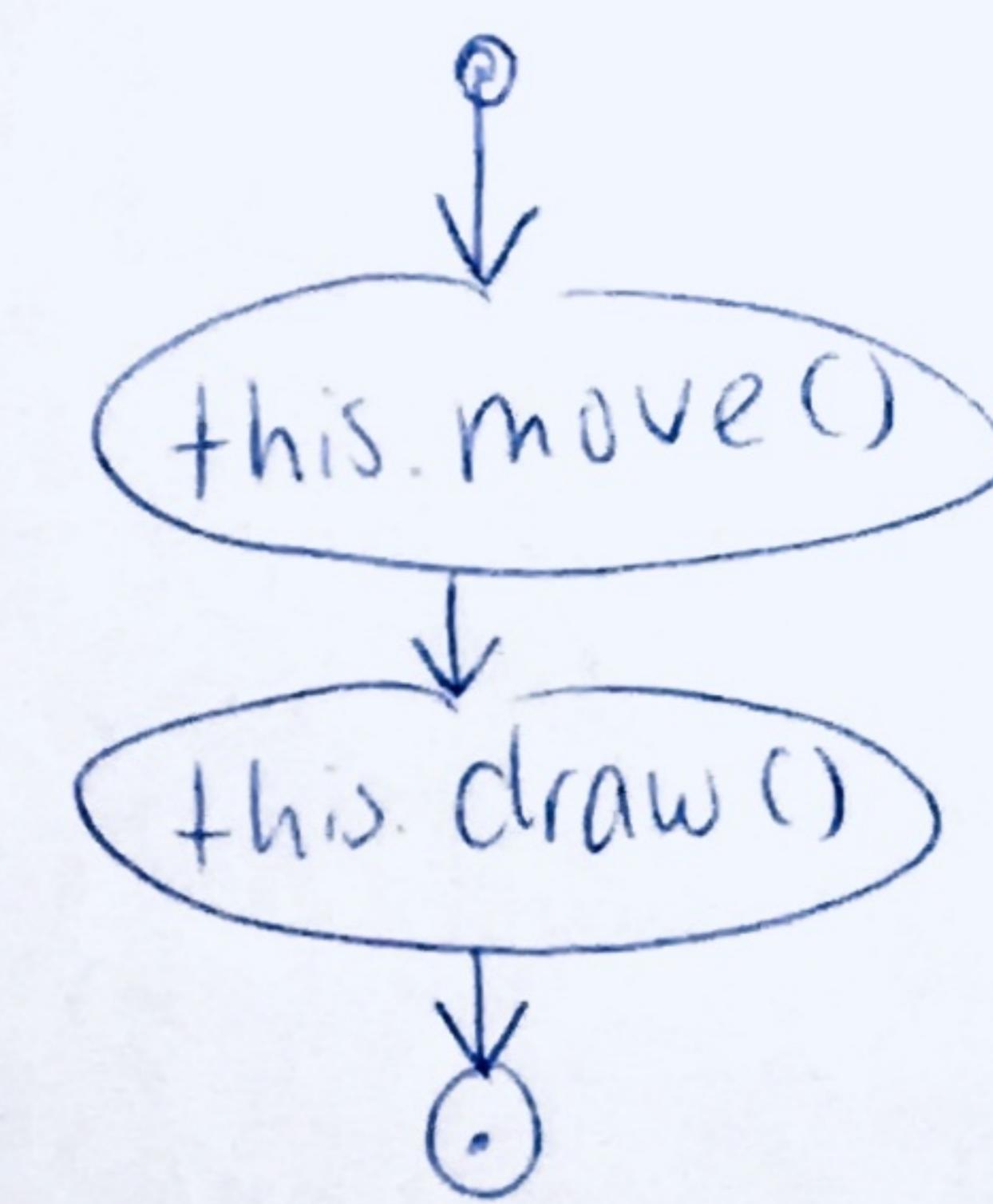




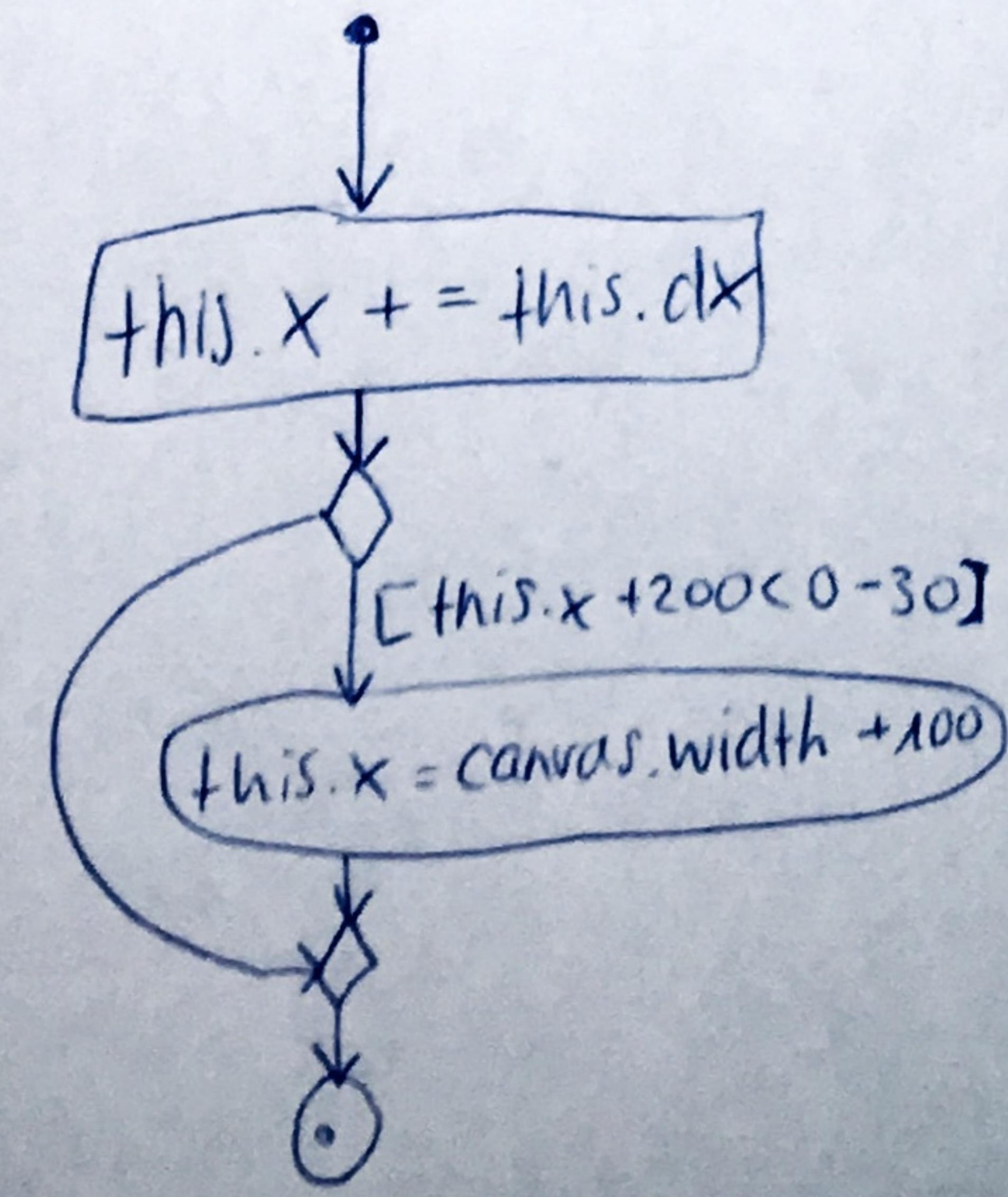
lilafisch draw()



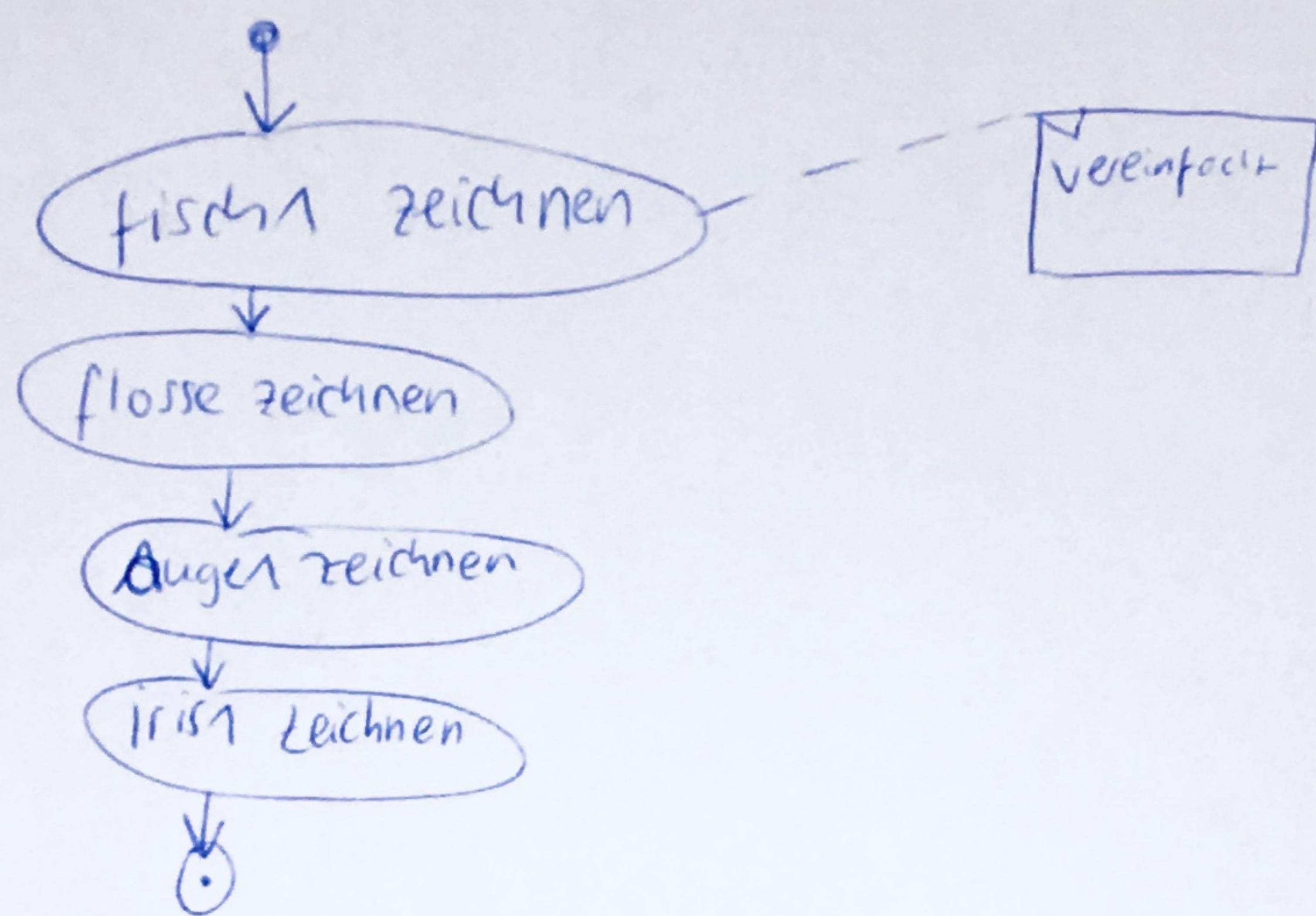
lilafisch update()



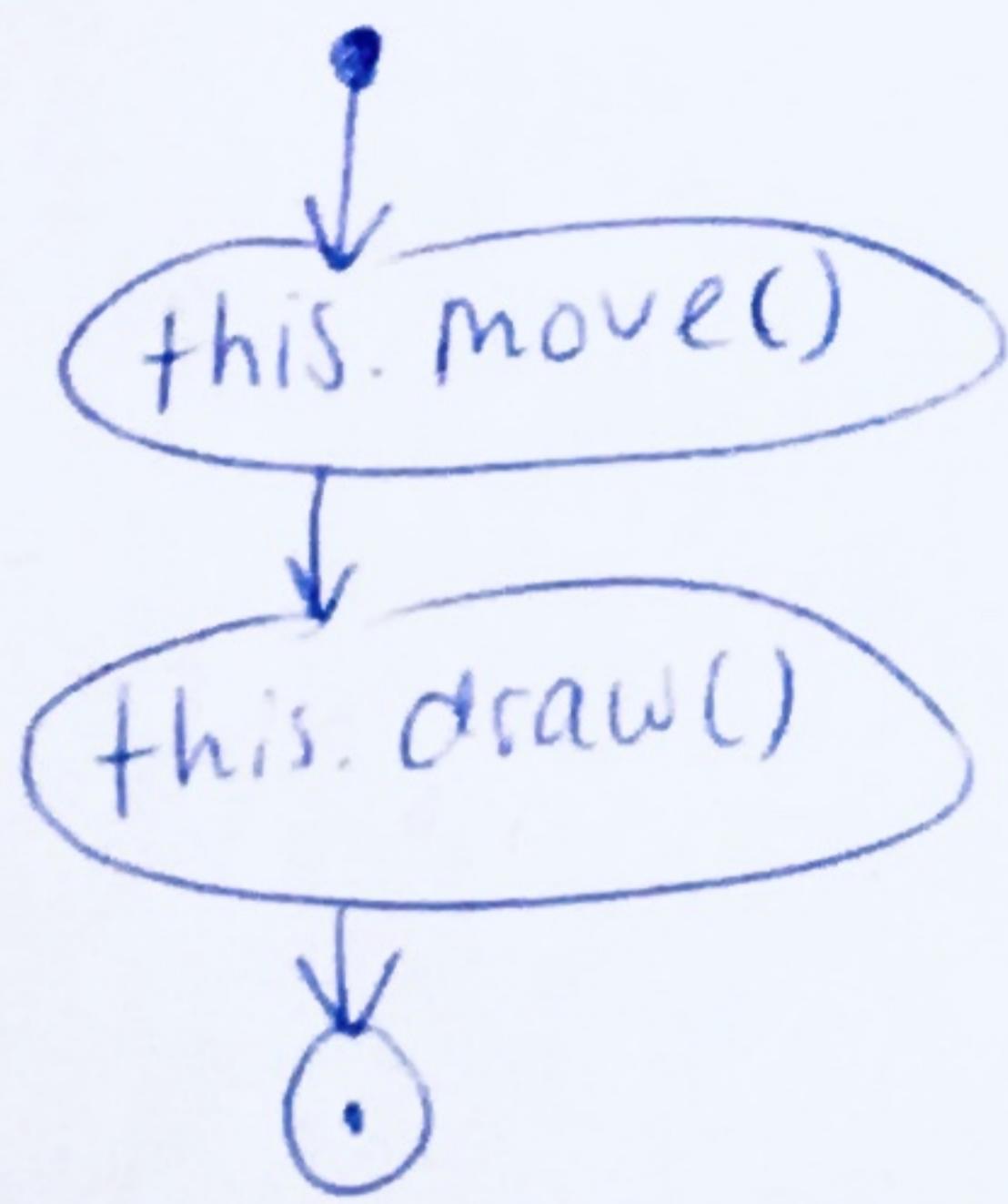
lilafisch move()



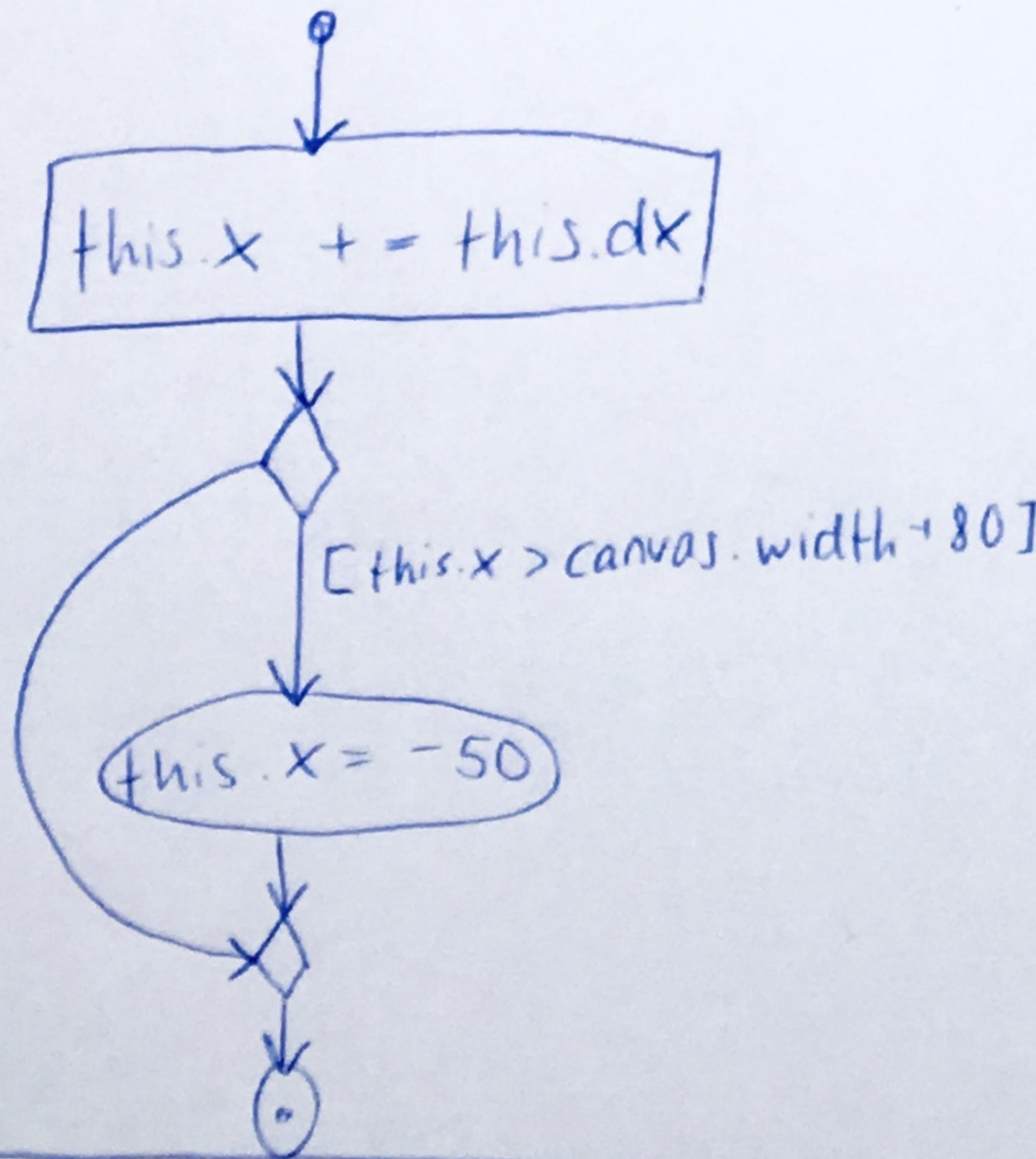
gruenerfisch. draw()



gruenerfisch. update()

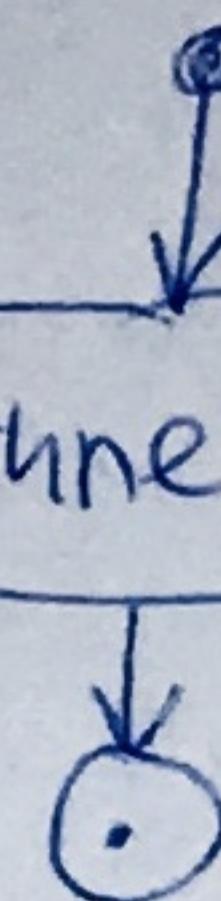


gruenerfisch. move()



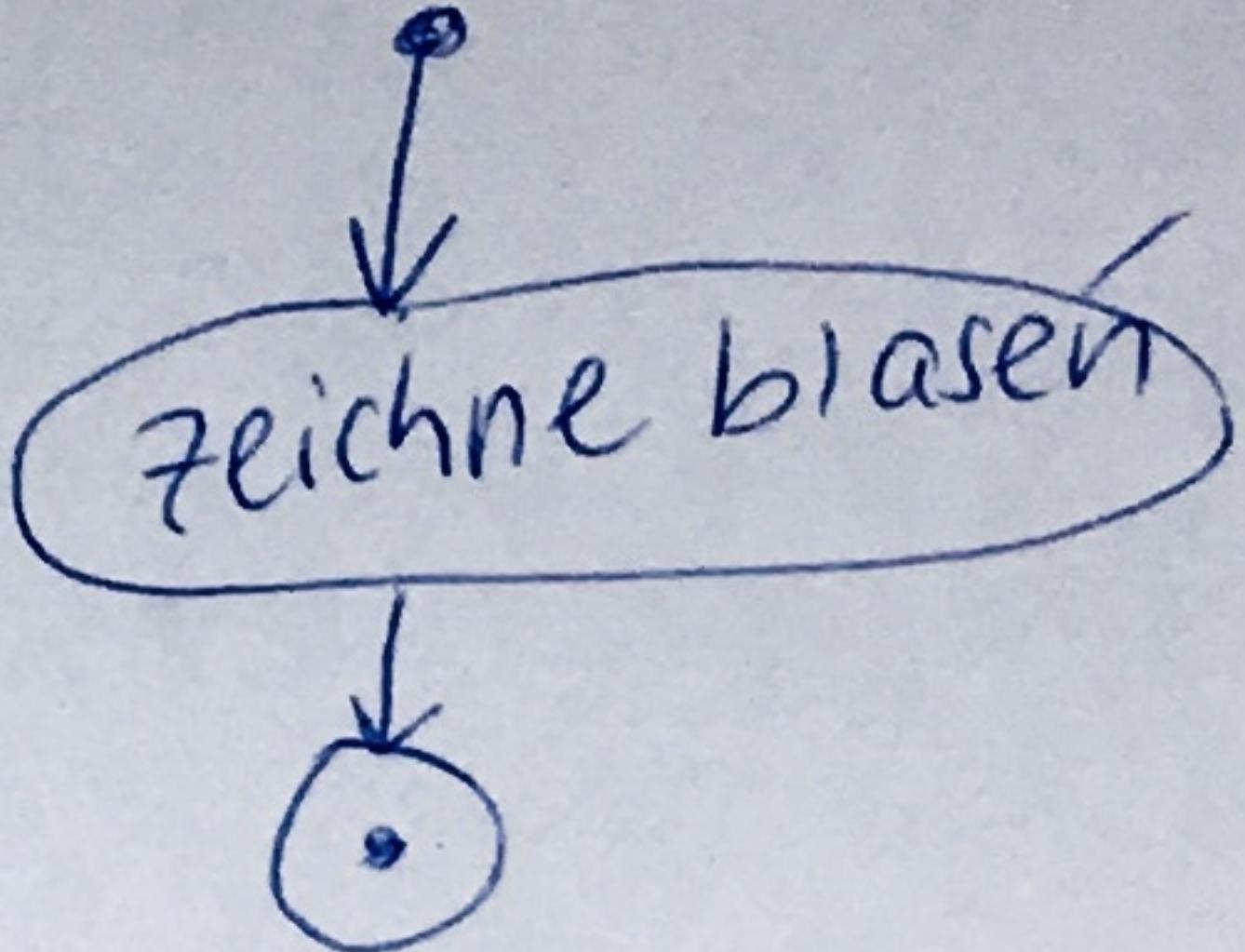
Blasegross. draw()

Vereinfacht

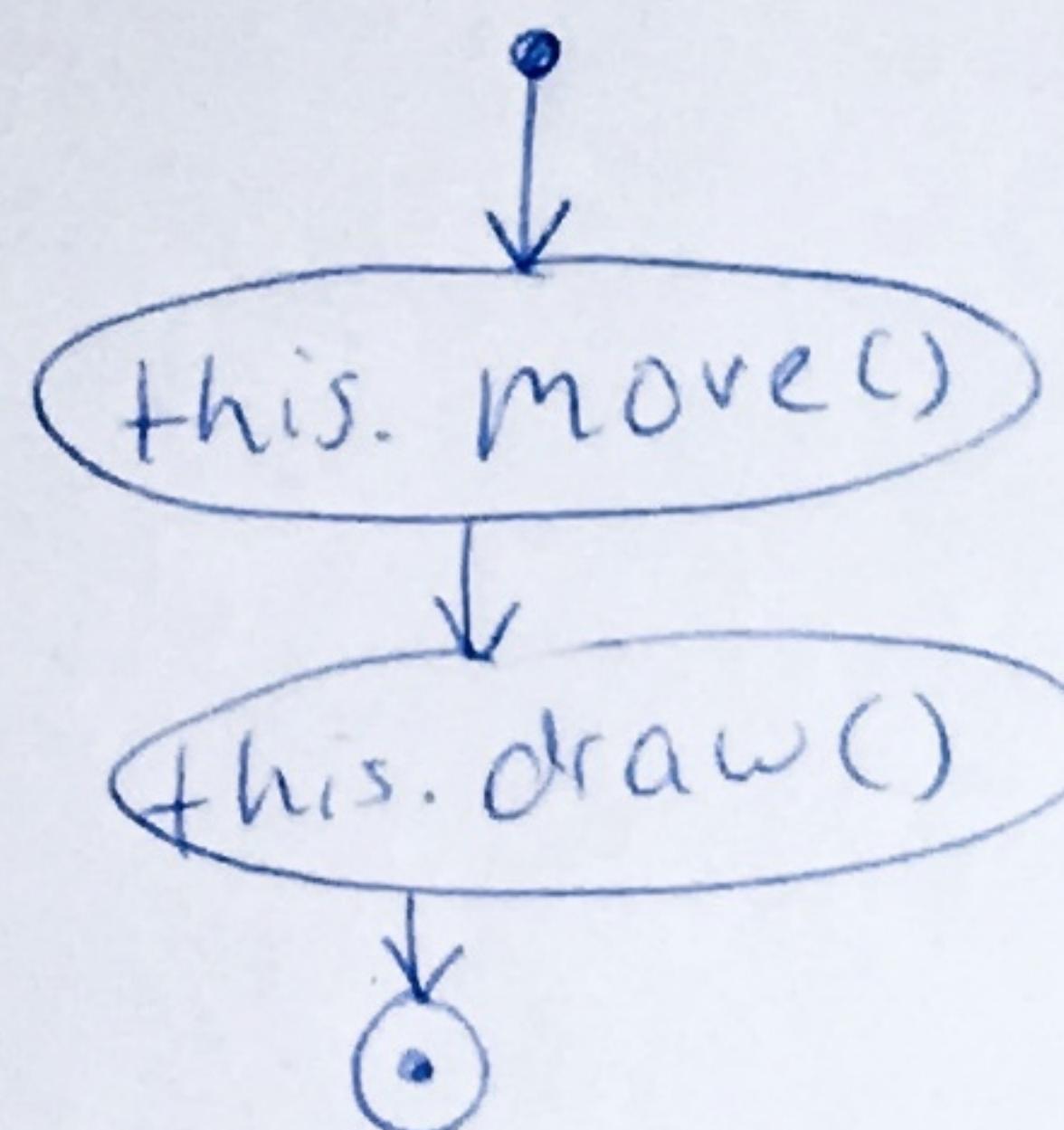


Blaseklein. draw()

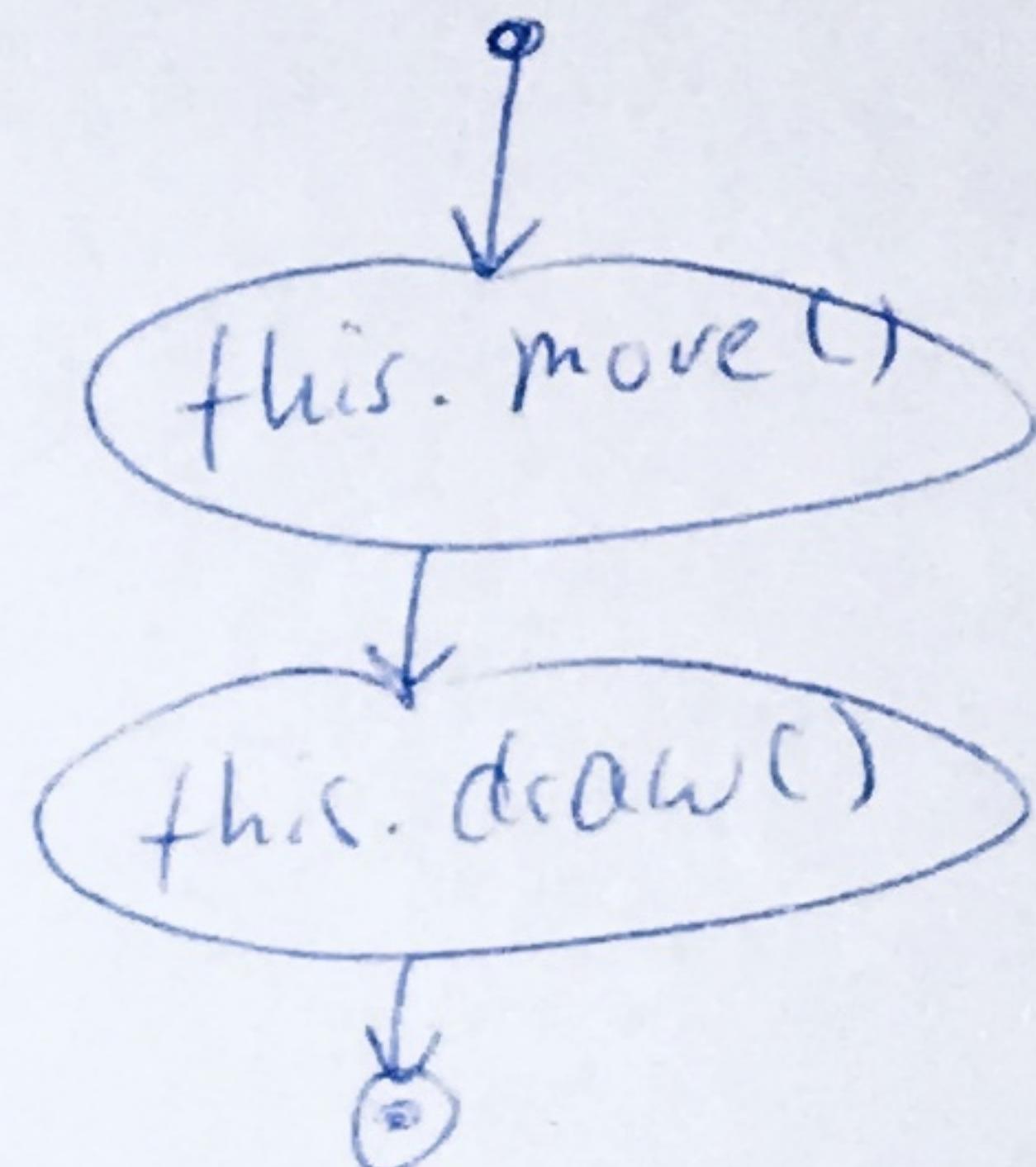
Vereinfacht



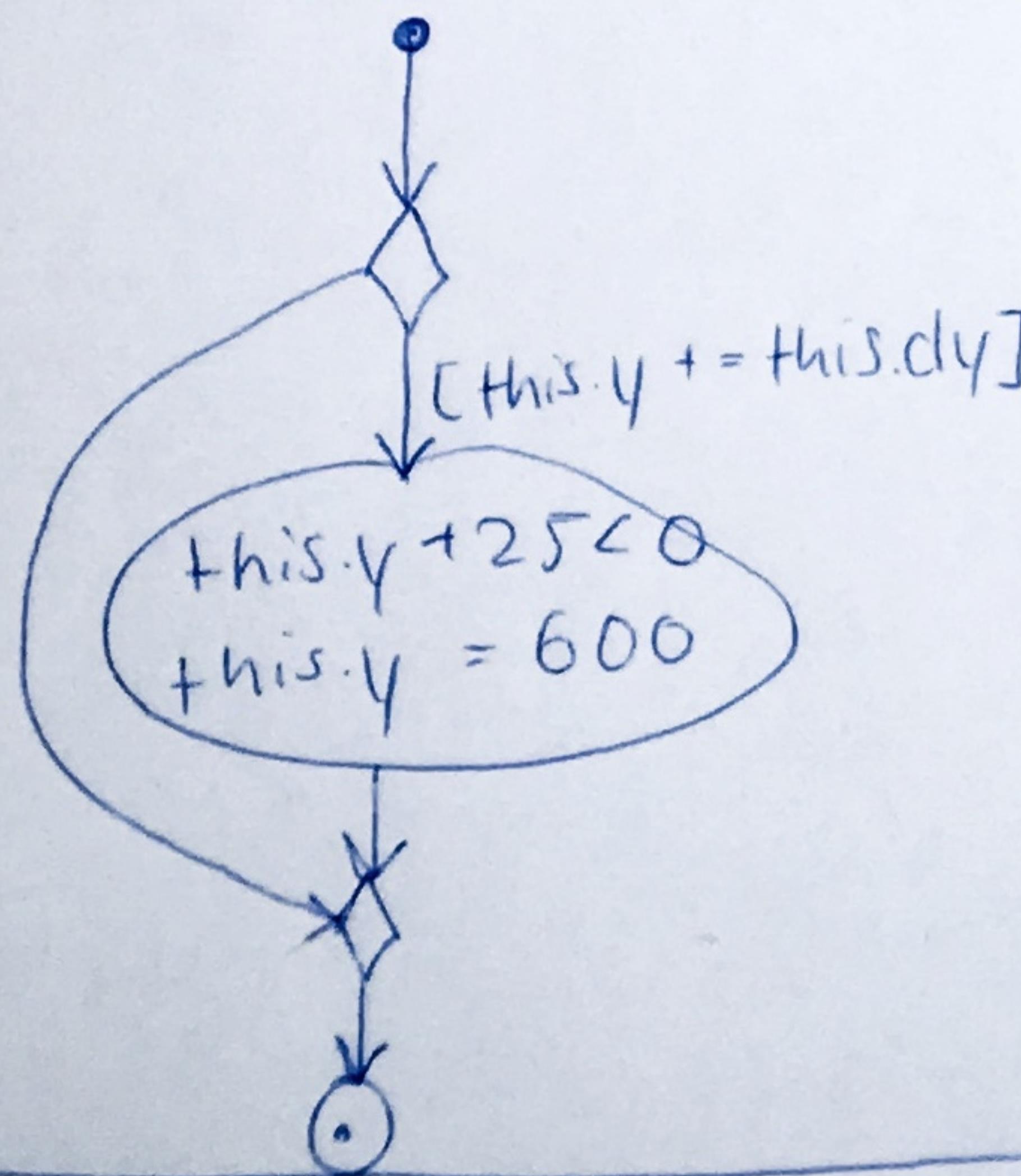
Blasegross. update()



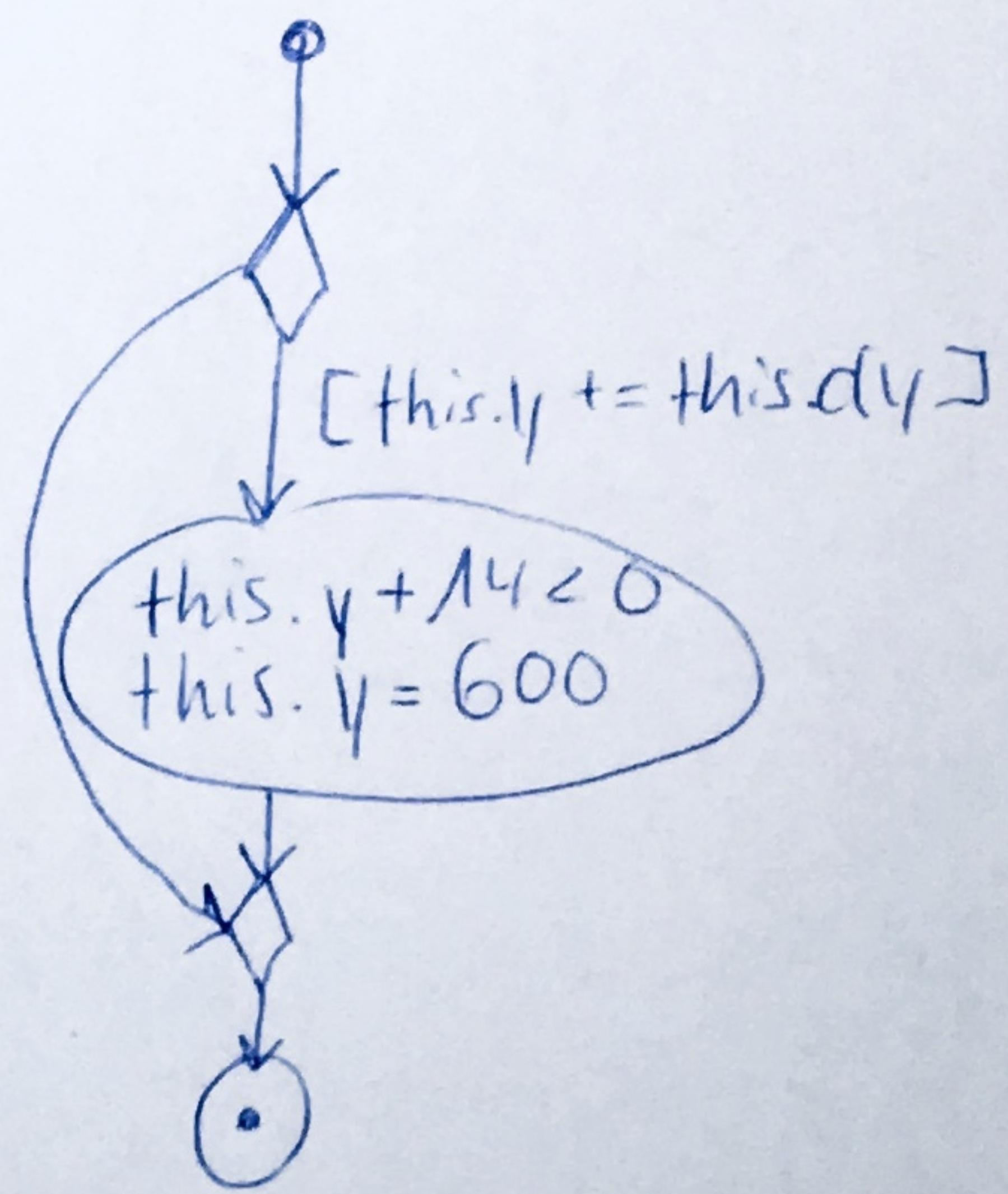
Blaseklein. update()




Blase Gross . move()

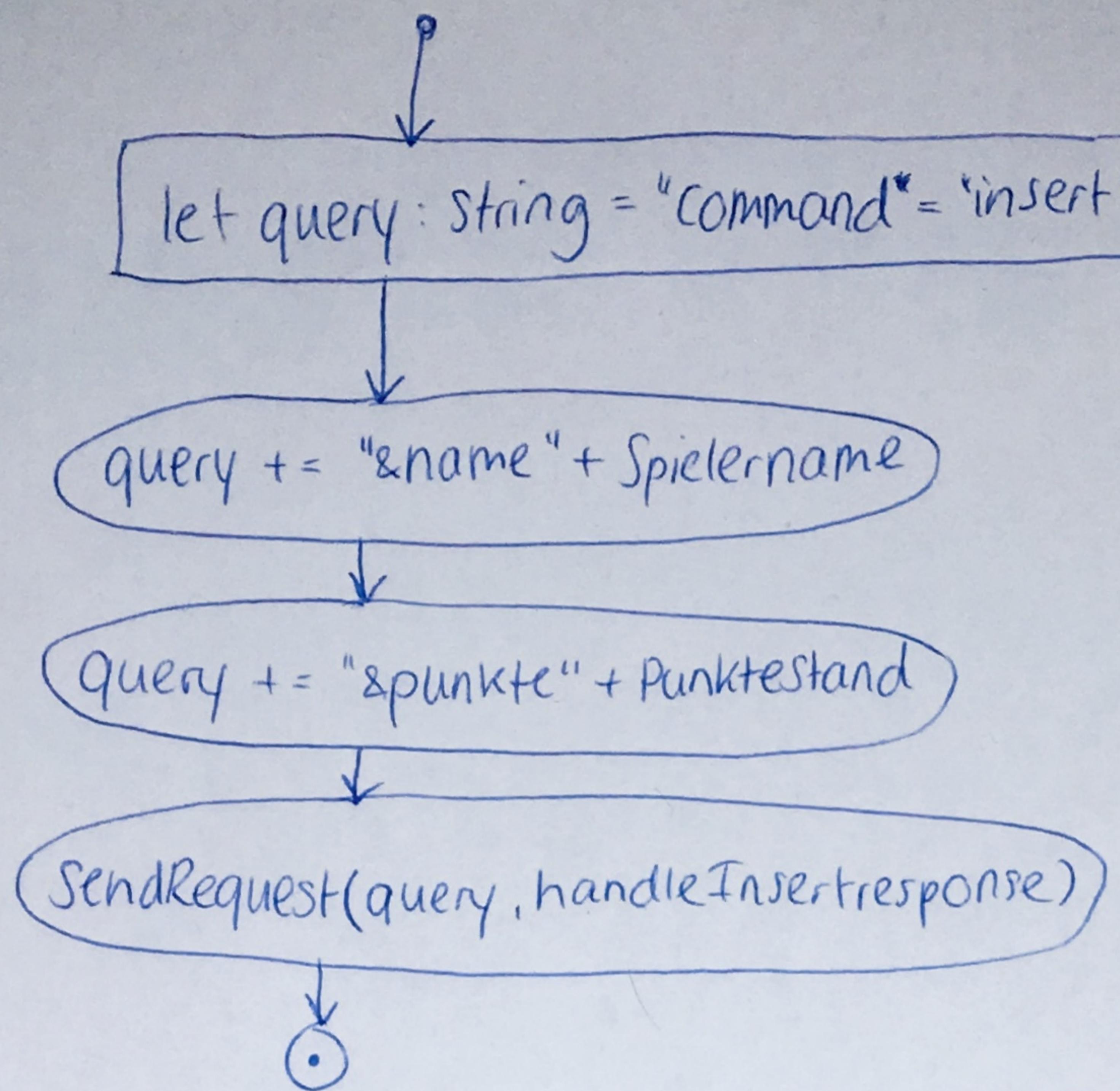


Blase klein move()

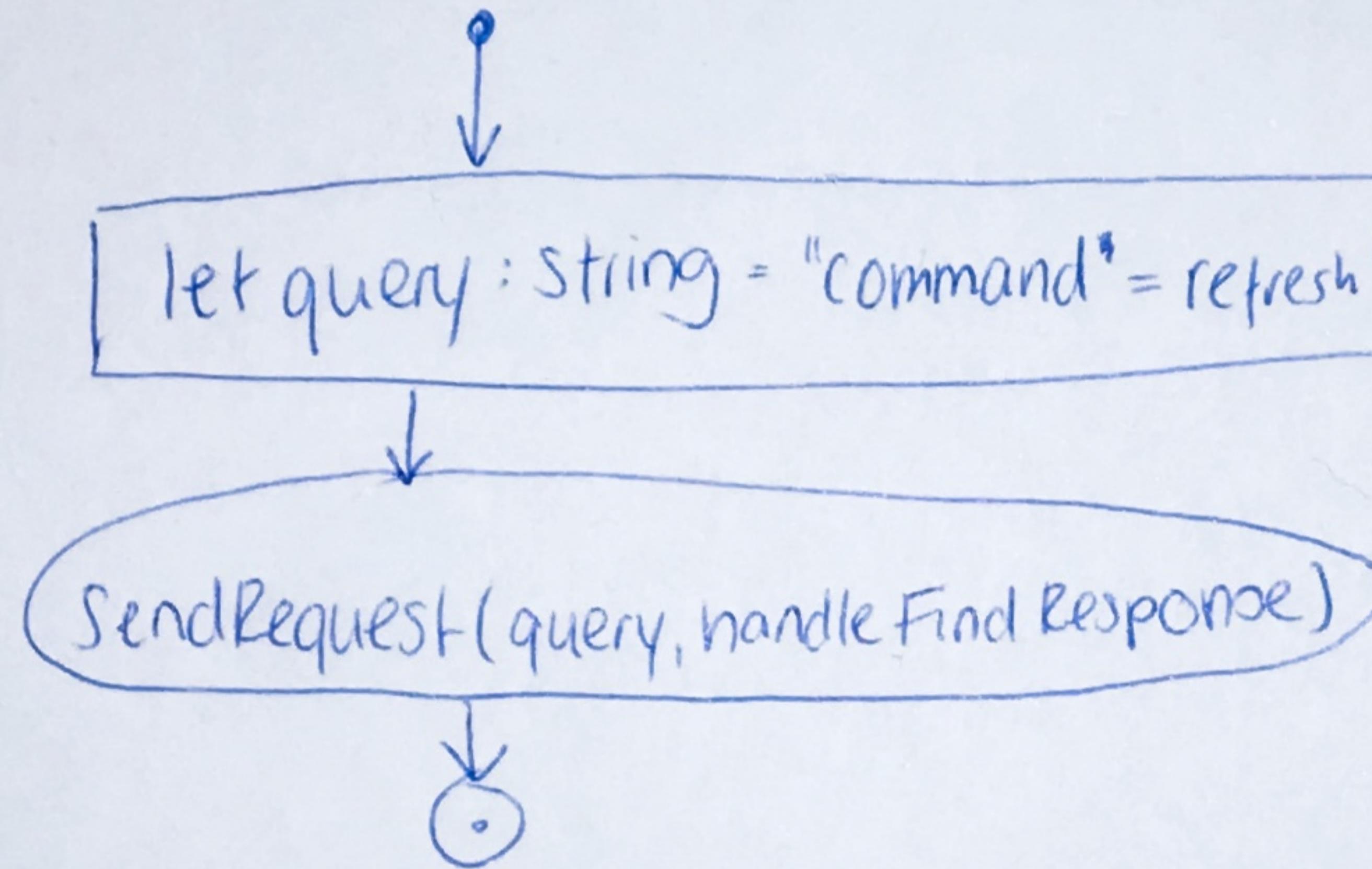




DB Client insert



DB Client refresh



Send Request

```
-query: string, -callback: EventListener
```

```
let xhr : XMLHttpRequest =  
    new XMLHttpRequest()
```

```
xhr.open("GET", serveraddress + "?" + -query, true)
```

```
xhr.addEventListener("readystatechange", -callback)
```

```
xhr.send()
```



handleInsertResponse


```
_event: ProgressEvent
```

```
let xhr: XMLHttpRequest =  
(<XMLHttpRequest>_event.target)
```

```
[xhr.readyState == XMLHttpRequest.DONE]
```

```
Alert(xhr.response)
```





Rangliste

_1: Spieler, _2: Spieler: number

[-1.punkte < -2.punkte]

[_1.punkte > -2.punkte]

1

0

-1

< interface >

Punkteliste

[key:string]:string;

< interface >

Spieler

name: string;
punkte: number;

types.ts