

Gemüsegarten-Simulator 27.06.2022

Desktop (+ Smartphone)

min. 40 Felder

Samen → Pflanze: Zustand + Zahlen

Schädlinge manchmal Pflanze → zerstören (mehrere)

Pflanzen, düngen, gießen, ernten, bekämpfen durch Interaktion

R ✓ mind. 5 Gemüse (sich unterschließlich anpassen, Wasser, Dünger)

+V Marktpreise für Gemüse, Samen, Dünger, Pestizide

Kapital Pflege der Gärten

1/2+ Vor Start Kapital + Preis schwankung einstellen

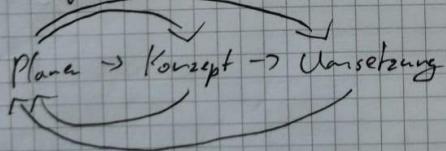
leeres Feld → Siedlung

+V/P überdüngt übergossen

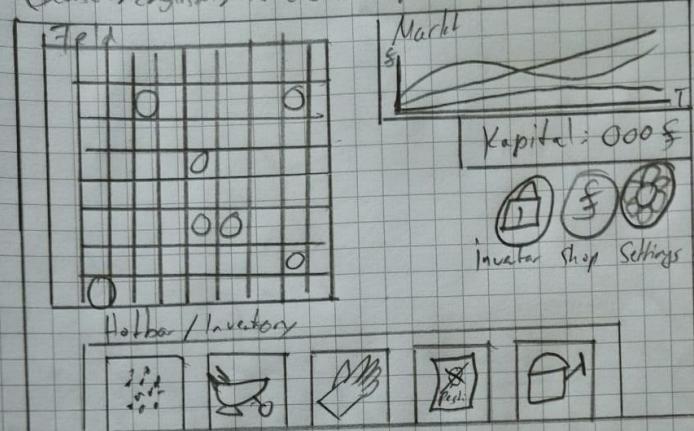
Tot → leer Erde → leer

Pestizide Punkte

Schädlinge realzeit rekt speed cyc



UI Scribble: Gemüse Garten-Simulator 27.06.2022
(Deutsch/English Mix als Planung für das Konzept)



Pillow Plant

Teddy Plant

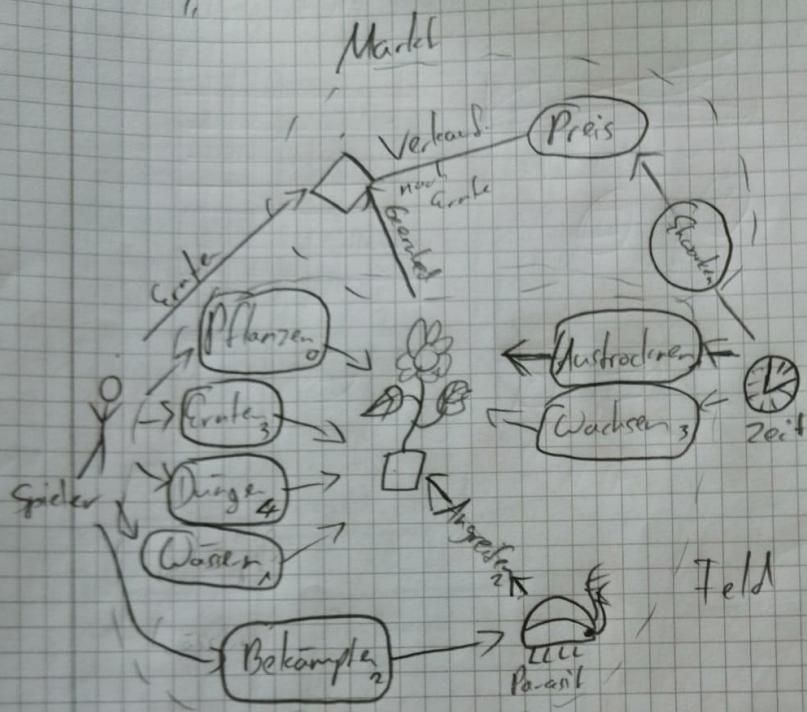
Plaeked Plant

Sock Plant

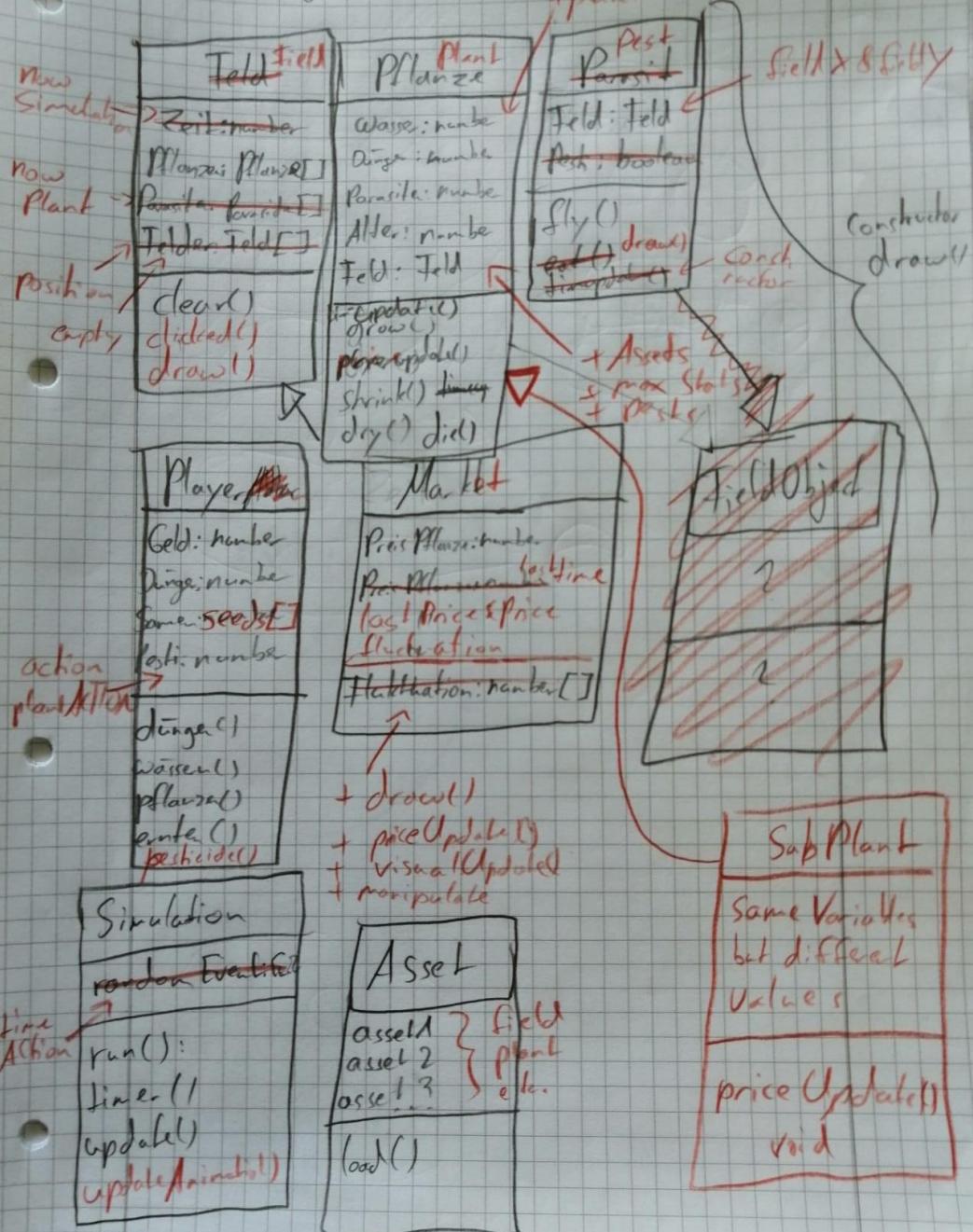
Scarf Plant

Enemies: Moths

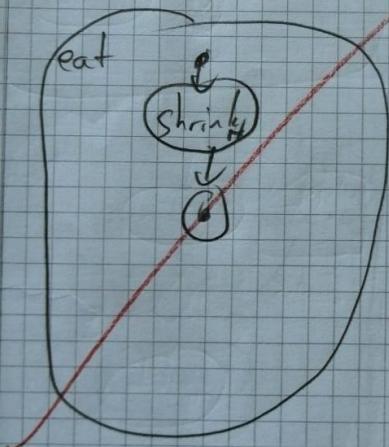
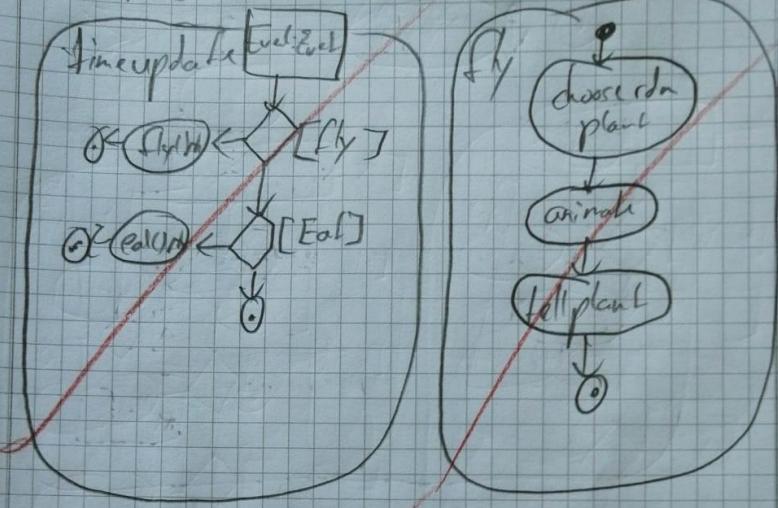
USE-Case: GG-Simulator 27.06.2012



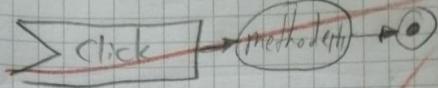
Class Diagramm: GG-Sim 27.06.2022



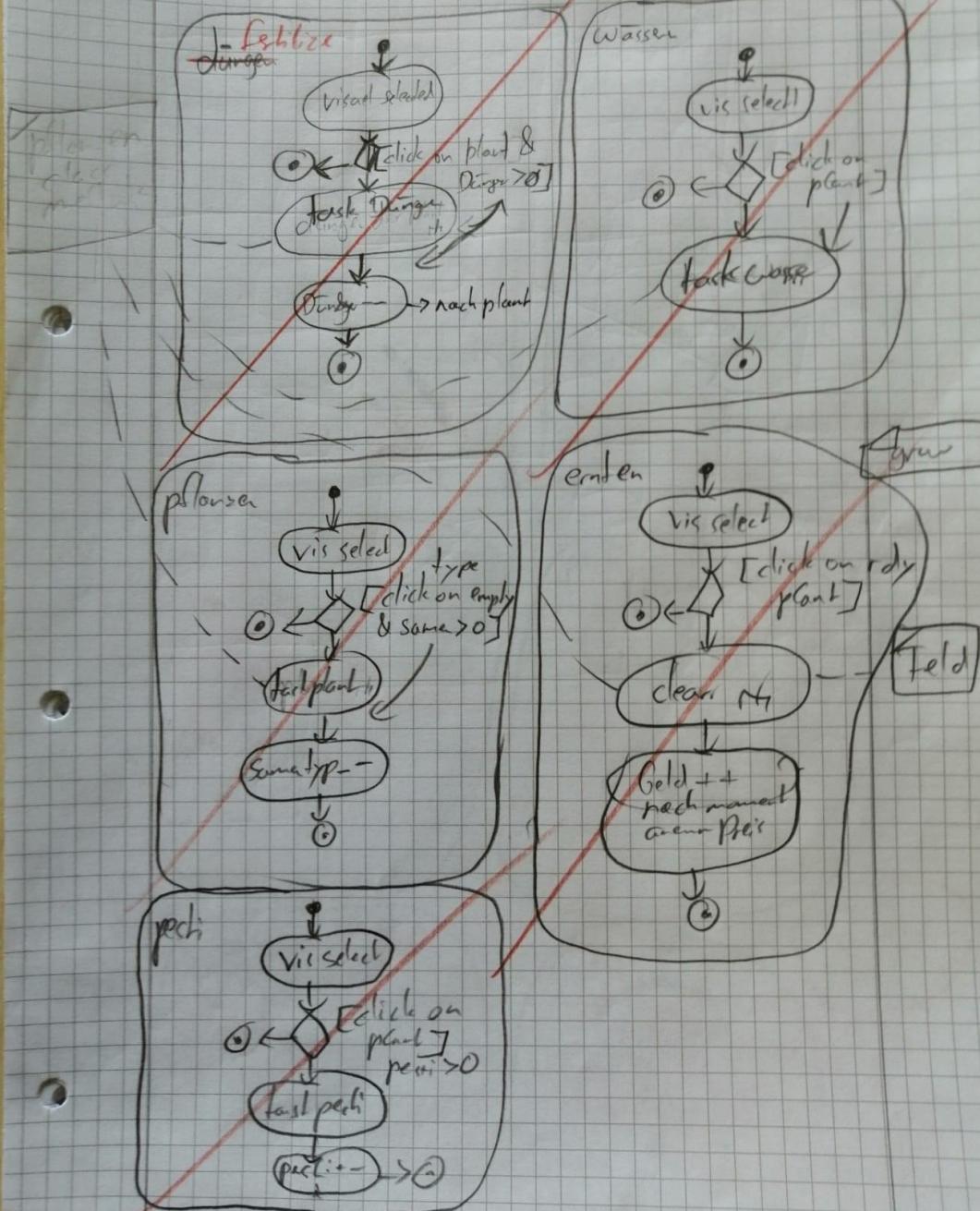
AD/GG-Sim / Paracit



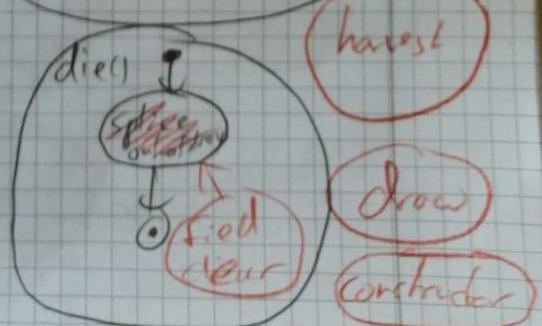
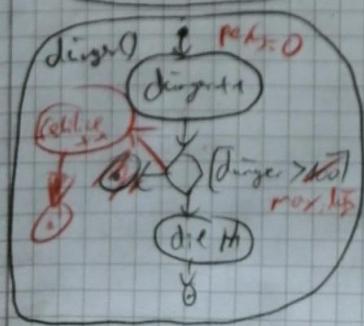
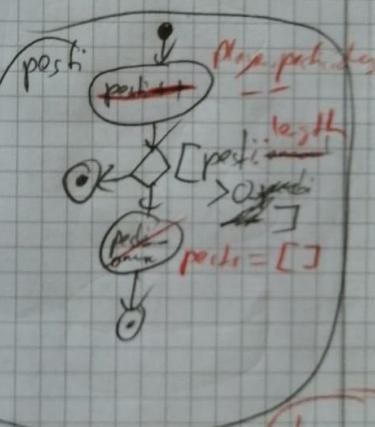
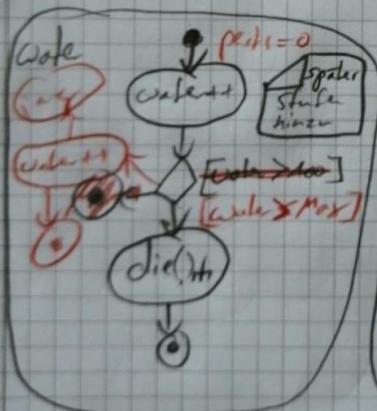
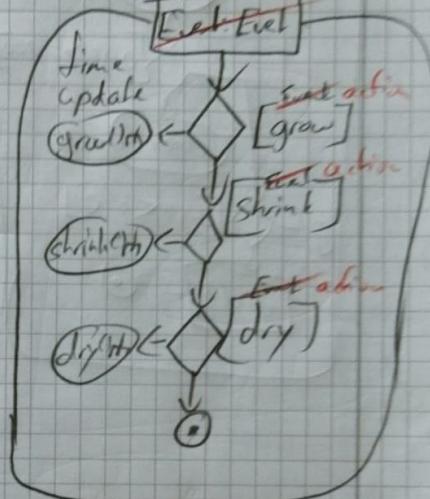
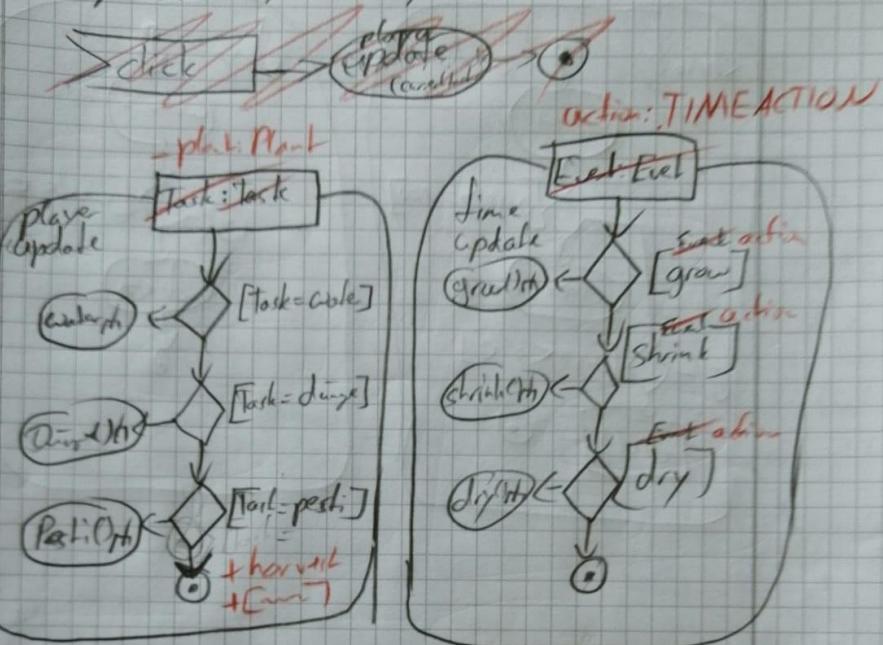
AD / GG-Sim / ~~Hobby~~ / Player

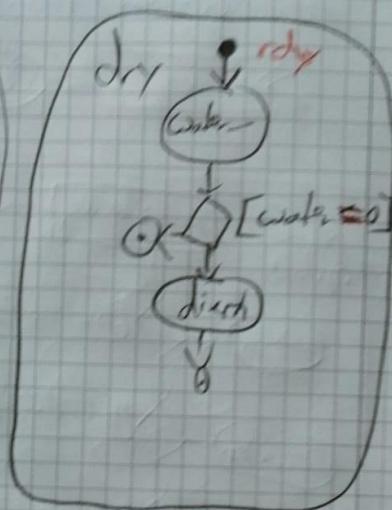
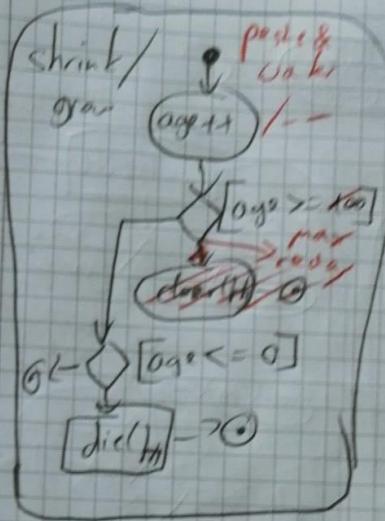


ACTION
PLANTACTION
interface Seed



AD/ GG-Sim / Plant /

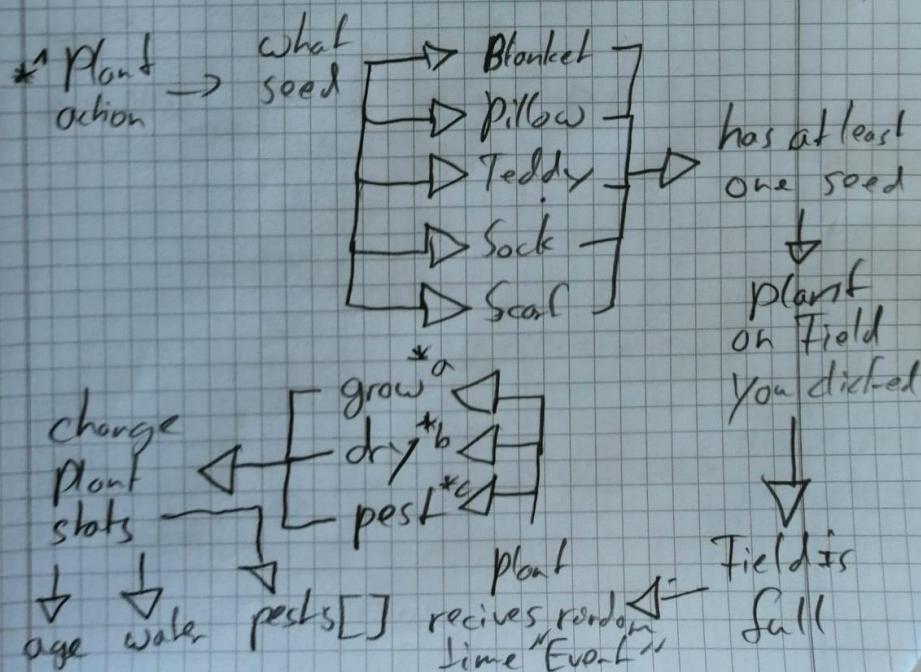
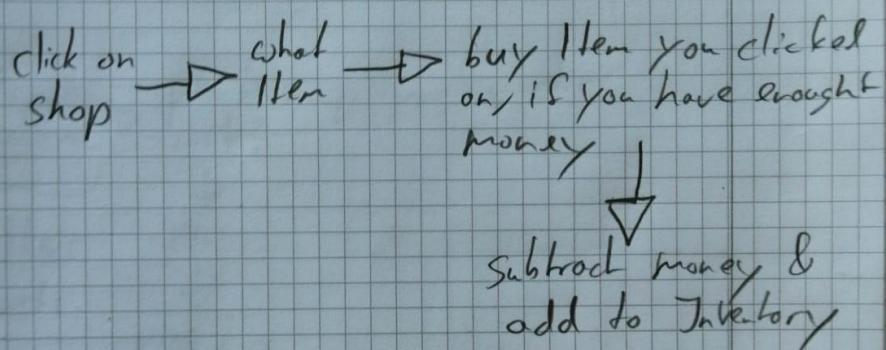
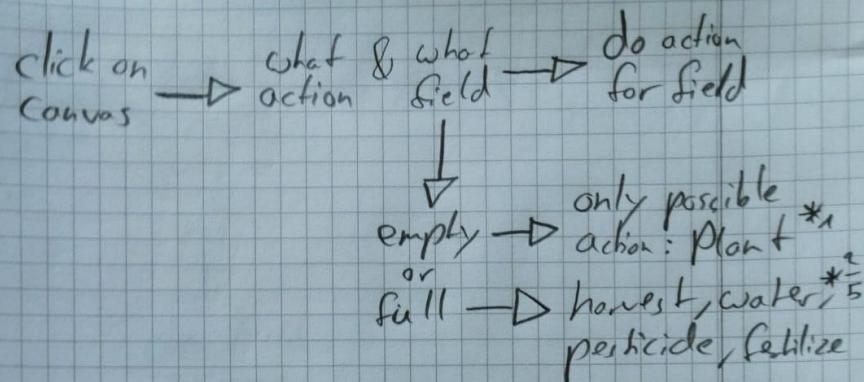




Ende erstes
Konzept

Beginn letztes
Konzept

Logic Flowchart 1



Logic Flowchart 2

*² harvest action → plant is ready? → yes → clear field
money ↑

↓
no

↓
clear field
no money

*³ water action → plant is wet? → yes → die

Only if there are no pests

↓
rise waterlevel

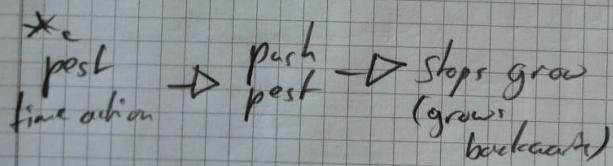
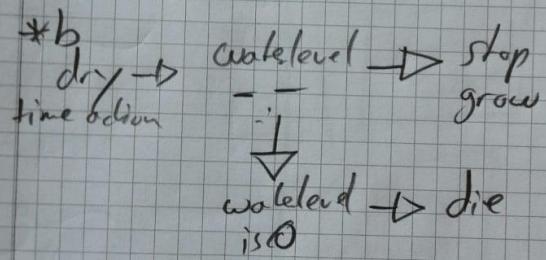
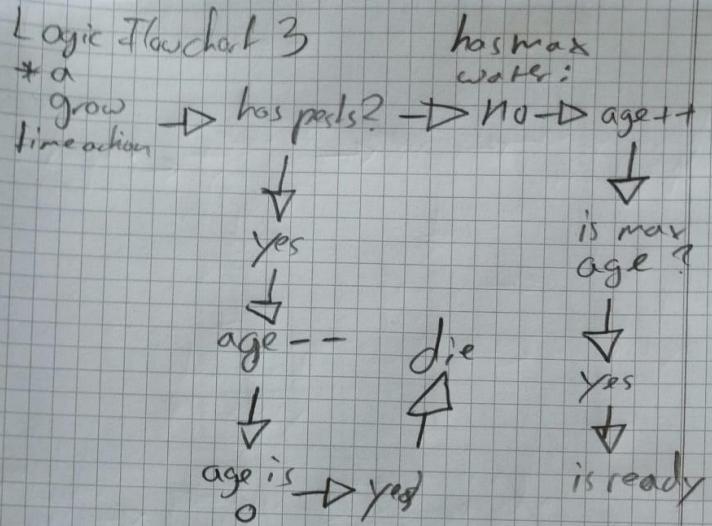
*⁴ fertilize action → plant is fertilized? → yes → die

↓
no

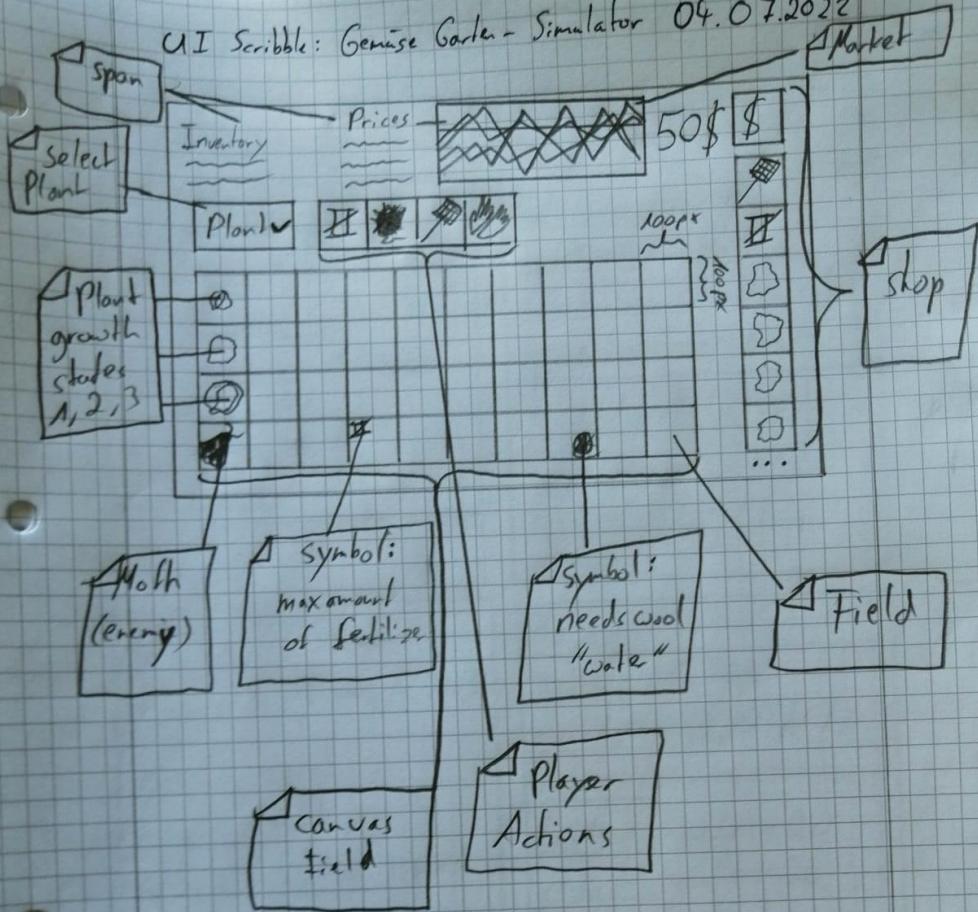
↓
rise fertilized level & fertilize --

*⁵ pesticide action → plant has pests? → yes → clear pests
pesticides --

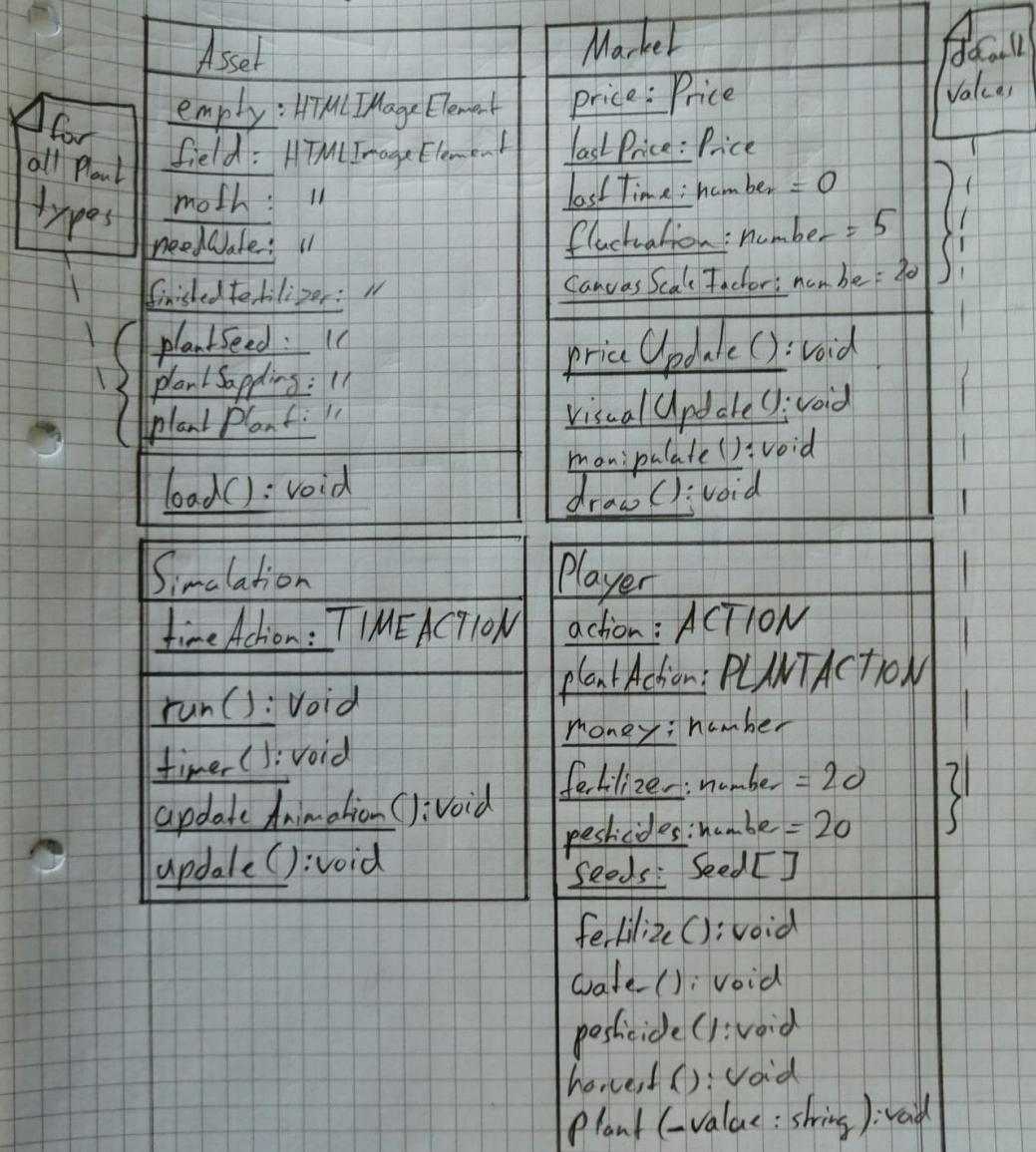
↓
no



UI Scribble: Gemüse Garten - Simulator 04.07.2022



Class diagram: GG-Sim 04.07.2022



Class diagram: GG-Sim 04.07.2022

Field

size: number = 100

positionX: number

positionY: number

isEmpty: boolean = true

plant: Plant

constructor (- positionX: number, - positionY: number): void

clicked (- x: number, - y: number): void

clear (- plant: Plant): void

draw (): void

Pest

fieldX: number

fieldY: number

constructor (- fieldX: number, - fieldY: number): void

fly (): void

draw (- fieldX: number, - fieldY: number): void

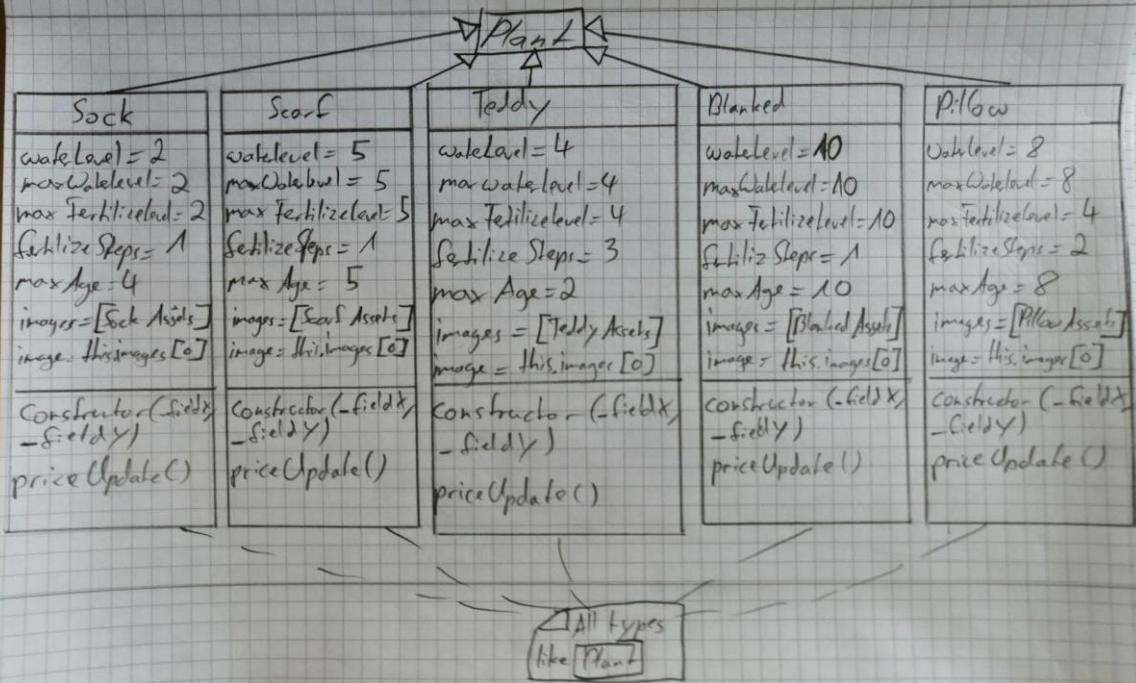
Class diagram: 66-Sim 04.07.2022

Plant

fieldX: number
fieldY: number
waterLevel: ""
maxWaterLevel: ""
fertilizeLevel: "" = 0
maxFertilizeLevel: ""
age: "" = 0
maxAge: ""
priceValue: "" = 1
isReady: boolean = false
pests: Pest[] = []
images: HTMLImageElement[]
image: HTMLImageElement
statusLevelImages: HTMLImageElement[] = [Asset.empty, Asset.woodWater]
statusLevelImageWater: HTMLImageElement = Asset.finishedFertilizer
statusLevelImageFertilizer: HTMLImageElement = Asset.finishedFertilizer
plant: Plant

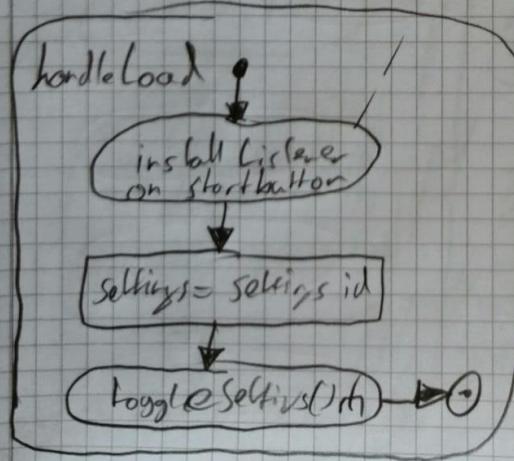
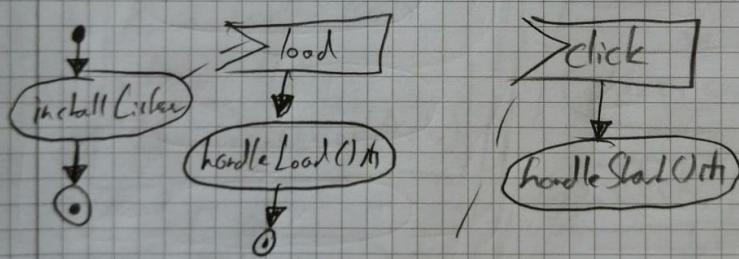
constructor (-fieldX: number, -fieldY: number): void
priceUpdate(): void
timeUpdate(-action: TIMEACTION): void
playerUpdate(-plant1: Plant): void
getWatered(): void
getFertilized(): void
getPesticed(): void
getHarvested(): void
grow(): void die(): void
shrink(): void draw(): void
dry(): void

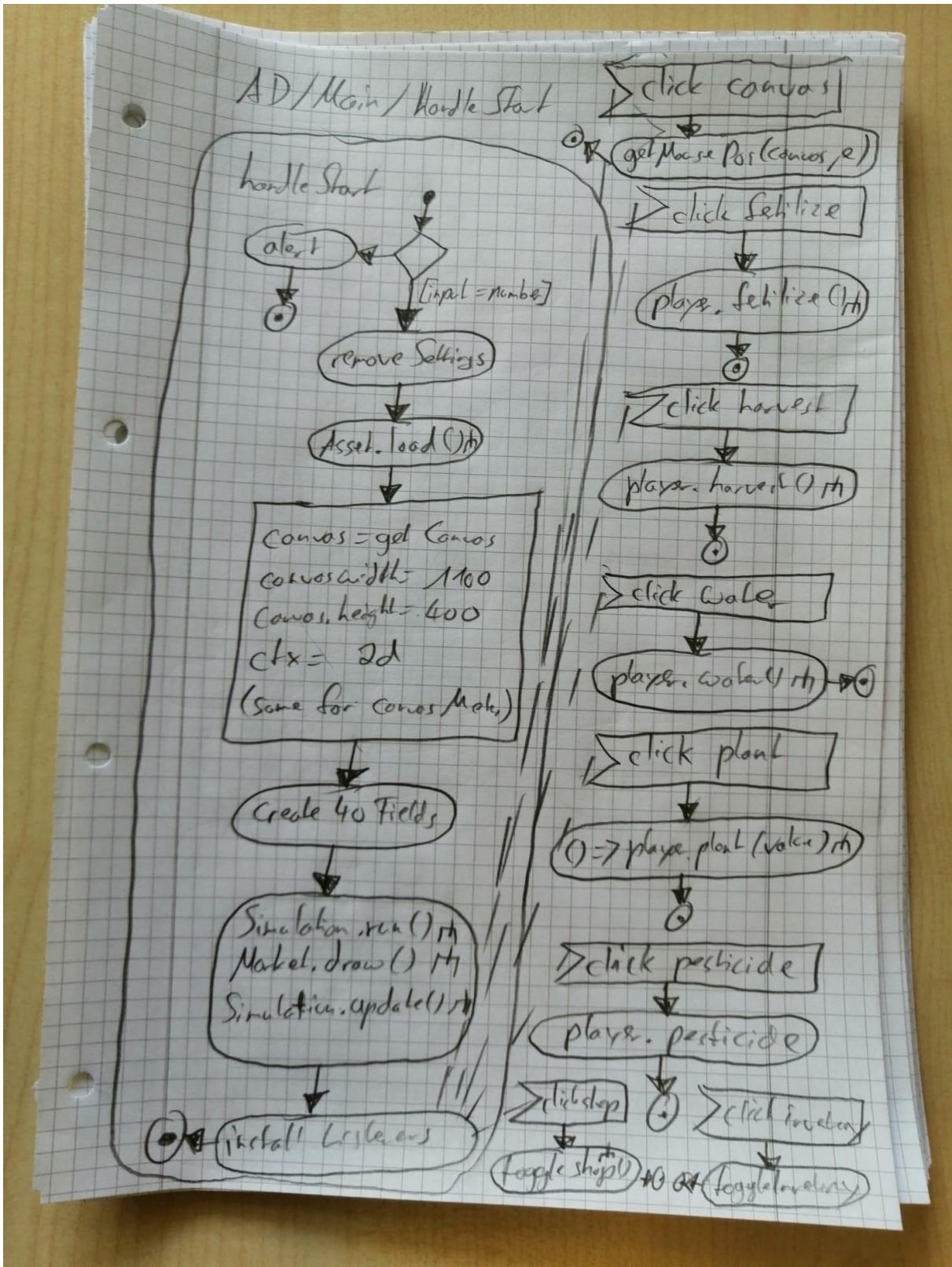
Class diagram GG-Sim 04.07.2022



AD/Main/Variables & Listener for WillLoad

```
export: canvas: HTMLCanvasElement  
canvasM: ""  
ctxX: CanvasRenderingContext2D  
ctxM: ""  
mouseX: number  
mouseY: number  
player: Player = new Player  
market: Market = new Market  
plants: Plant[] = []  
fields: Field[] = []  
Shop: HTMLElement  
Settings: ""
```





AD/Main/1

Canvas: HTMLElement, Element
getMousePos() -> evt: MouseEvent

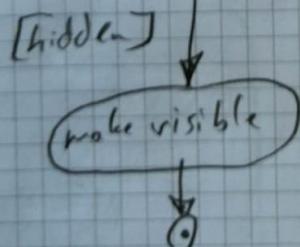
rect: DOMRect =
canvas.getBoundingClientRect()

mouseX = evt.clientX - rect.left
mouseY = evt.clientY - rect.top

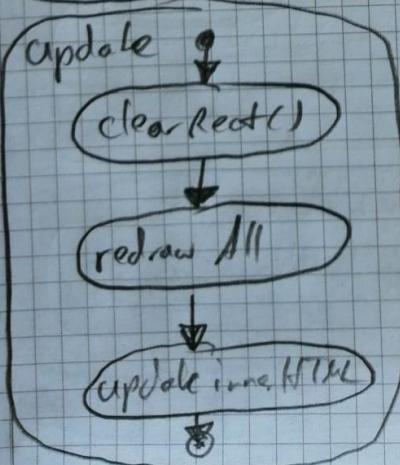
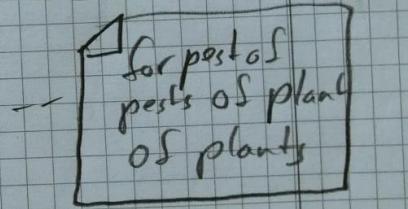
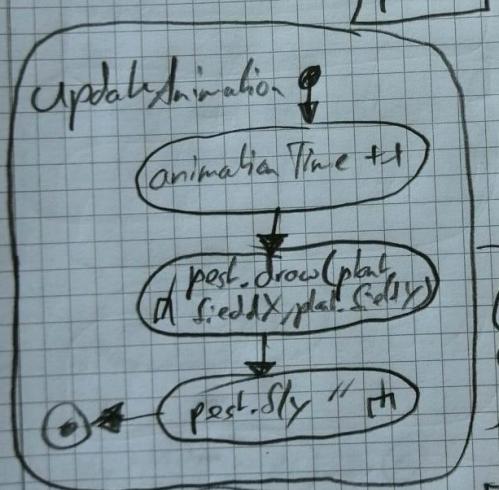
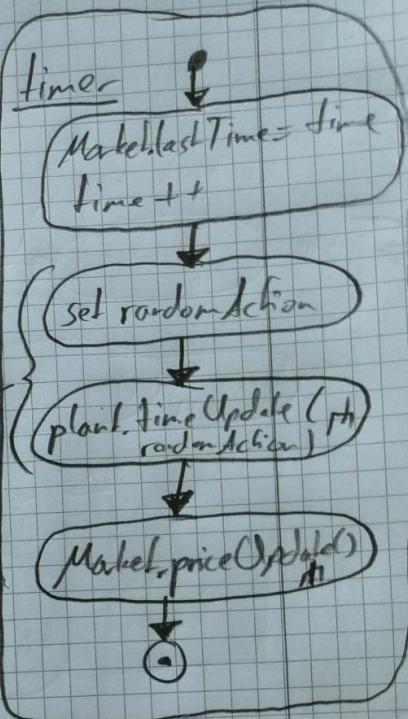
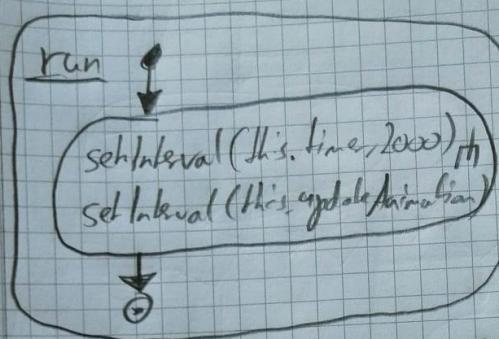
for
size of
fields

{ field.clicked(mouseX, mouseY) }

toggle Settings
toggle Inventory
toggle Shop



AD/Simulation



```

export timeNumber = 0
export animationTime = 0
export enum TIMEACTION
  GROW,
  DRY,
  REST
random Action: TIMEACTION
timeActions[] = []
  
```

AD/Field/1

Constructor [-positionX; -positionY]

• This.positionX = -positionX
This.positionY = -positionY

clear [-plantList: Plant]

This.plantList.removeAll()

plants.splice(plants.findIndex(
(e: plant) => e == plant), 1)

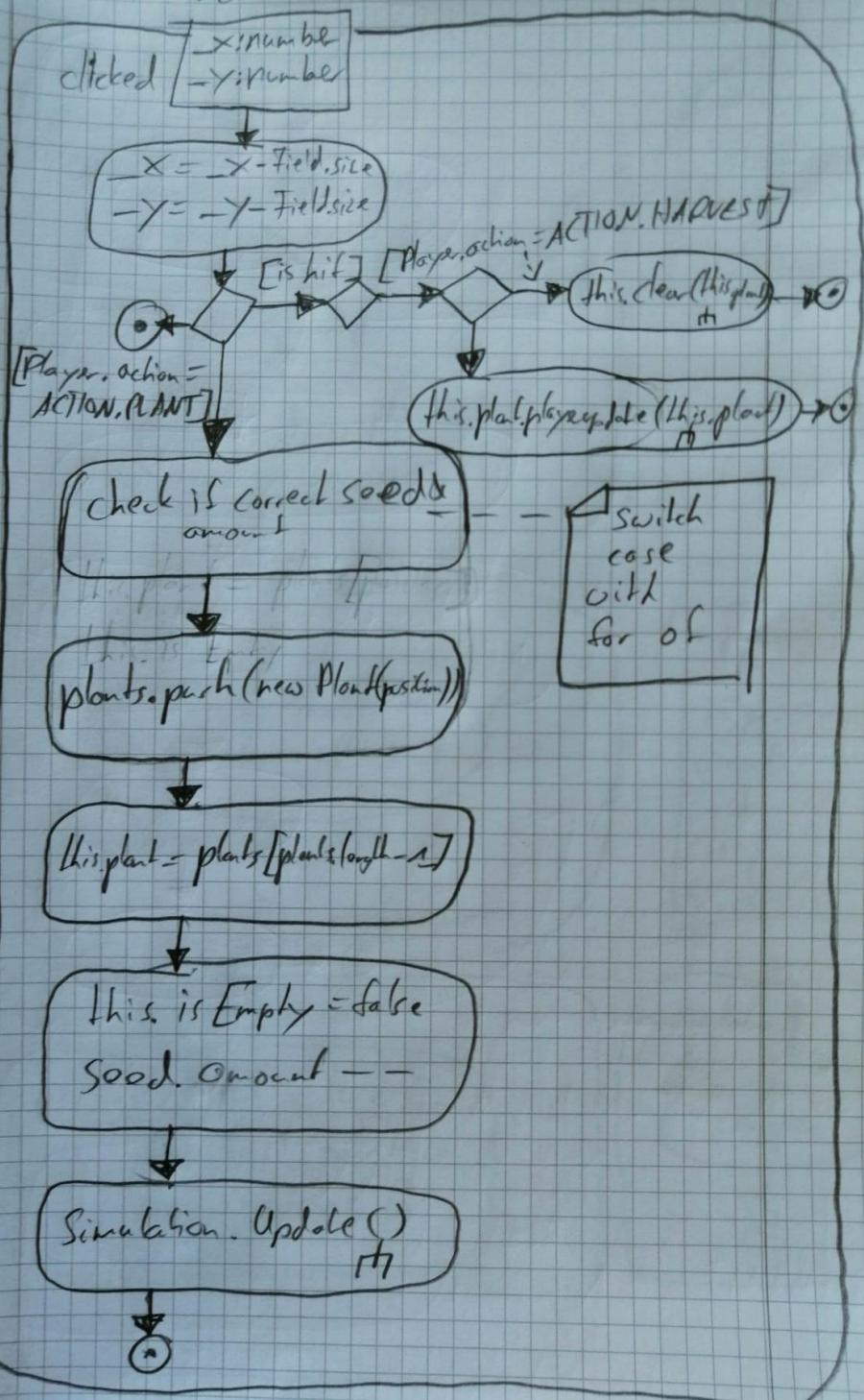
This.isEmpty = true

Simulation.
update() →

draw

translate(positionX / Y)
draw(Asset Field)

AD/Field/2



AD/Market

priceUpdate

Market.lastPrice = Market.price

Market.price = Random.Sim + X

Market.draw H

For every
Item you
can buy/sell

VisualUpdate

change innerHTML
to "Name": Price + \$

export interface
price

type: string

amount: number

Cost 11

Cost 11

Cost 11

Cost 11

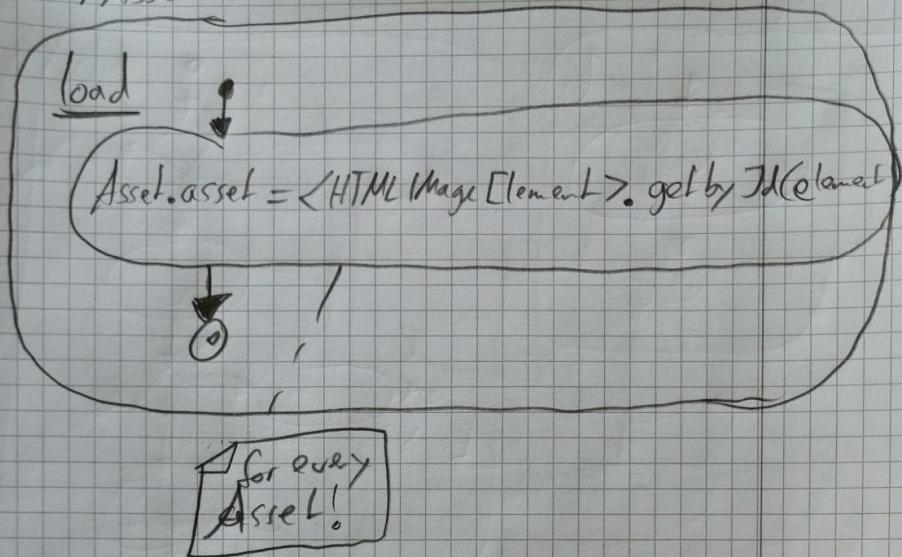
draw

draw a line from
the last point to
the next one by
time

manipulate

create
stepper

AD/Asset



AD/Pest

Constructor

-fieldXnumber
-fieldYnumber

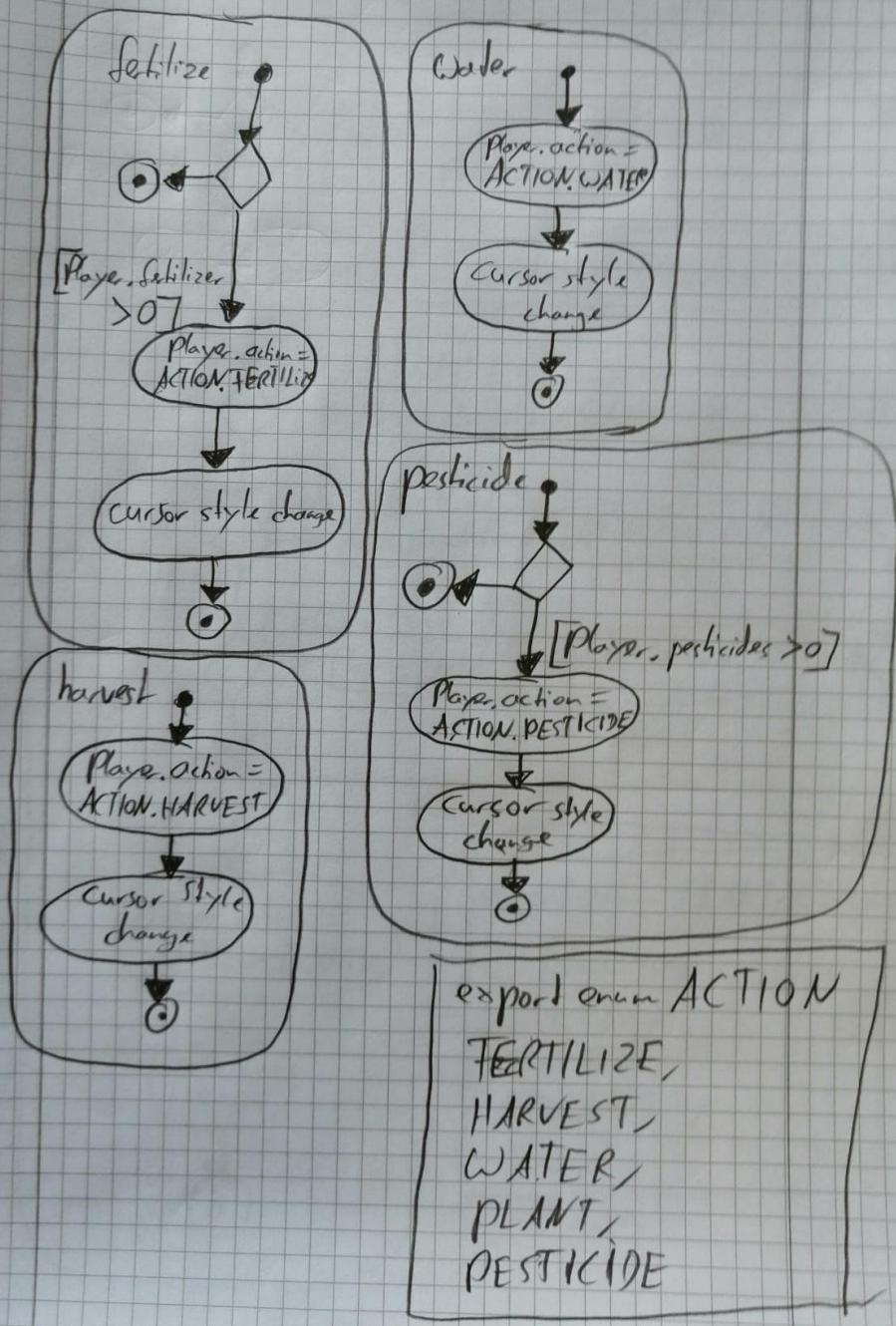
this.fieldX=-fieldX
this.fieldY=-fieldY

-fieldXnumber
-fieldYnumber

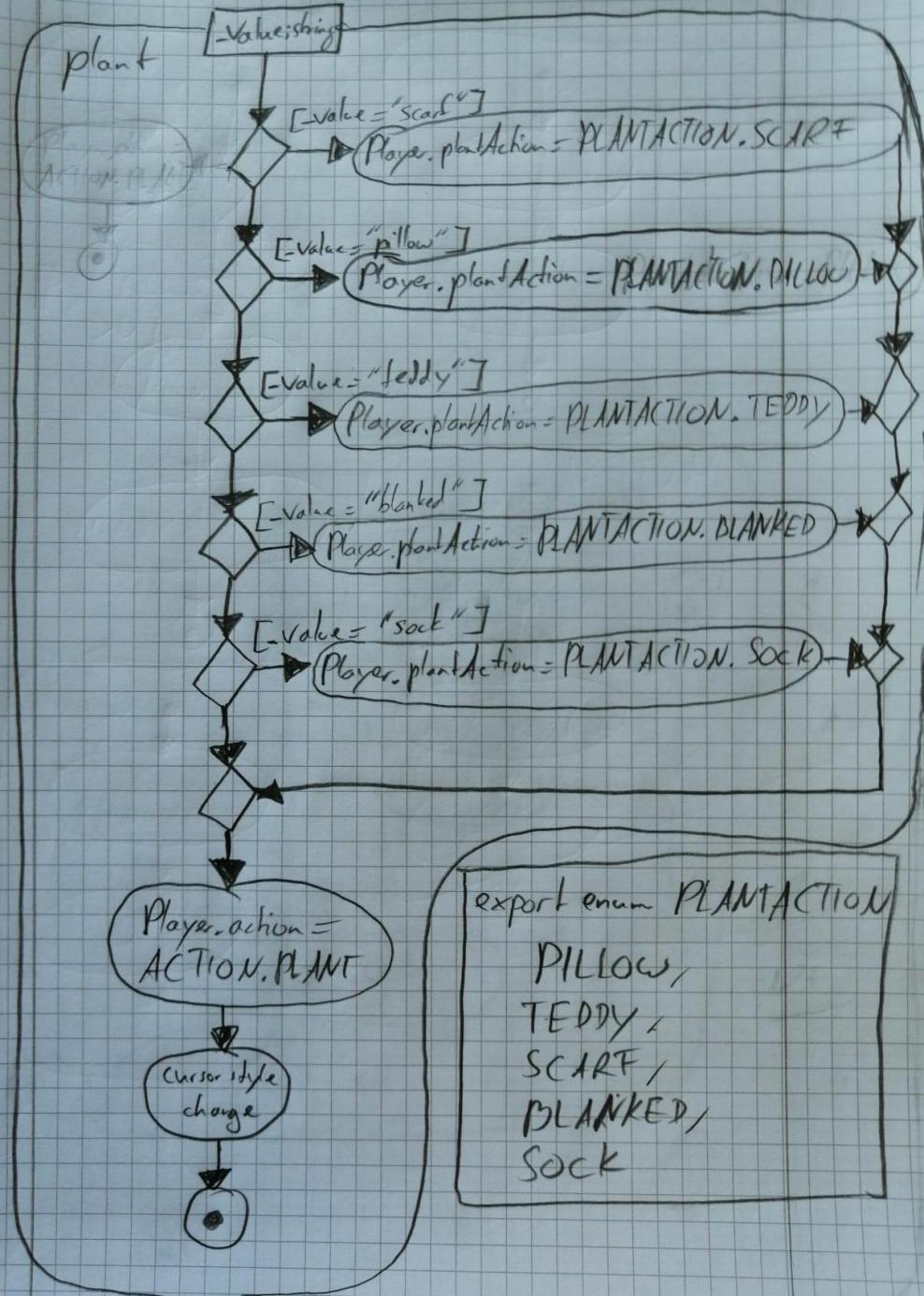
draw
drawImage(
MothAsset)
rotate(orientation)

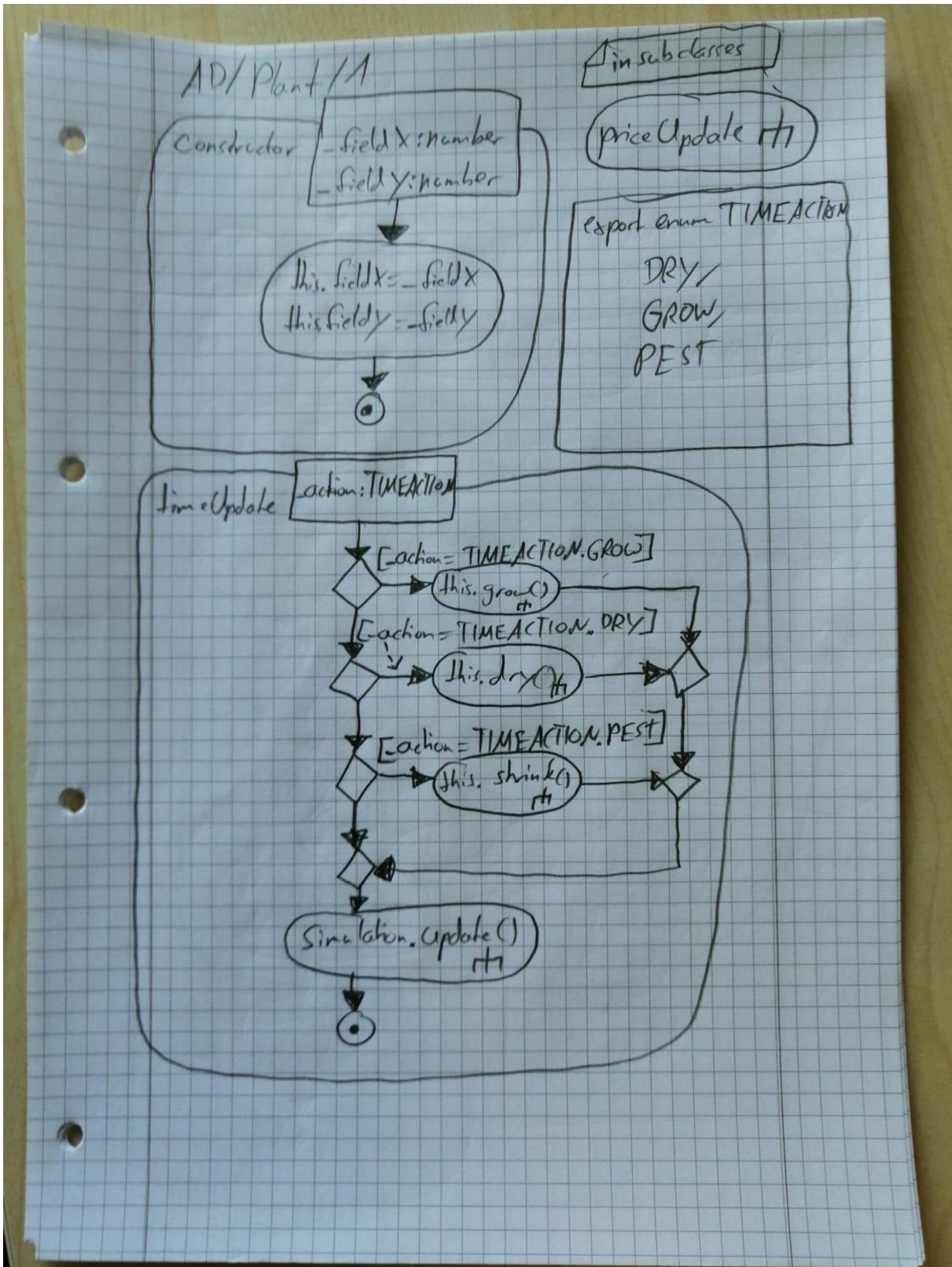
fly

AD/Player/1

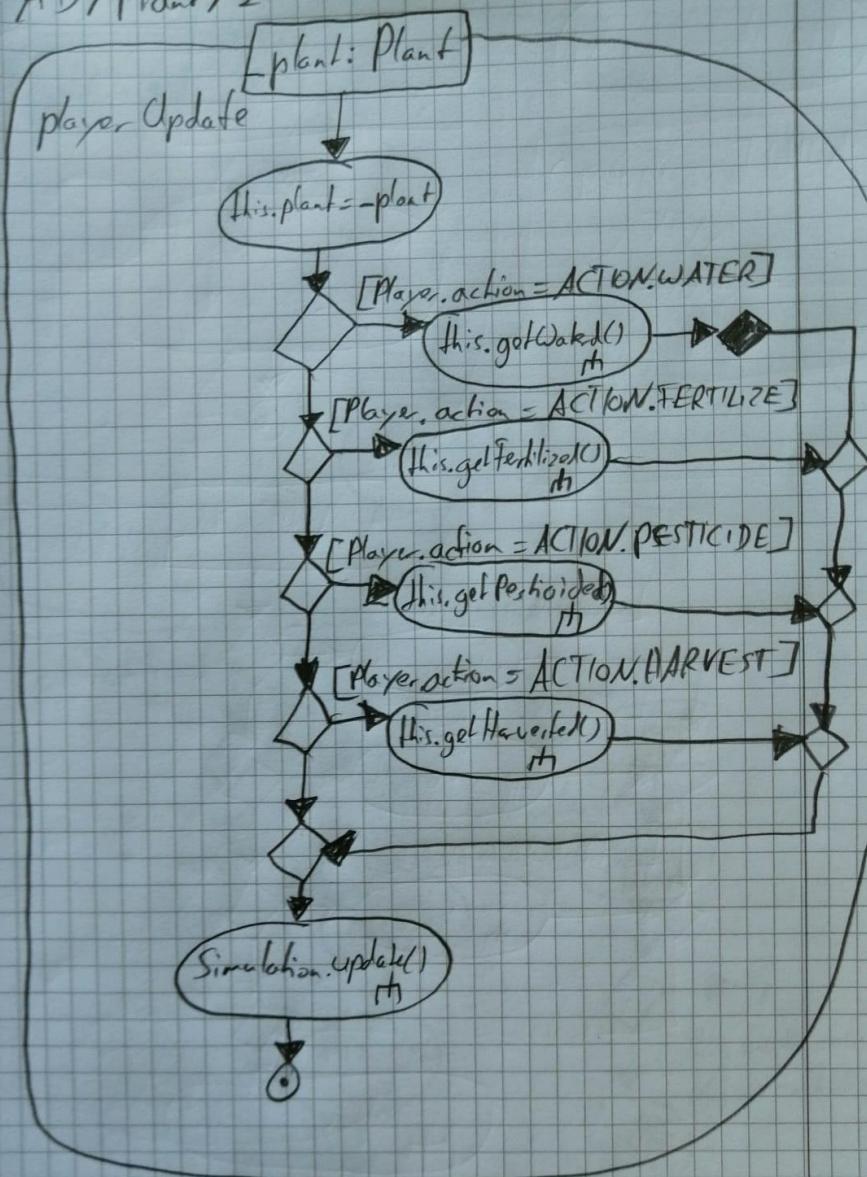


AD/Player/12

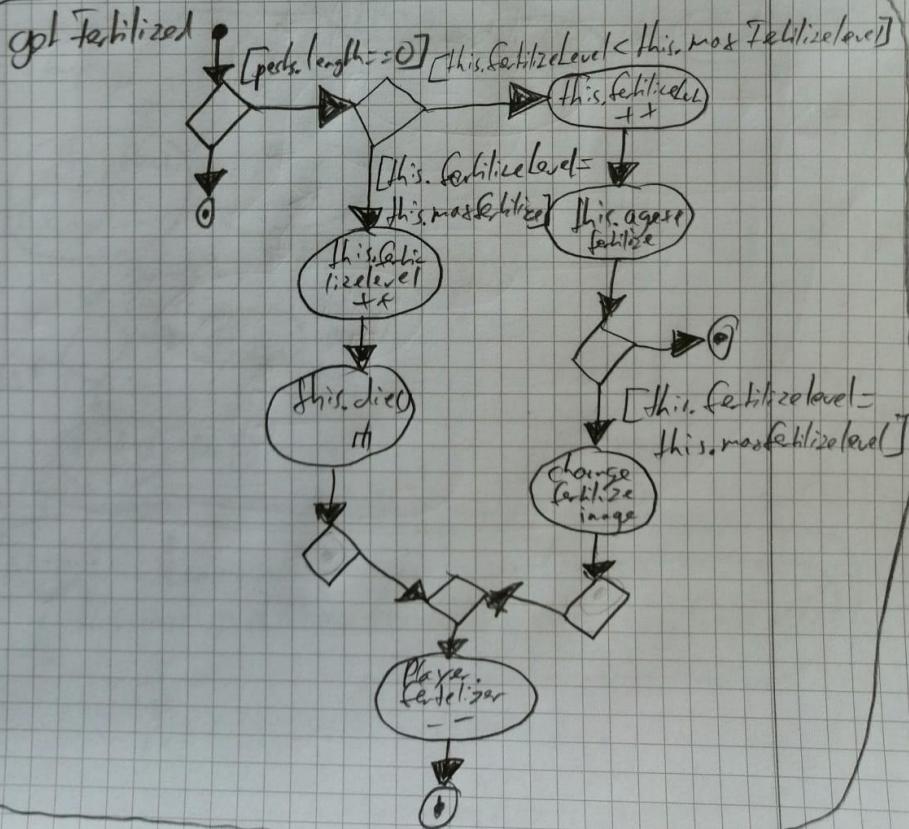
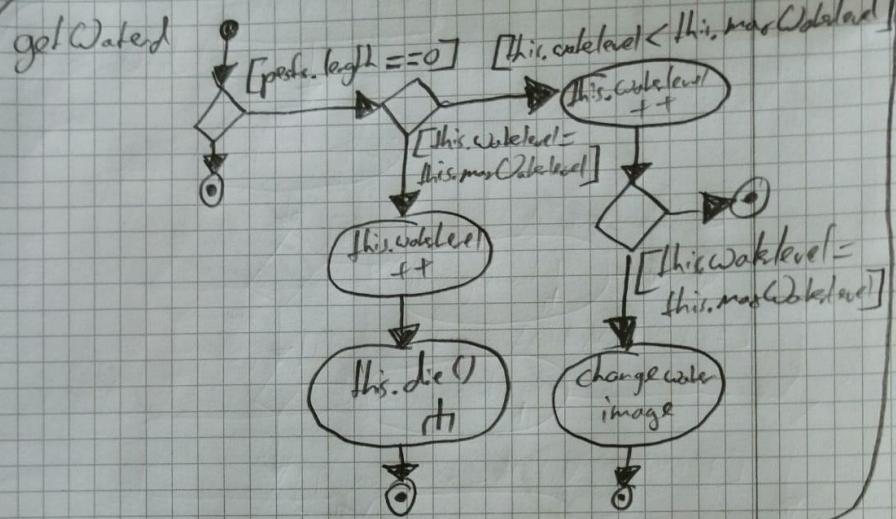




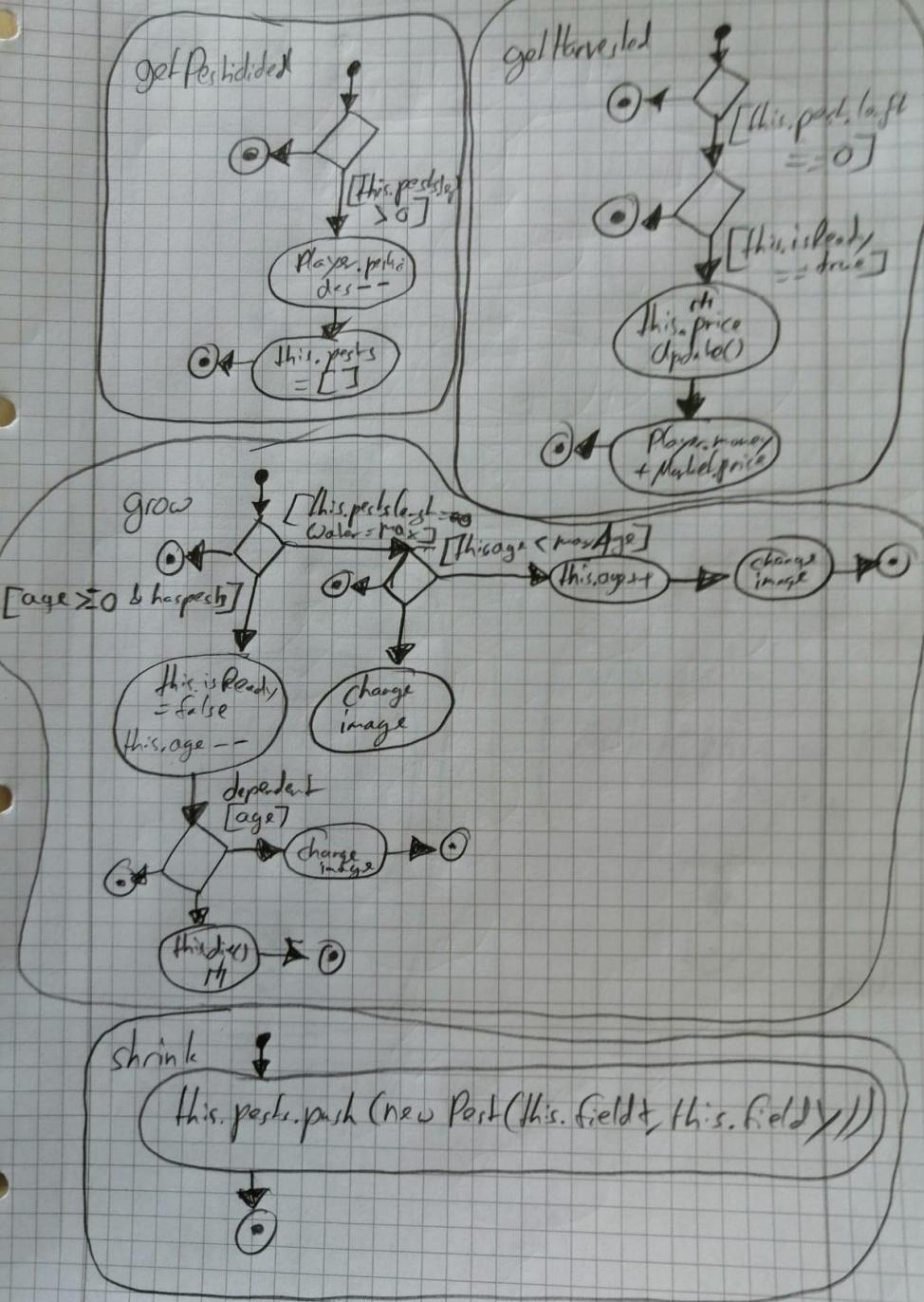
AD/Plant/2



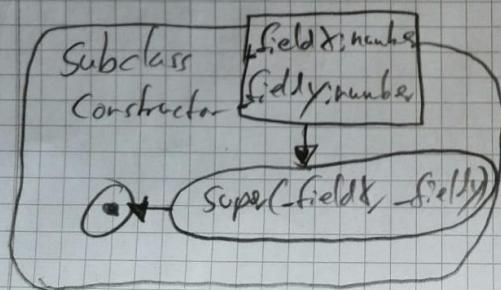
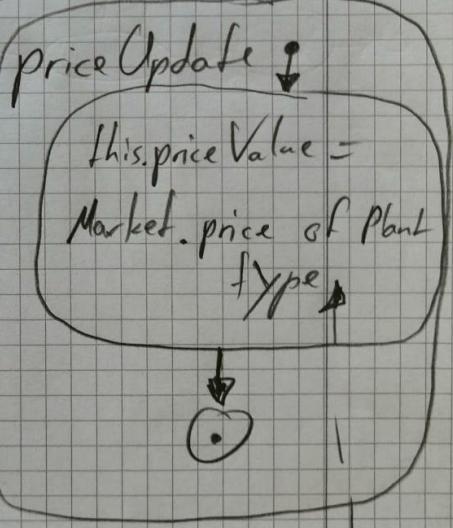
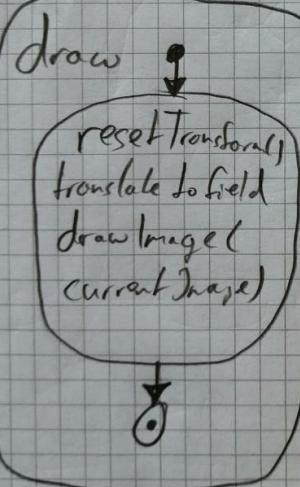
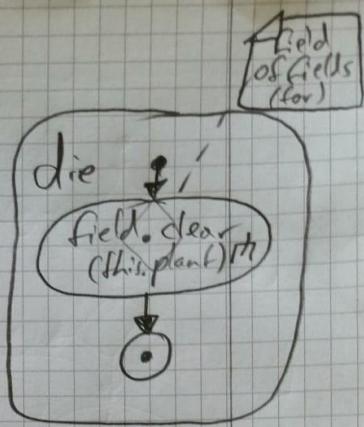
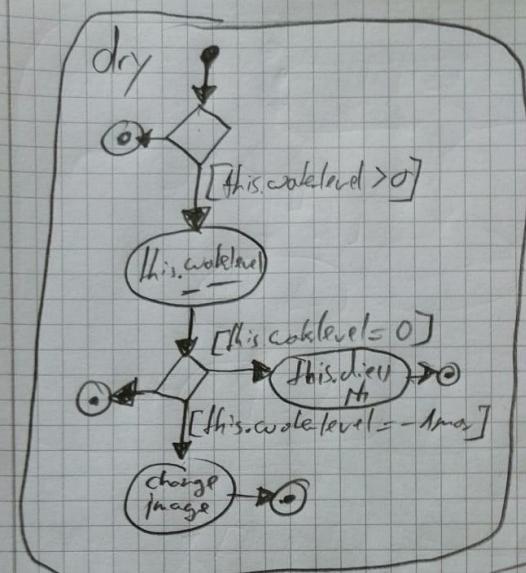
AD/Plant/3



AD/Plant 1/4



AD/Plant/5



Different Price
is in each
Sub class
defined