Gemüsegarten - Simulator 27.06.2022

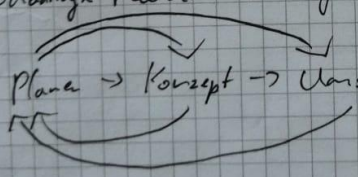
Desktop (+ Smartphone)
min. 40 Felder
Samen → Pflanze: Zustand + Zahlen
Schädlinge manchmal Pflanze → Zerstören (mehrere)
pflanzen, düngen, gießen, ernten, bekämpfen durch Interaktion
+✓ mind. 5 Gemüse (sehen unterschiedlich aus/Wachstum, Wasser, Dünger)
+✓ Marktpreise für Gemüse, Samen, Dünger, Pestizide
Kapital Pflege des Gartens

½+ Vor Start Kapital + Preisschwankung einstellen

leeres Feld → Setzling
+✓✓ überdüngt übergossen
Tot → leer Ernte → leer

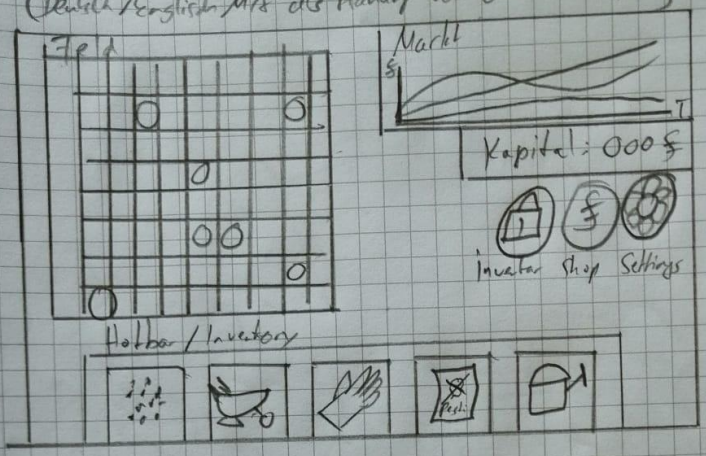
Pestizide Punktuell

Schädlinge realzeit real speed up



UI Scribble: Gemüse Garten-Simulator 22.06.2022

(Deutsch/English Mix als Planung für das Konzept)



Pillow Plant

Teddy Plant

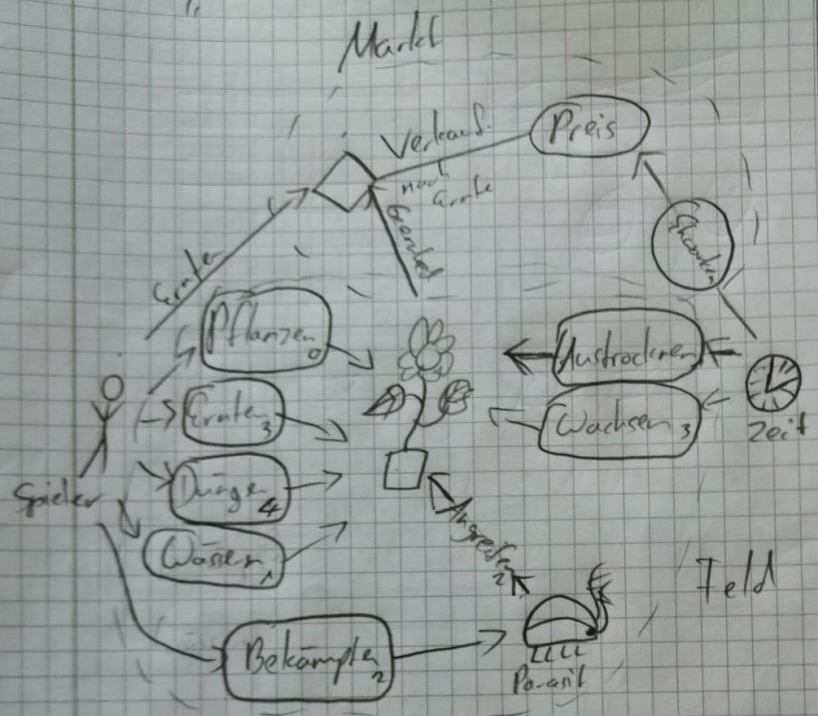
Bloated Plant

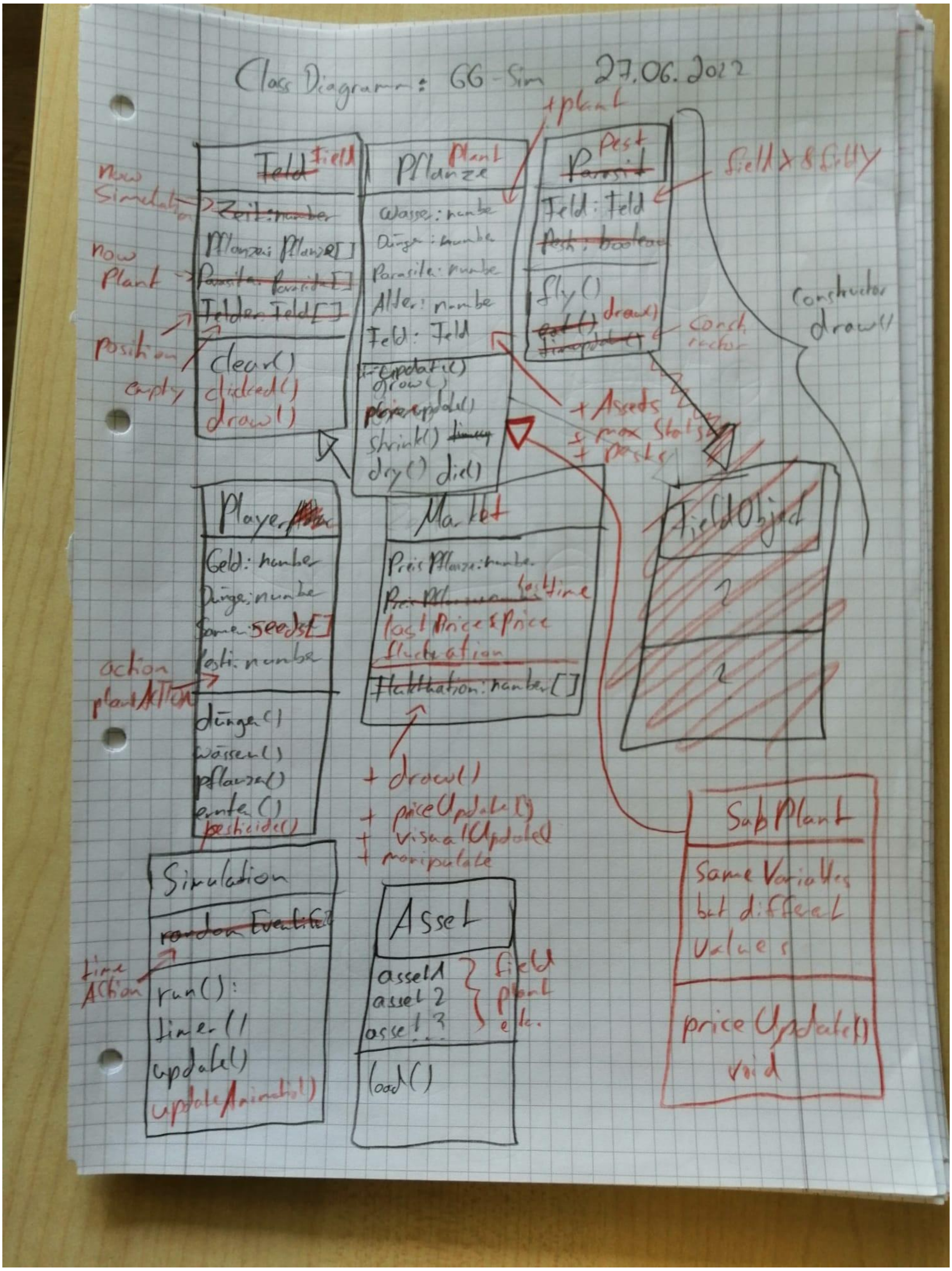
Sock Plant

Scarf Plant

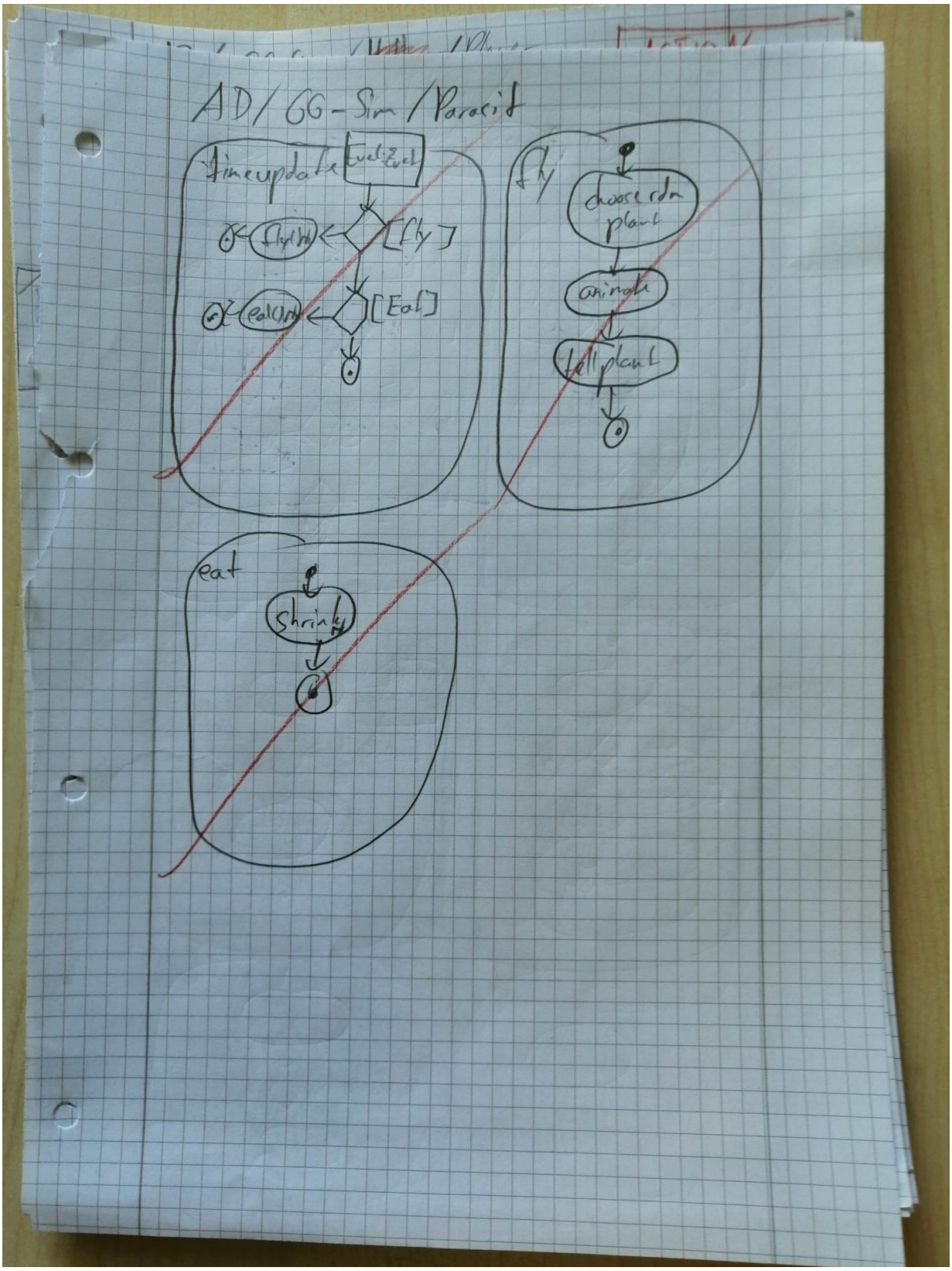
Enemies: Moths

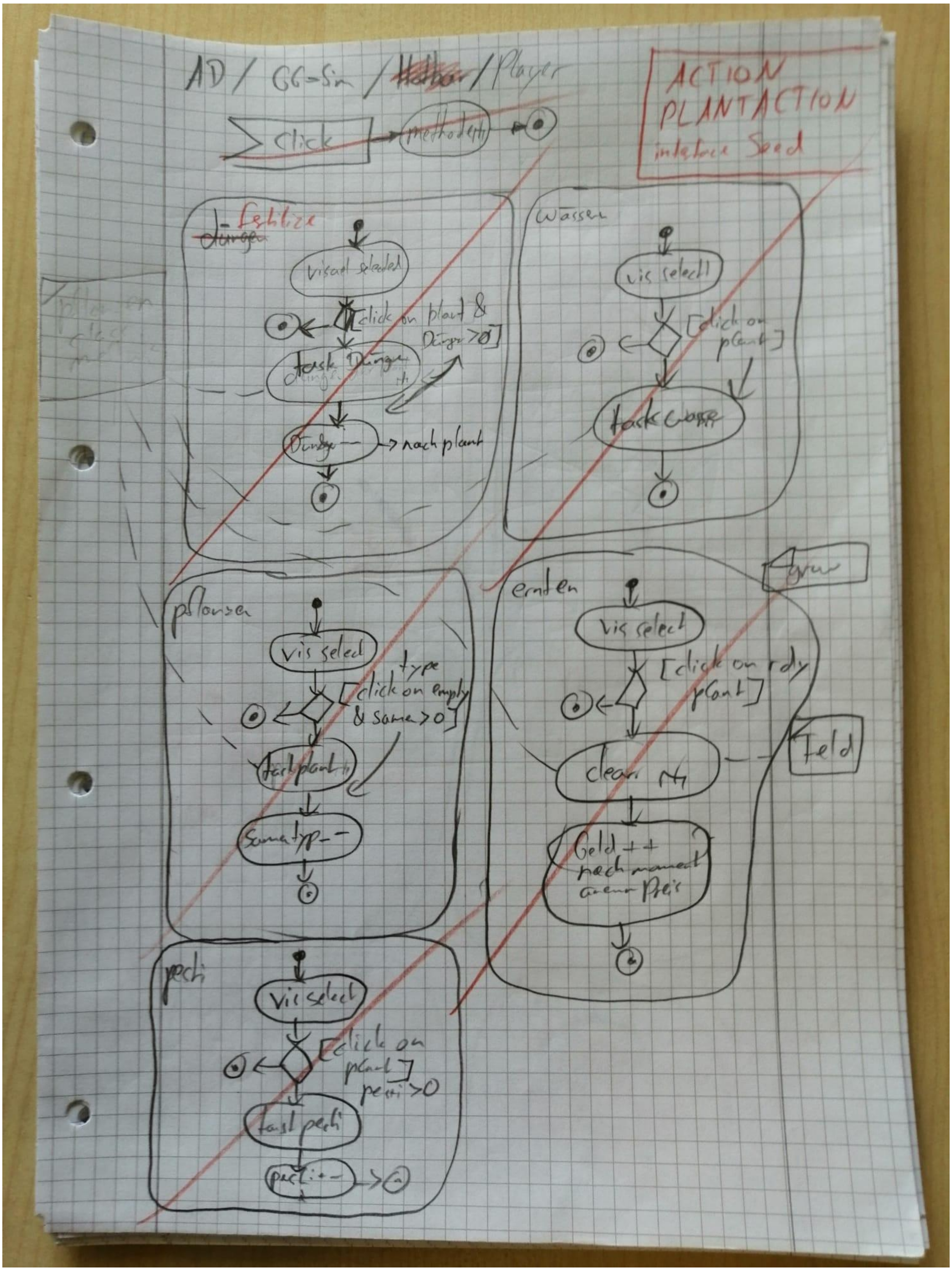
USE-Case: GG-Simulator 27.06.2022



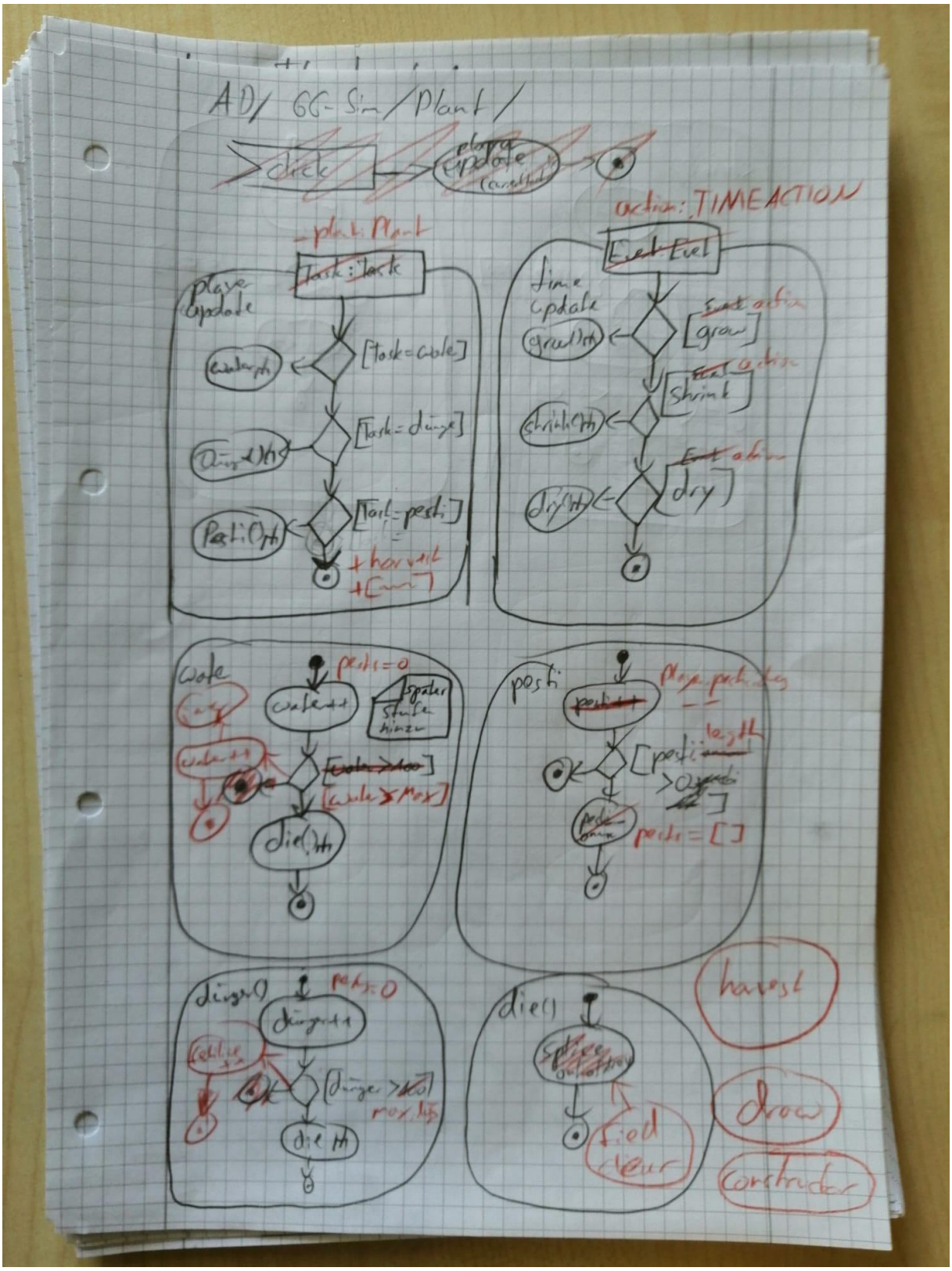


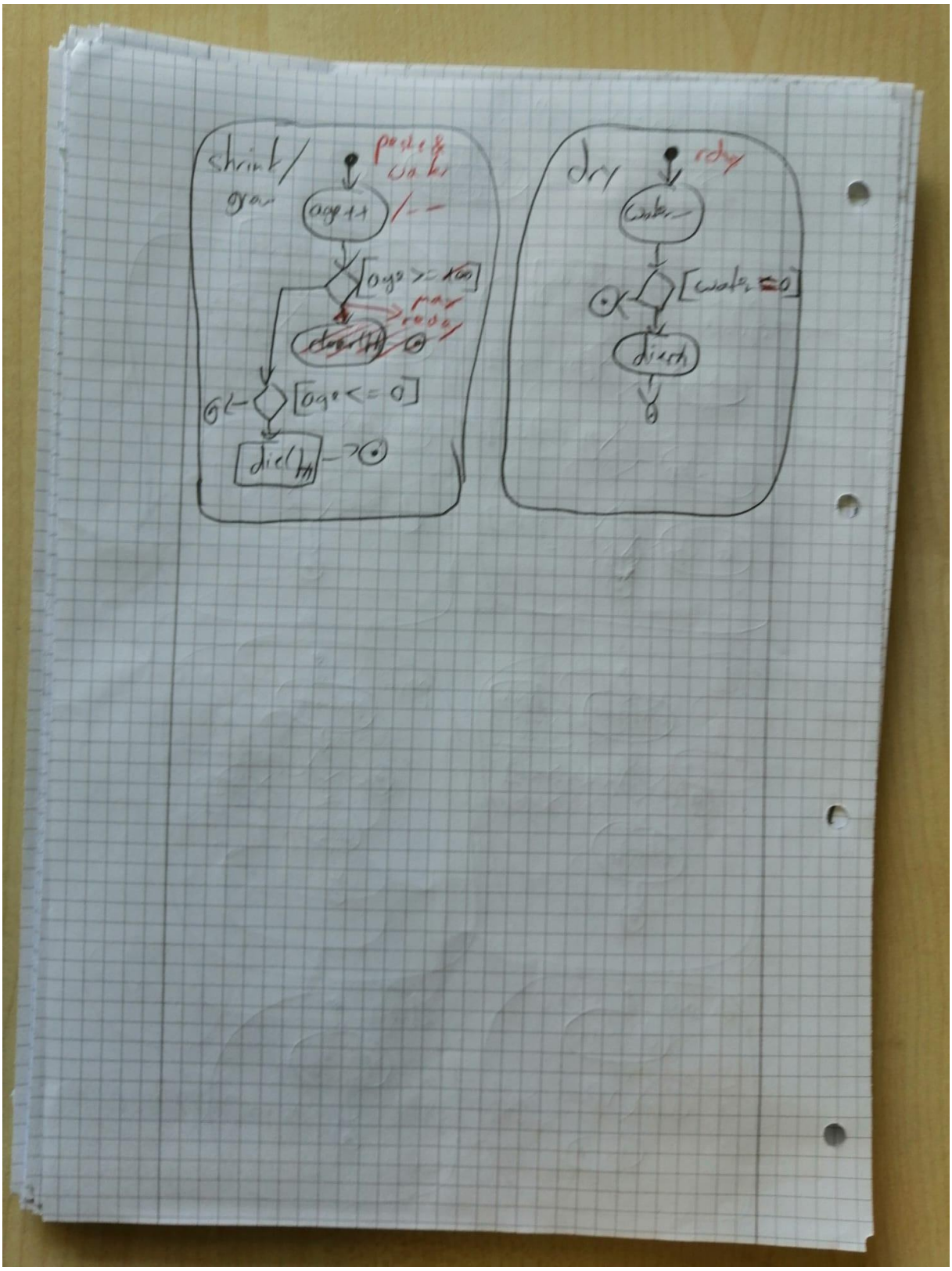
Class Diagramm: GG-Sim 27.06.2022
Feld
Field
Zeit: number
Pflanzen: Pflanze[]
Parasiten: Parasit[]
Felder: Feld[]
clear()
clicked()
draw()
now Simulation
now Plant
position
empty
Pflanze
Plant
+plant
Wasser: number
Dünger: number
Parasite: number
Alter: number
Feld: Feld
update()
grow()
shrink()
dry()
die()
Parasit
Pest
Feld: Feld
fly()
draw()
Constructor
draw()
+ Assets
+ max Stats
+ Pests
Player
Geld: number
Dünger: number
seeds[]
action
plantAction
düngen()
wässern()
pflanzen()
ernten()
pesticide()
Market
Preis Pflanze: number
lastTime
last Price & Price fluctuation
Fluktuation: number[]
+ draw()
+ priceUpdate()
+ visualUpdate()
+ manipulate
FieldObject
?
?
SubPlant
Same Variables but different Values
priceUpdate()
void
Simulation
random Events
time Action
run()
timer()
update()
updateAnimation()
Asset
asset1
asset2
asset3
field
plant
etc.
load()


AD/GG-Sim/Parasit
timeupdate
[fly]
[Eat]
fly
choose rdm plant
animate
kill plant
eat
Shrink



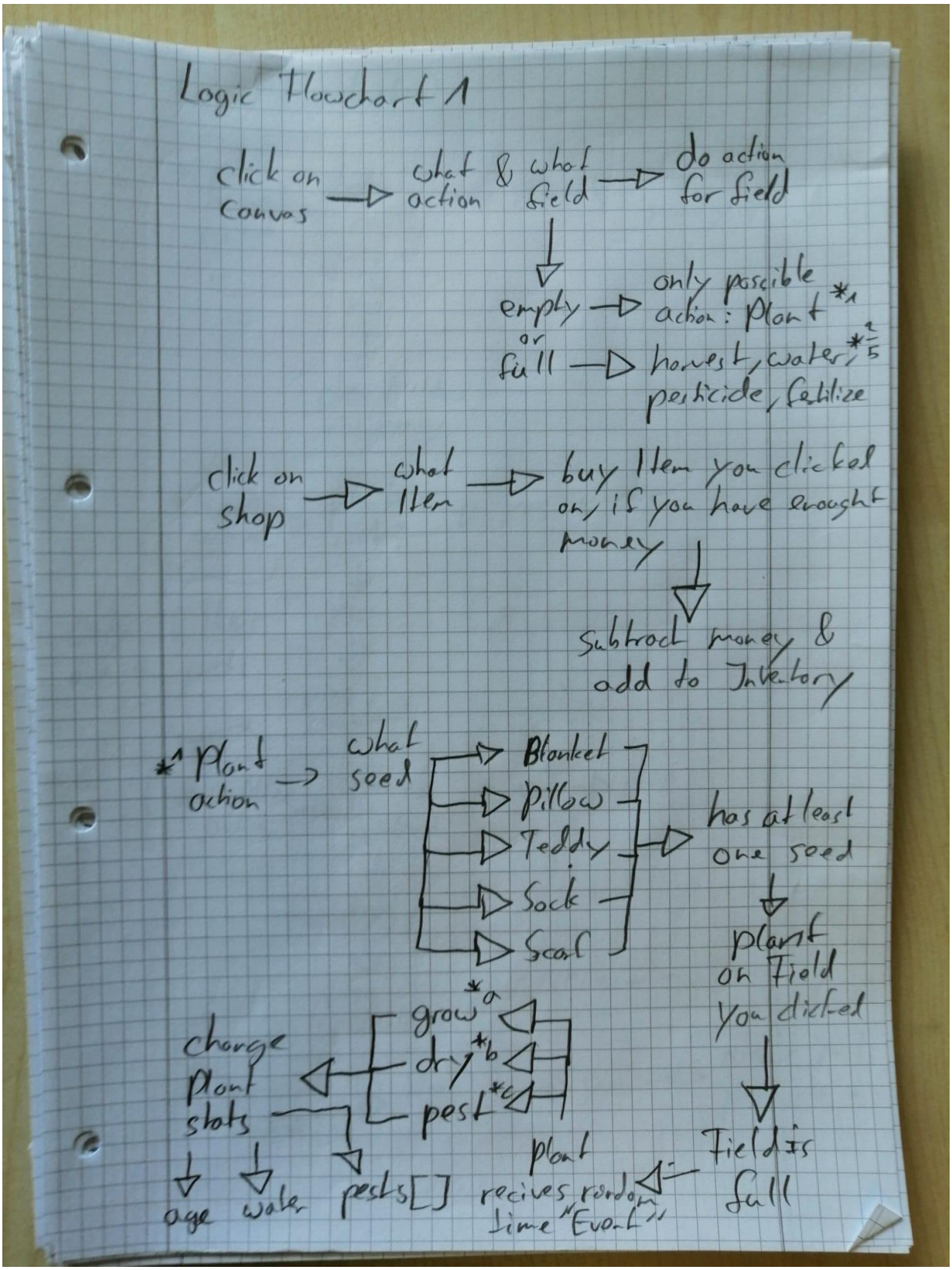
AD / GG-Sim / Hotbar / Player
ACTION
PLANTACTION
interface Seed
click
fertilize
visual selected
[click on plant & Dünger > 0]
Dünger -- → nach plant
Wässern
vis select
[click on plant]
pflanzen
vis select
[click on empty & Same > 0]
Sametyp --
ernten
vis select
[click on rdy plant]
clear
grow
Feld
Geld ++
nach momentanem Preis
pechi
vis select
[click on plant]
pechi > 0
task pechi
pechi --

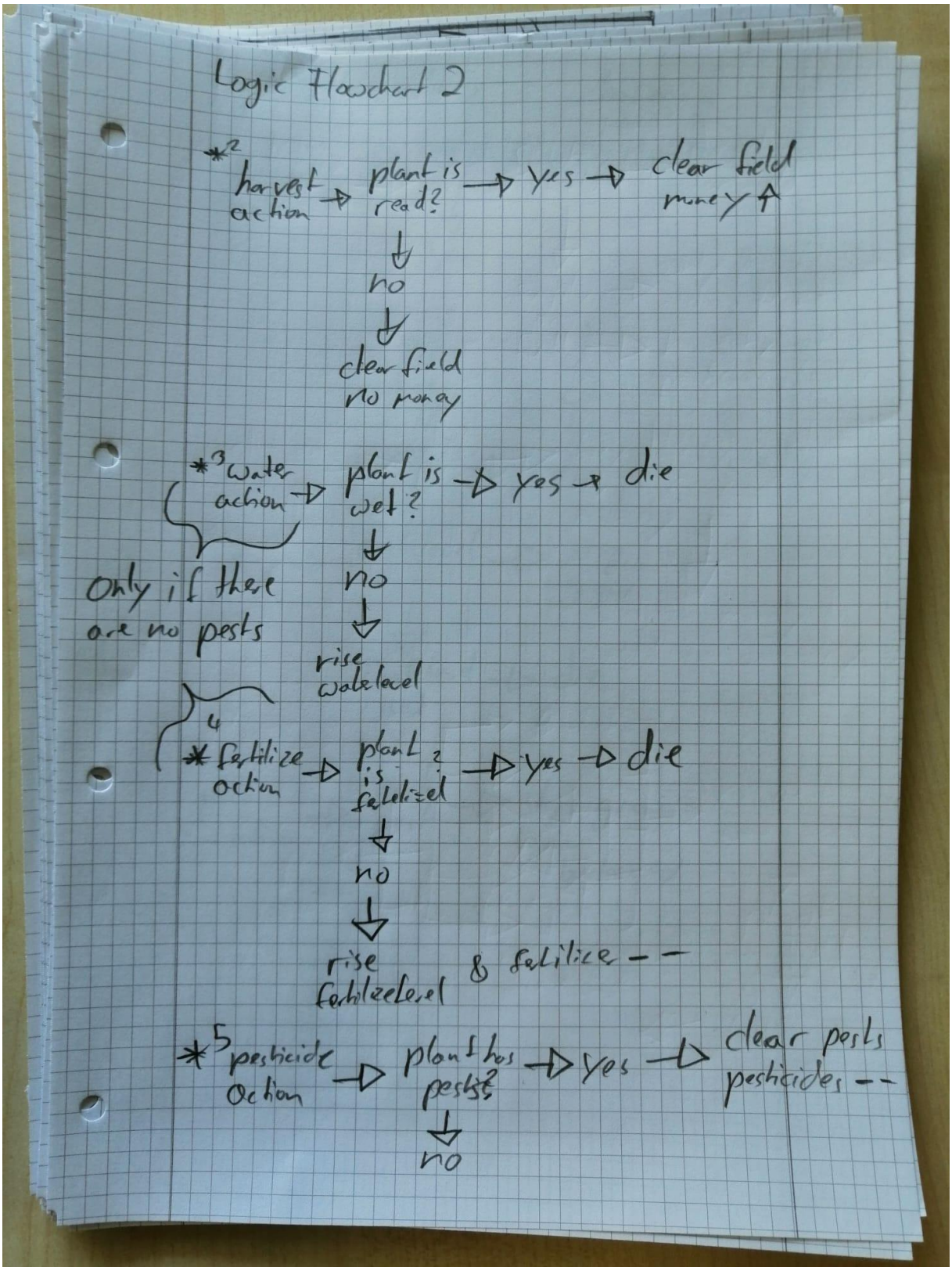

AD / GG-Sim / Plant /
tick
plant update
action: TIMEACTION
plant: Plant
task: task
plant update
[task=cole]
[task=dünge]
[task=pesti]
time update
grow
shrink
dry
cole
pesti
dünge()
die()
harvest
draw
constructor

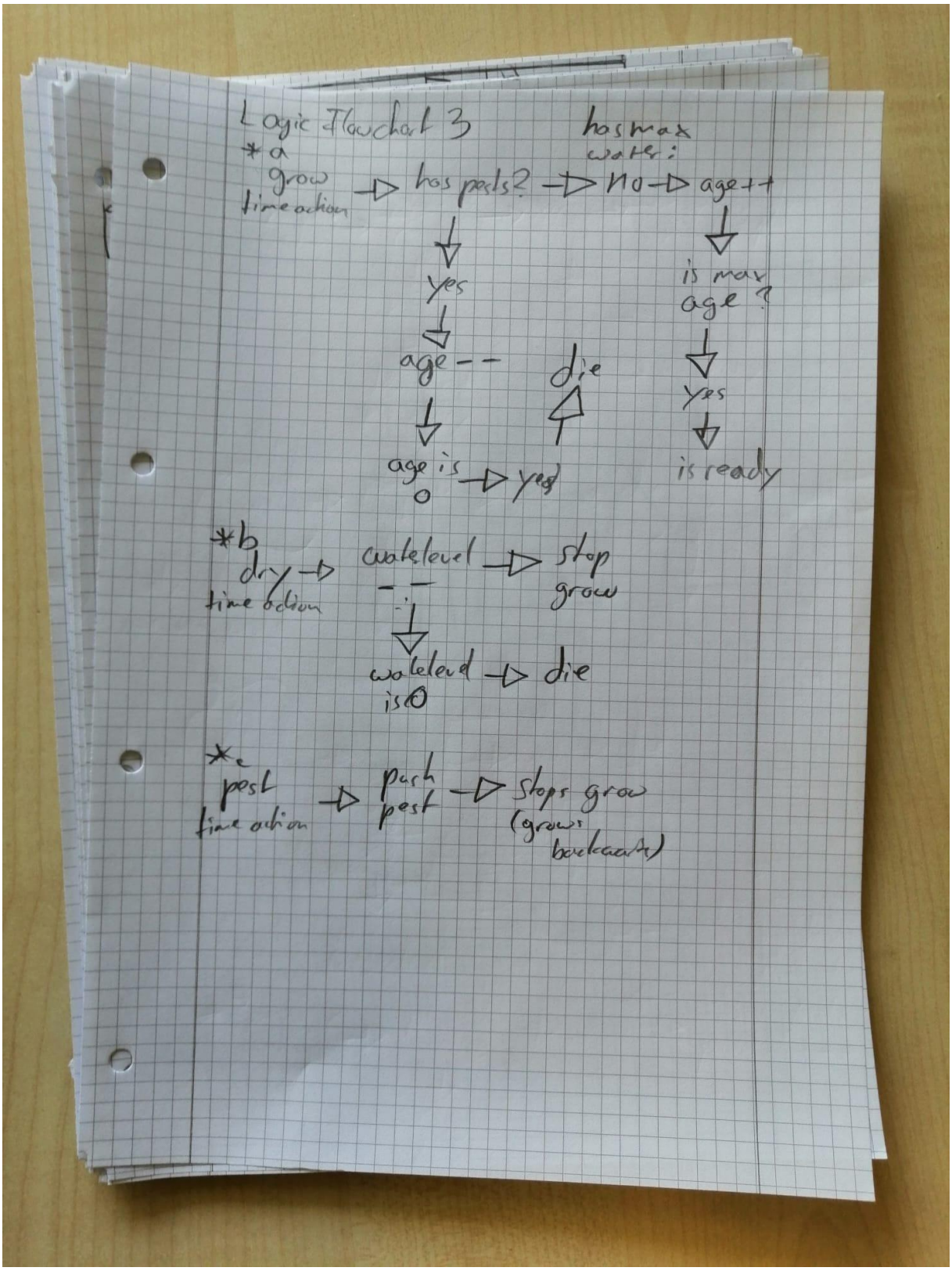

shrink/
grow
dry
[ogo <= 0]

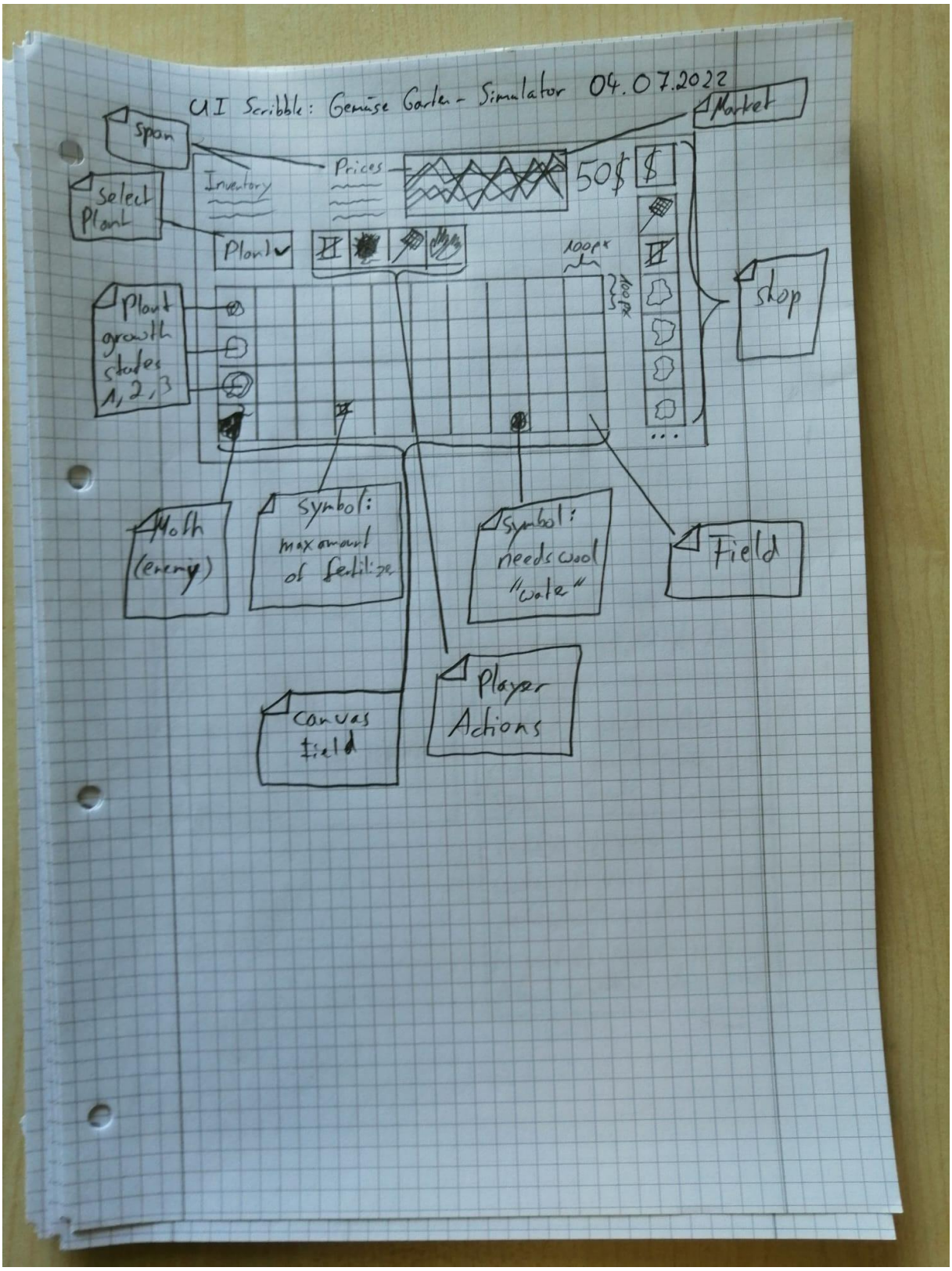
Ende erstes Konzept

Beginn letztes Konzept

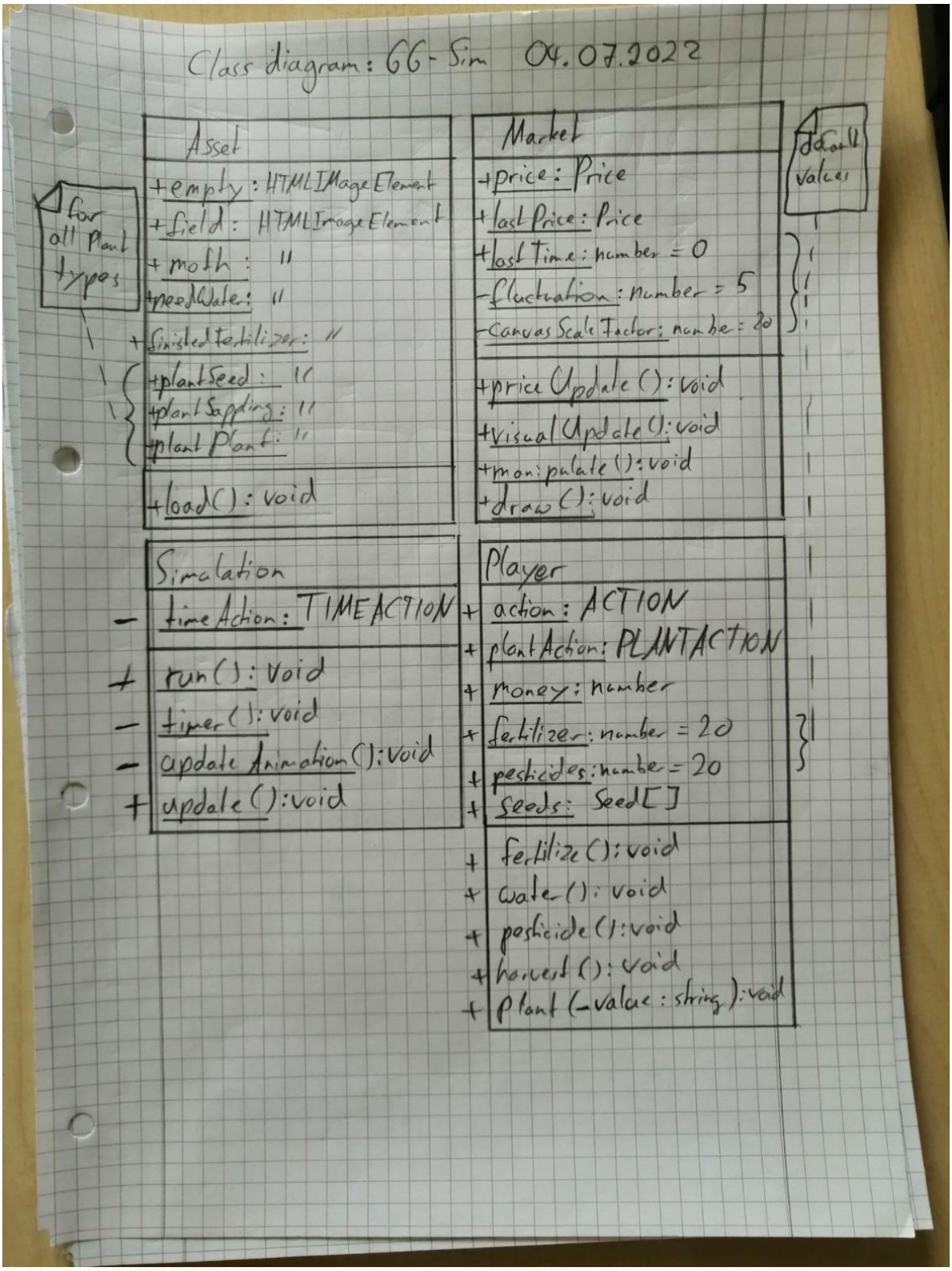

Logic Flowchart 1
click on canvas
what action
&
what field
do action for field
empty
only possible action: Plant *1
or
full
harvest, water, *2/5
pesticide, fertilize
click on shop
what Item
buy Item you clicked on, if you have enought money
Subtract money & add to Inventory
*1 Plant action
what seed
Blanket
pillow
Teddy
Sock
Scarf
has at least one seed
plant on Field you clicked
grow *a
dry *b
pest *c
charge Plant stats
age
water
pests[]
Plant recives random time "Event"
Field is full


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
no
Only if there are no pests
rise watelevel
4
*fertilize action
plant is fatelized?
yes
die
no
rise fertilizelevel
& falilize --
*5 pesticide action
plant has pests?
yes
clear pests
pesticides --
no

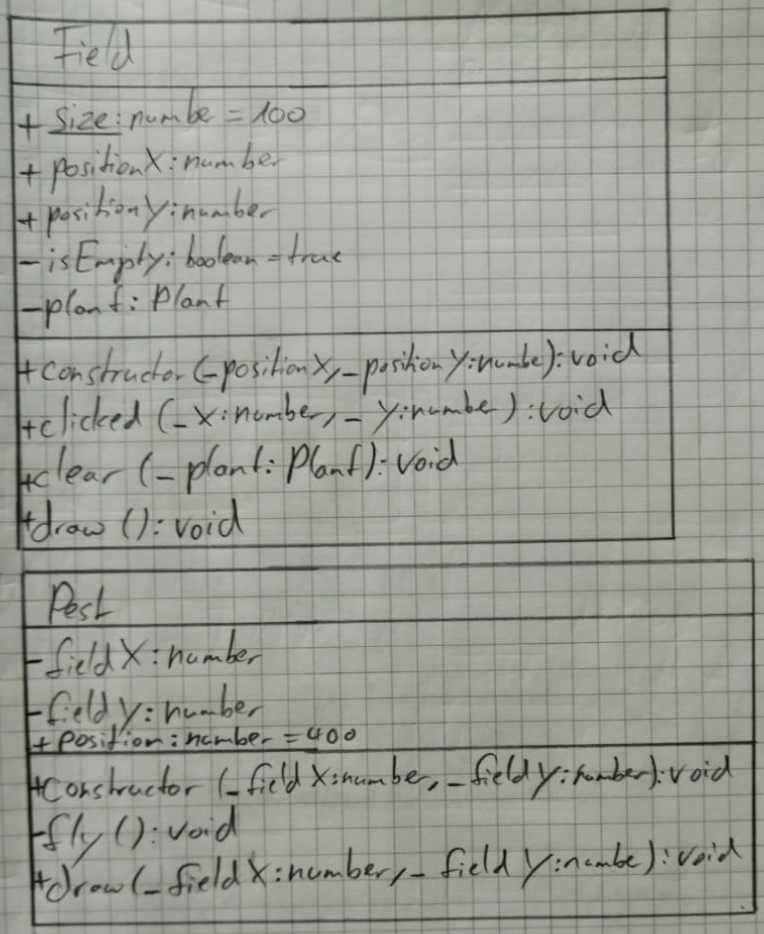

Logic flowchart 3
*a
grow
time action
has pests?
yes
no
age--
age++
is max age?
yes
is ready
die
age is 0
yes
has max water
*b
dry
time action
waterlevel --
waterlevel is 0
stop grow
die
*c
pest
time action
push pest
stops grow
(grows backwards)


UI Scribble: Gemüse Garten - Simulator 04.07.2022
Spon
Market
Inventory
Prices
50$
$
Select Plant
Plant
100px
100px
Plant growth states 1, 2, 3
shop
Moth (enemy)
Symbol: max amount of fertilize
Symbol: needs cool "wate"
Field
Player Actions
canvas field

Class diagram: 66-Sim 04.07.2022



Class diagram: 66-Sim 04.07.2022

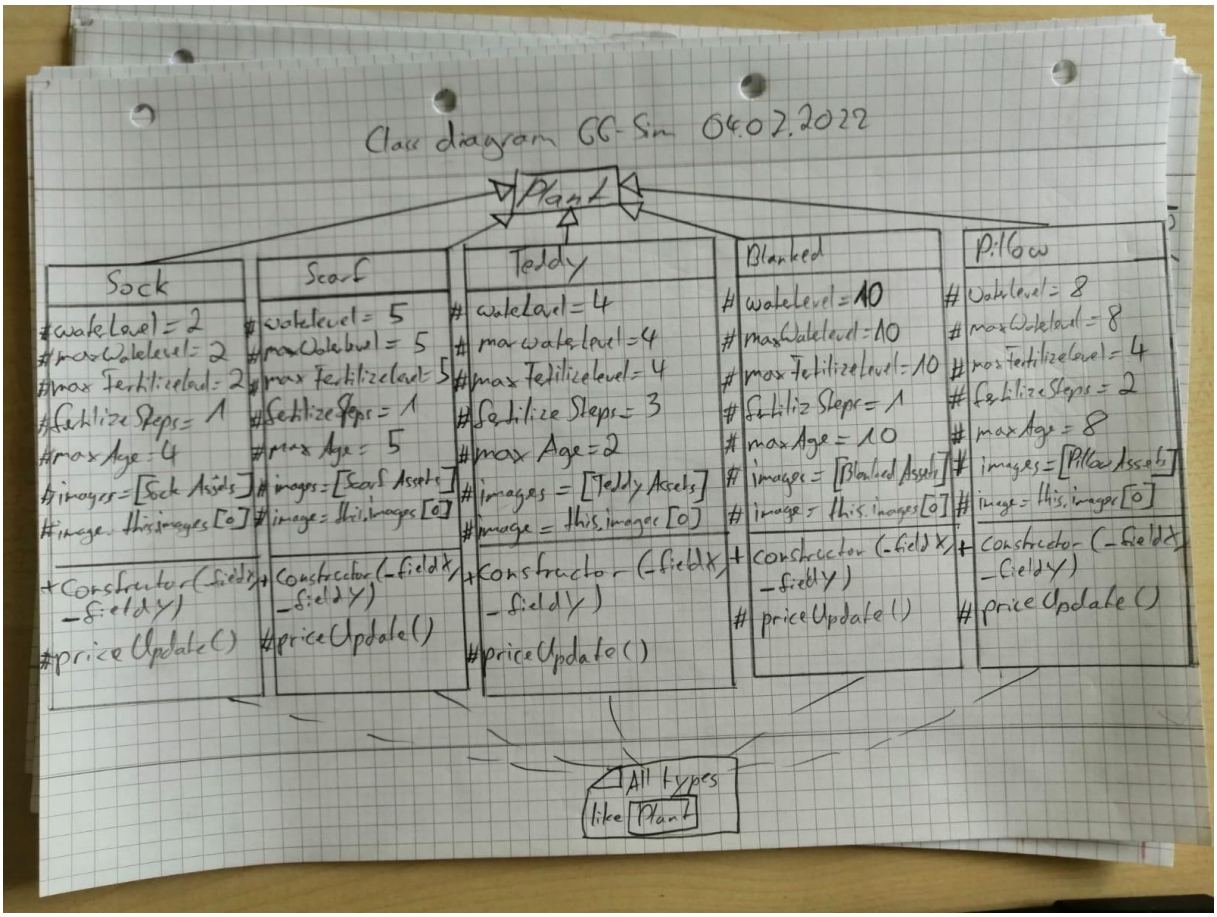


Class diagram: 66-Sim 04.07.2022

Plant

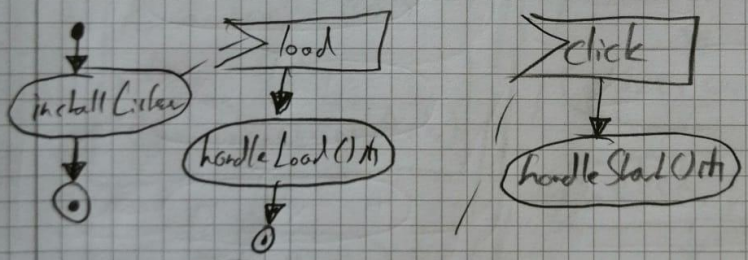
+ fieldX: number
+ fieldY: number
# waterLevel: "
# maxwaterLevel: "
- fertilizeLevel: " = 0
maxFertilizelevel: "
- age: " = 0
maxAge: "
priceValue: " = 1
- isReady: boolean = false
+ pests: Pest[] = []
# images: HTMLImageElement[]
# image: HTMLImageElement
- statusLevelImages: HTMLImageElement[] = [Asset.empty,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

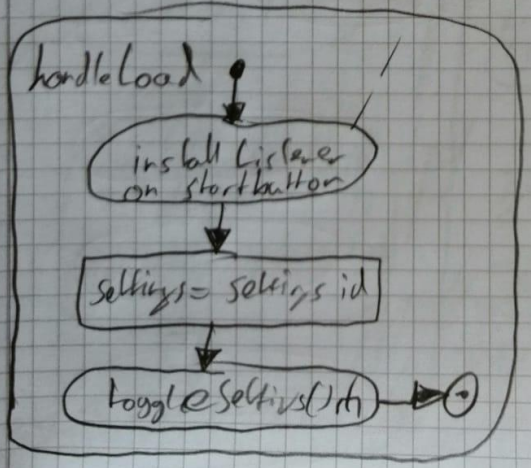
+ constructor(_fieldX: number, _fieldY: number): void
# priceUpdate(): void
+ timeUpdate(_action: TIMEACTION): void
+ playerUpdate(_plant: Plant): void
- getWatered(): void
- getFertilized(): void
- getPesticided(): void
- getHarvested(): void
- grow(): void
- shrink(): void
- dry(): void
- die(): void
+ draw(): void

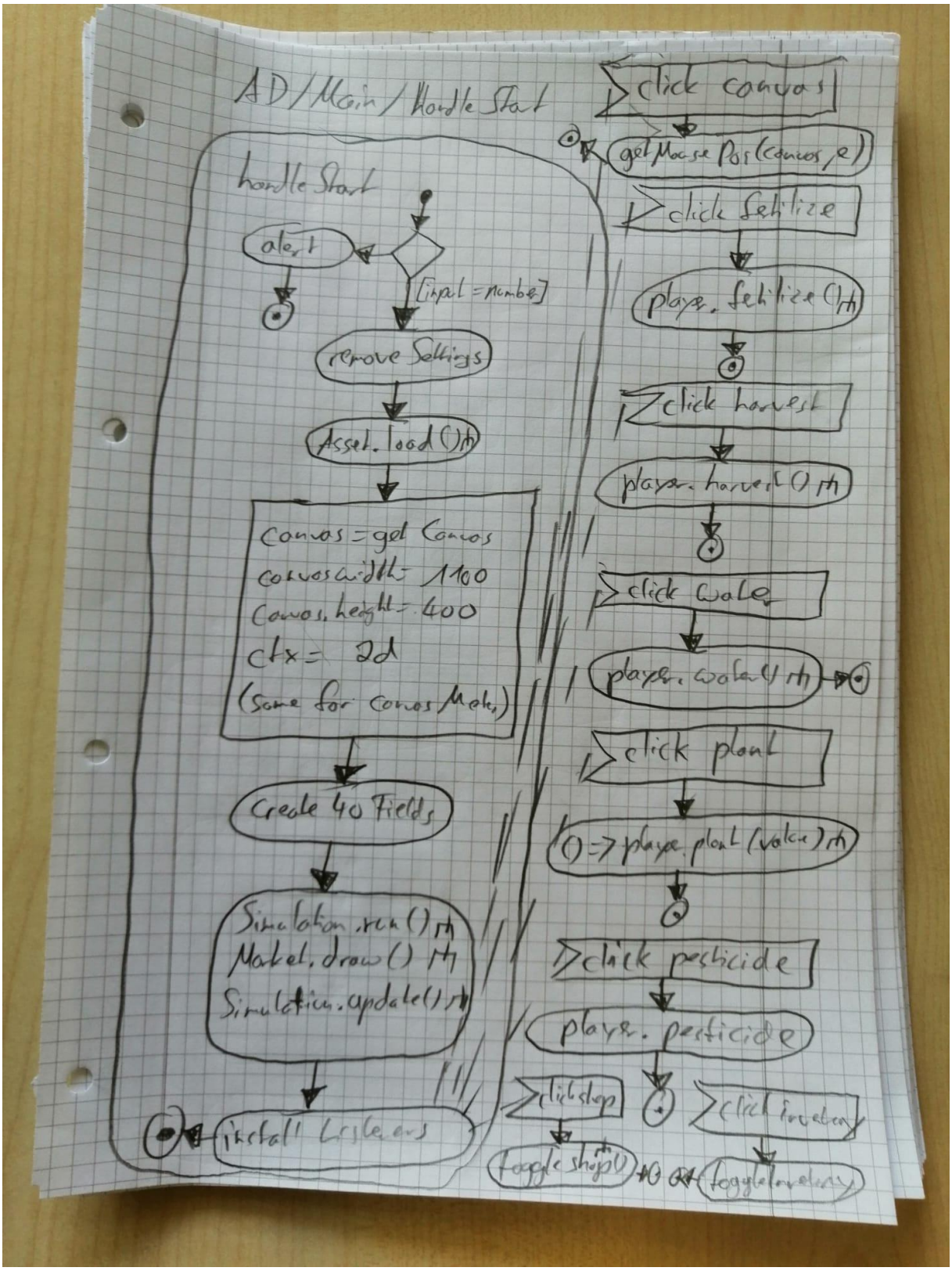

Class diagram GG-Sim 04.07.2022
Plant
Sock
#waterLevel = 2
#maxWaterLevel = 2
#maxFertilizeLevel = 2
#fertilizeSteps = 1
#maxAge = 4
#images = [Sock Assets]
#image = this.images[0]
+Constructor(-fieldX, -fieldY)
#priceUpdate()
Scarf
#waterLevel = 5
#maxWaterLevel = 5
#maxFertilizeLevel = 5
#fertilizeSteps = 1
#maxAge = 5
#images = [Scarf Assets]
#image = this.images[0]
+Constructor(-fieldX, -fieldY)
#priceUpdate()
Teddy
#waterLevel = 4
#maxWaterLevel = 4
#maxFertilizeLevel = 4
#fertilizeSteps = 3
#maxAge = 2
#images = [Teddy Assets]
#image = this.images[0]
+Constructor(-fieldX, -fieldY)
#priceUpdate()
Blanked
#waterLevel = 10
#maxWaterLevel = 10
#maxFertilizeLevel = 10
#fertiliz Steps = 1
#maxAge = 10
#images = [Blanked Assets]
#image = this.images[0]
+Constructor(-fieldX, -fieldY)
#priceUpdate()
Pillow
#waterLevel = 8
#maxWaterLevel = 8
#maxFertilizeLevel = 4
#fertilizeSteps = 2
#maxAge = 8
#images = [Pillow Assets]
#image = this.images[0]
+Constructor(-fieldX, -fieldY)
#priceUpdate()
All types like Plant

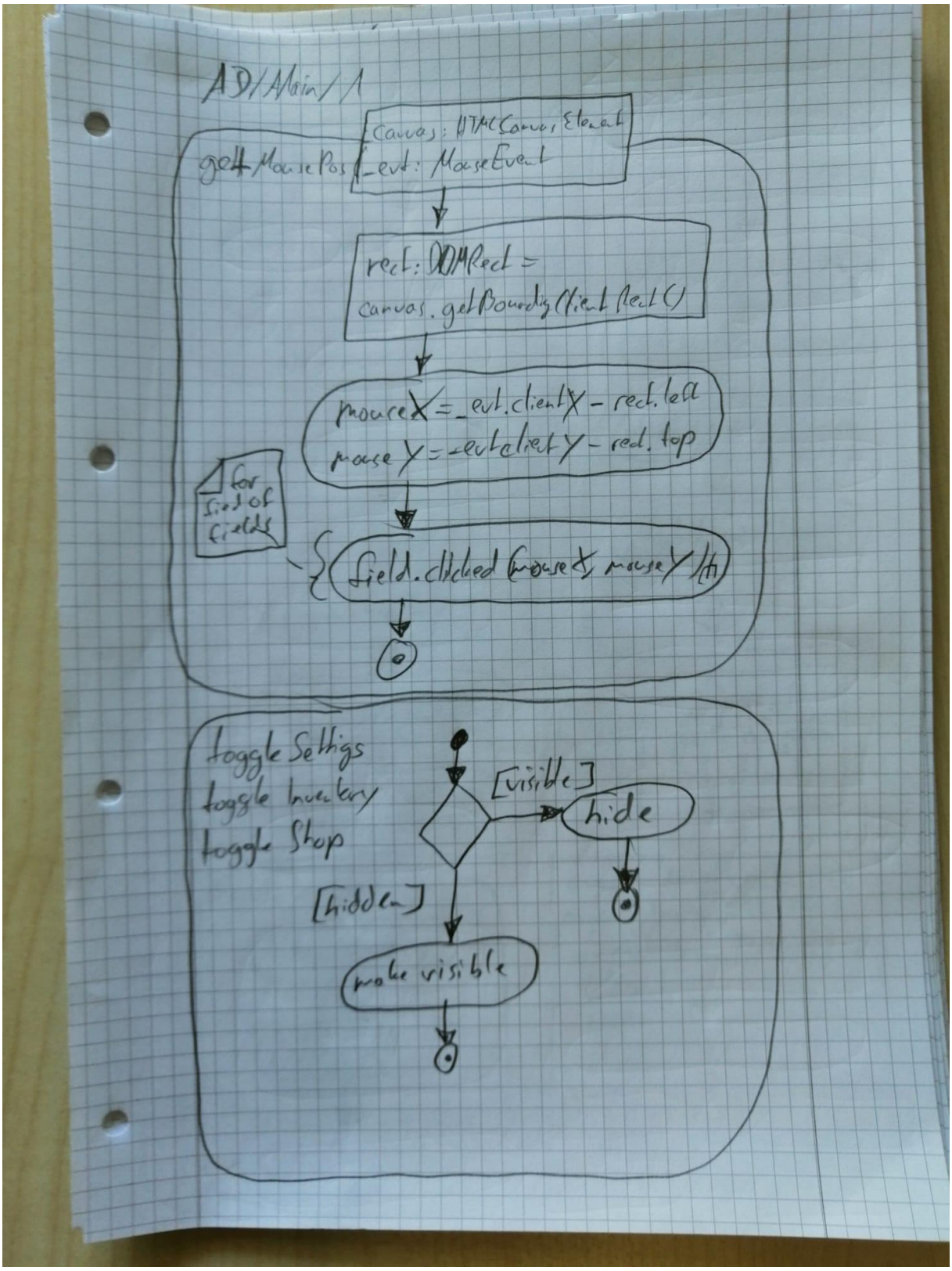
AD/Main/Variables & Listener for WillLoad

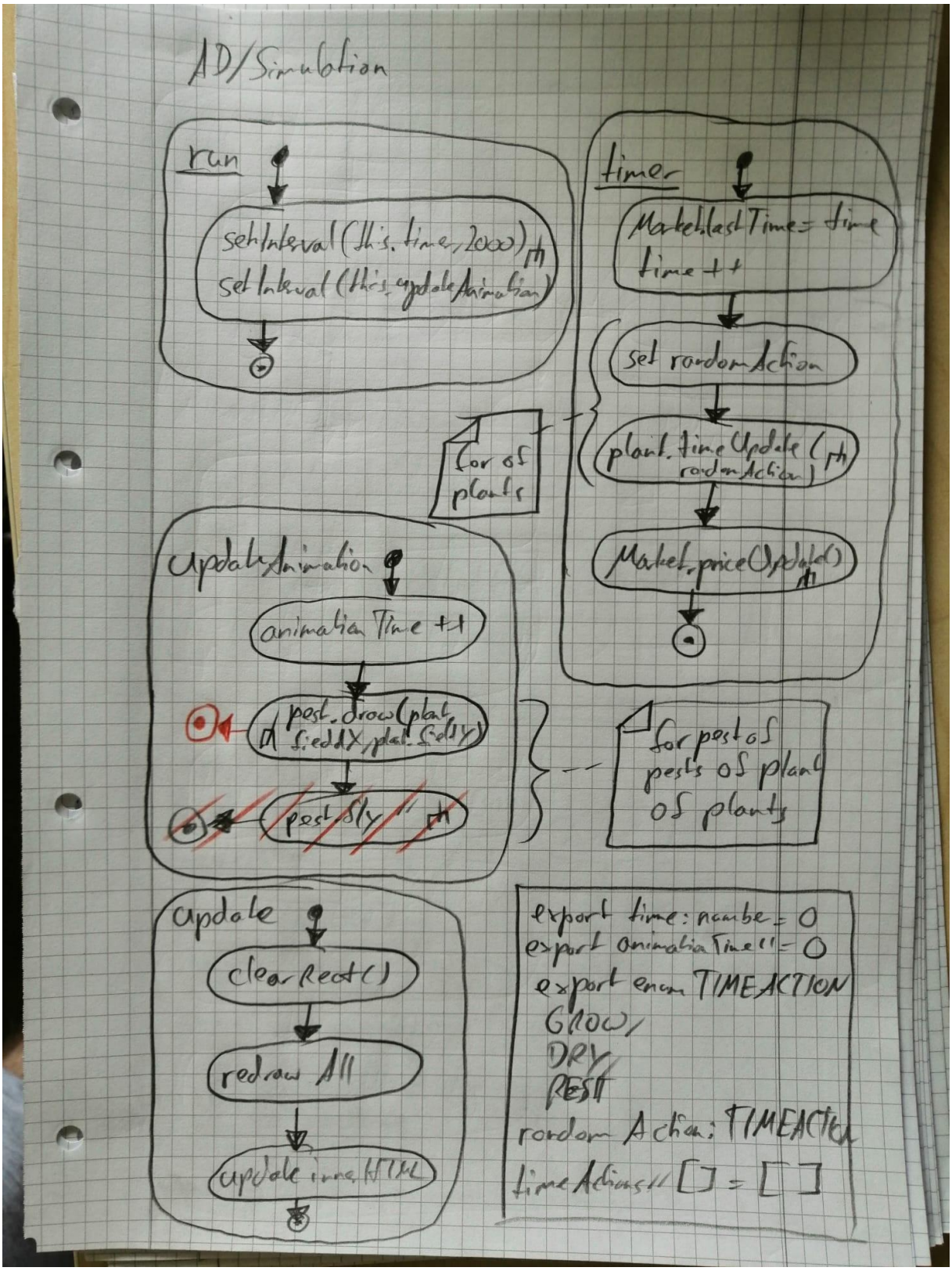
export: canvas: HTMLCanvasElement
canvasM: //
ctx: CanvasRenderingContext2D
ctxM: //
mousex: number
mousey: number
player: Player = new Player
market: Market = new Market
plants: Plant[] = []
fields: Field[] = []
Shop: HTMLElement
Settings: //

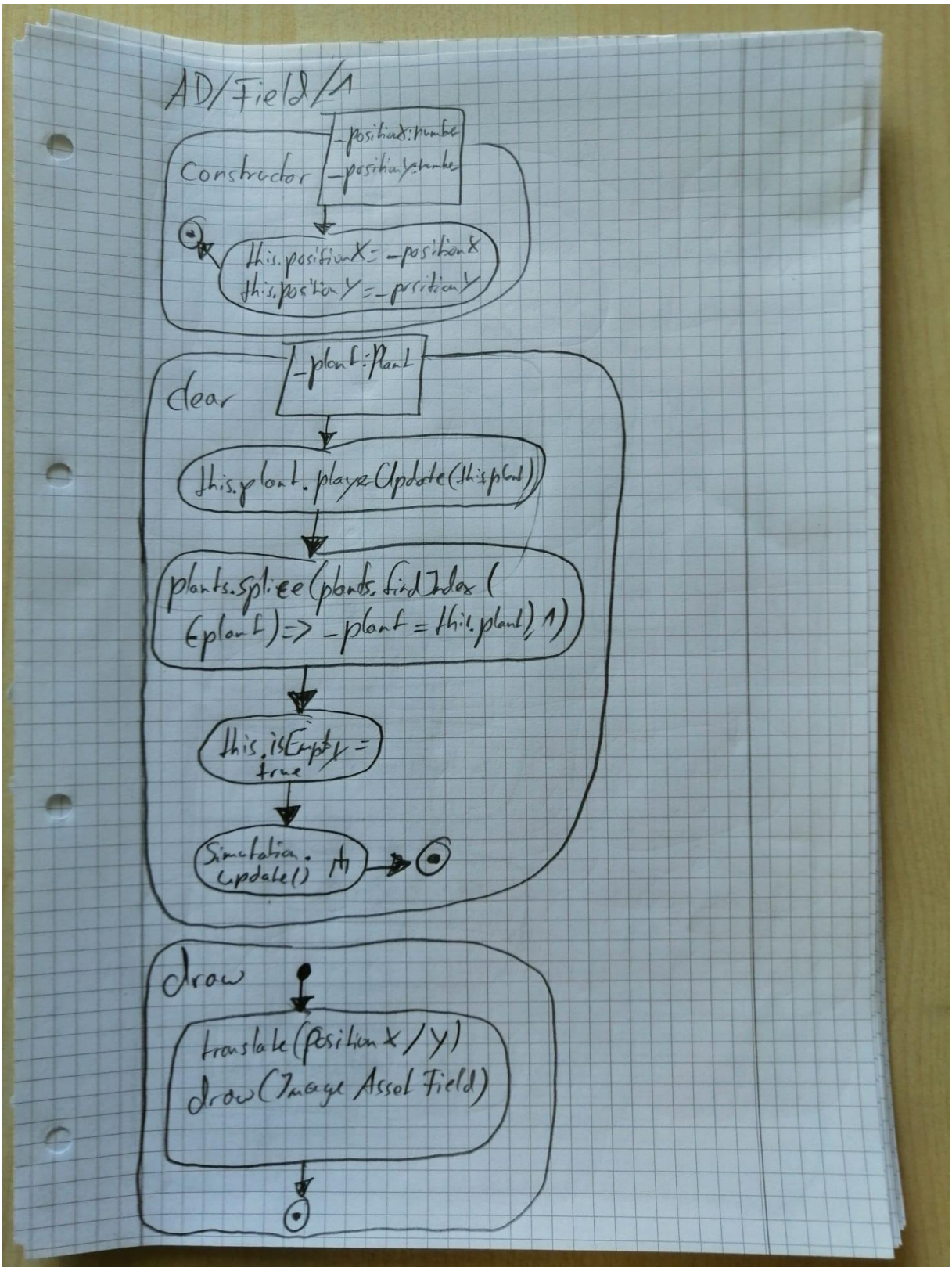


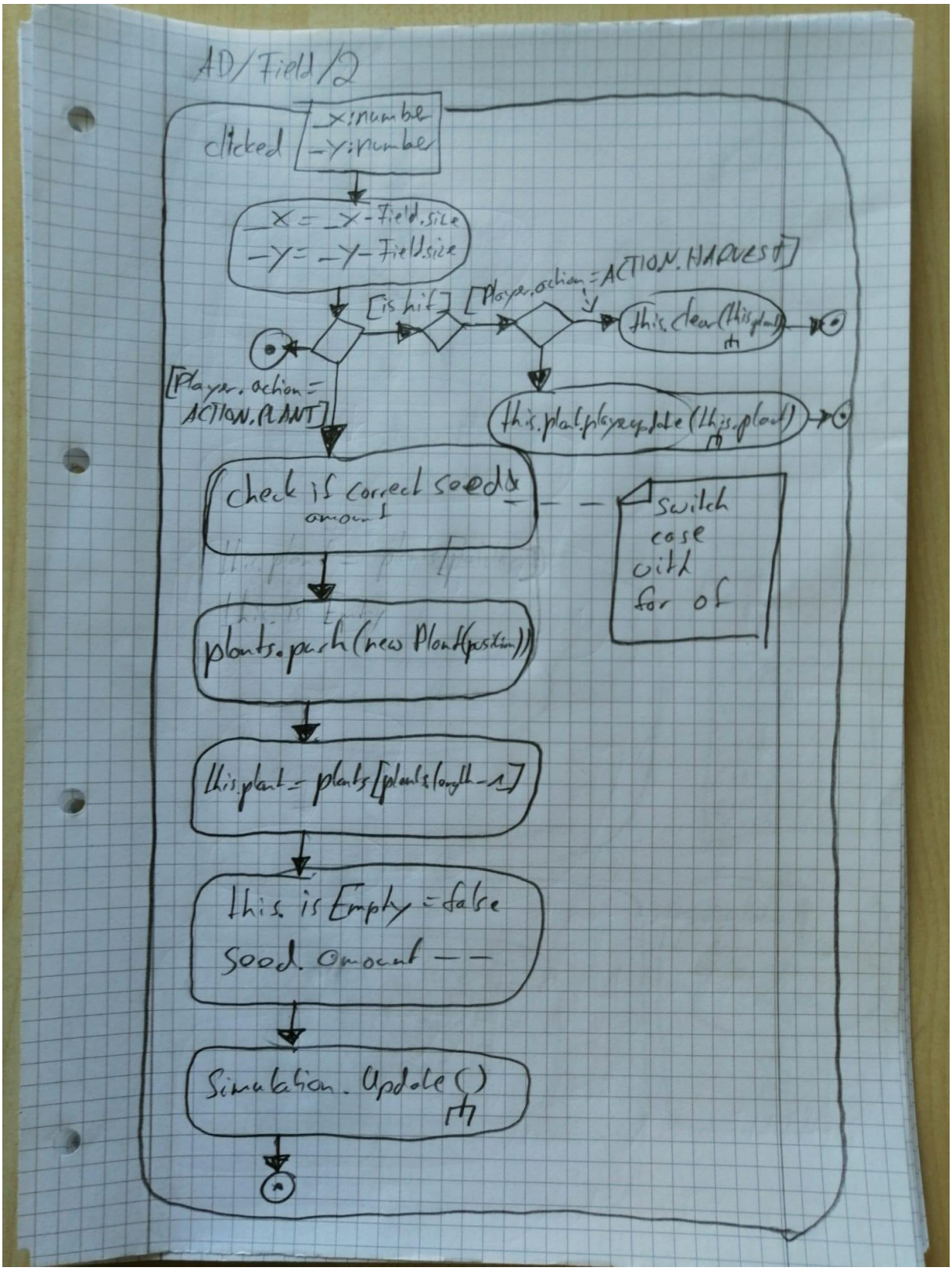


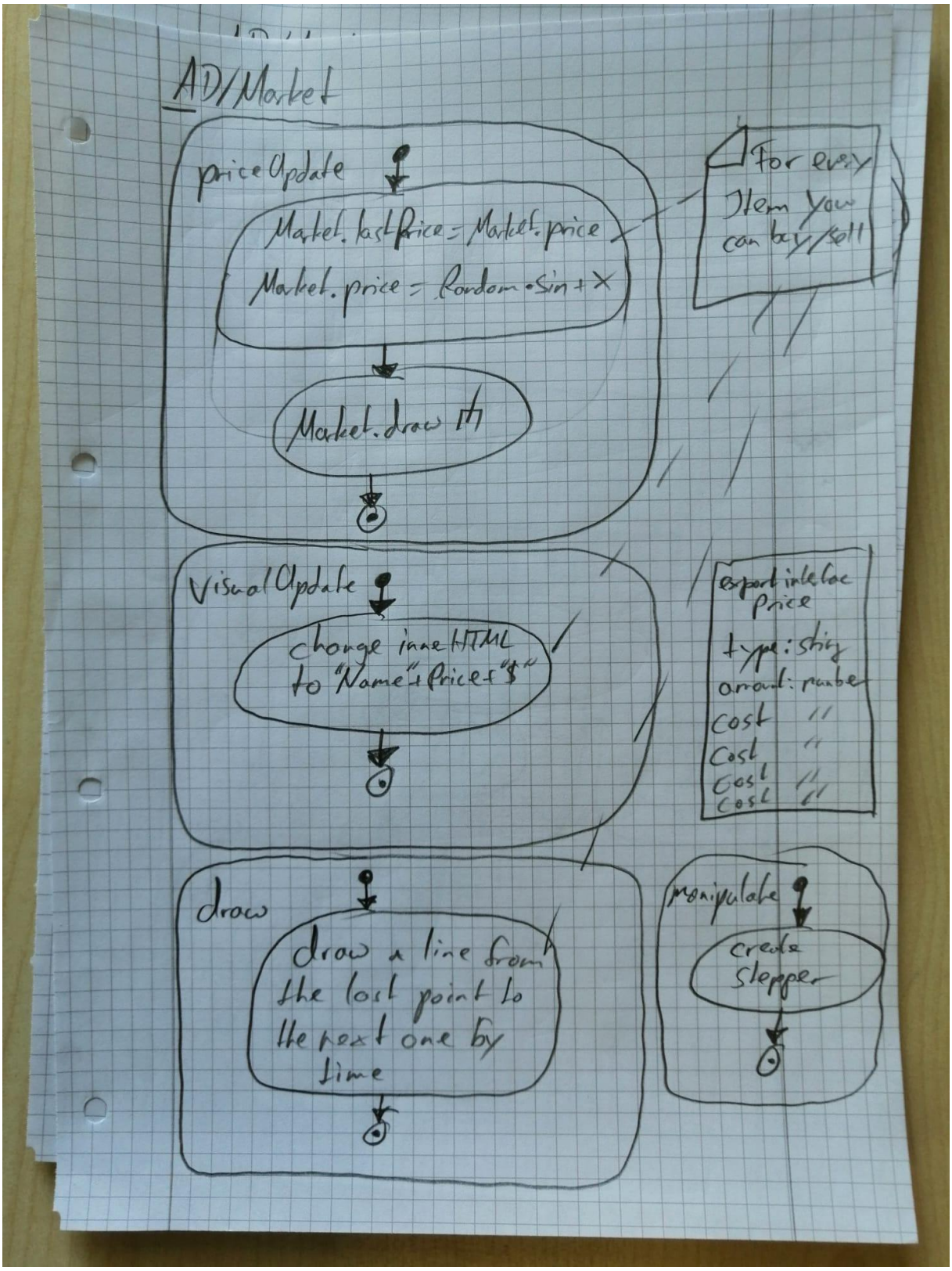

AD/Main/handleStart
handleStart
alert
[input = number]
remove Settings
Asset.load()
canvas = getCanvas
canvas.width = 1100
canvas.height = 400
ctx = 2d
(same for canvasMarket)
create 40 Fields
Simulation.run()
Market.draw()
Simulation.update()
install Listeners
click canvas
getMousePos(canvas, e)
click fertilize
player.fertilize()
click harvest
player.harvest()
click water
player.water()
click plant
() => player.plant(value)
click pesticide
player.pesticide
click shop
toggleShop()
click inventory
toggleInventory

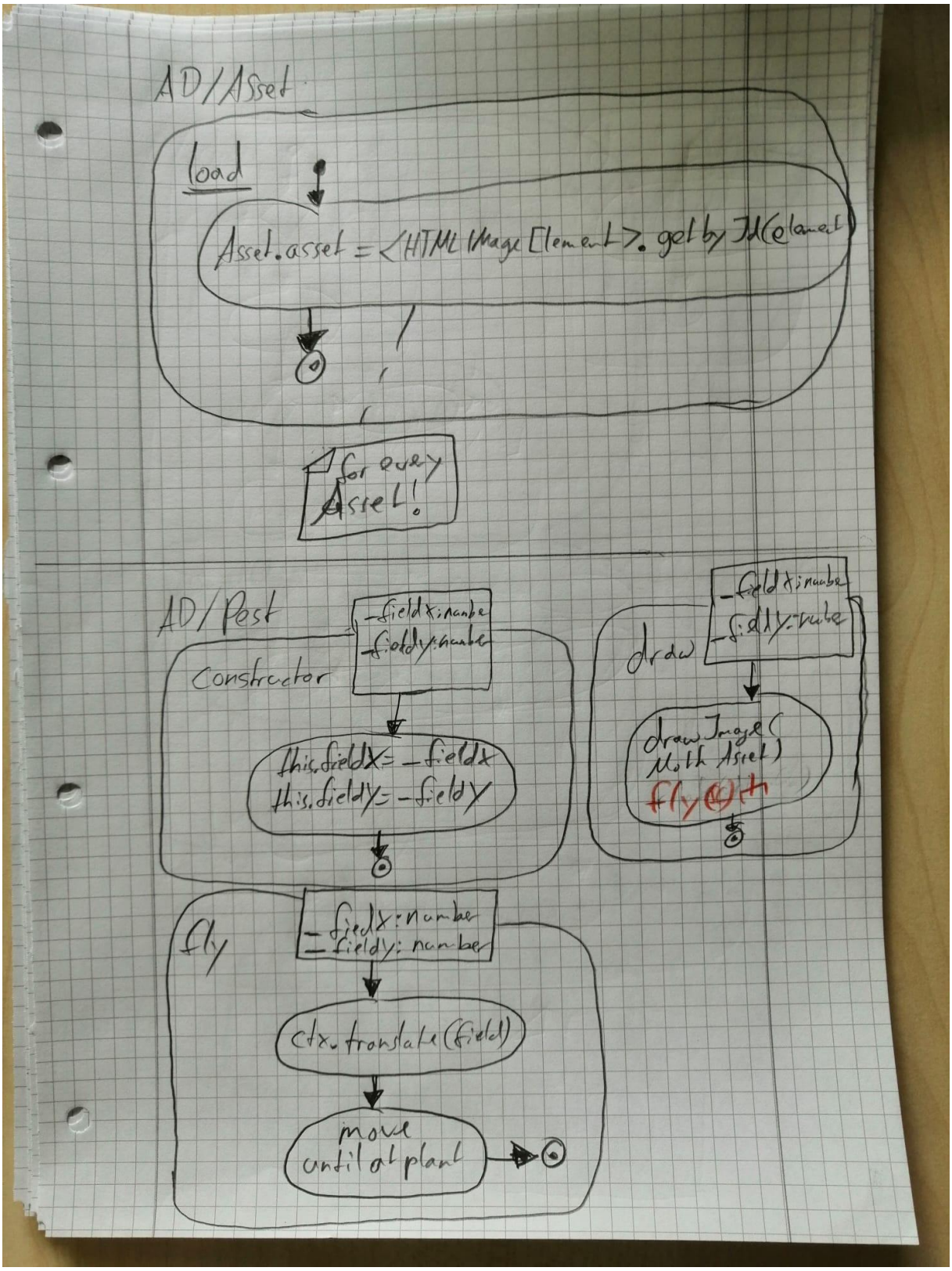

AD/Main/1
getMousePos(_evt: MouseEvent
canvas: HTMLCanvasElement
rect: DOMRect =
canvas.getBoundingClientRect()
mouseX = _evt.clientX - rect.left
mouseY = _evt.clientY - rect.top
for list of fields
field.clicked(mouseX, mouseY)
toggle Settings
toggle Inventory
toggle Shop
[visible]
hide
[hidden]
make visible

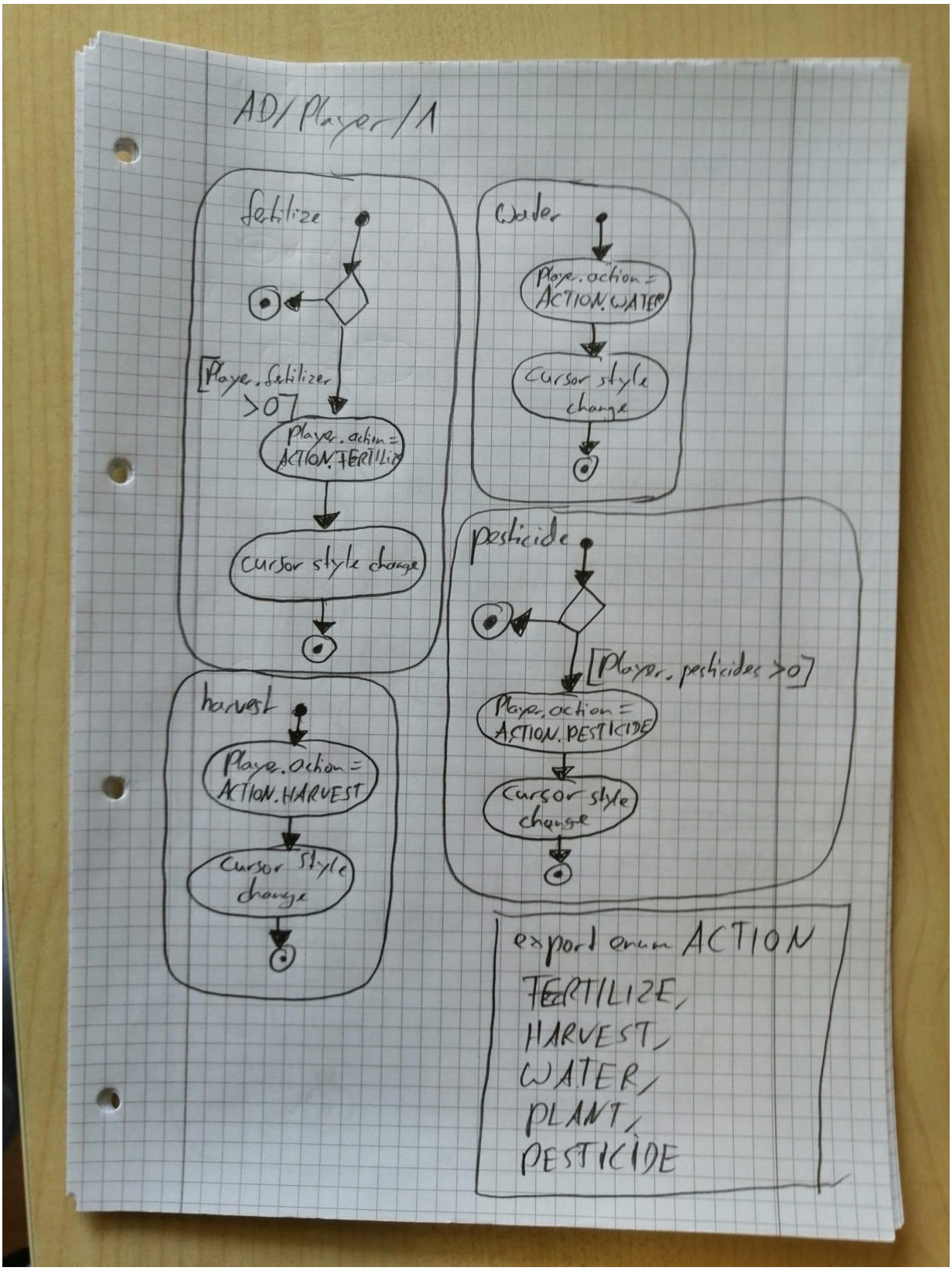

AD/Simulation
run
setInterval(this.time, 2000)
setInterval(this.updateAnimation)
time
Market.lastTime = time
time++
set randomAction
plant.timeUpdate(randomAction)
Market.priceUpdate()
for of plants
updateAnimation
animationTime++
pest.draw(plant.fieldX, plant.fieldY)
pest.fly()
for pest of pests of plant of plants
update
clearRect()
redraw All
update innerHTML
export time: number = 0
export animationTime = 0
export enum TIMEACTION
GROW,
DRY,
REST
randomAction: TIMEACTION
timeActions = [] = [ ]

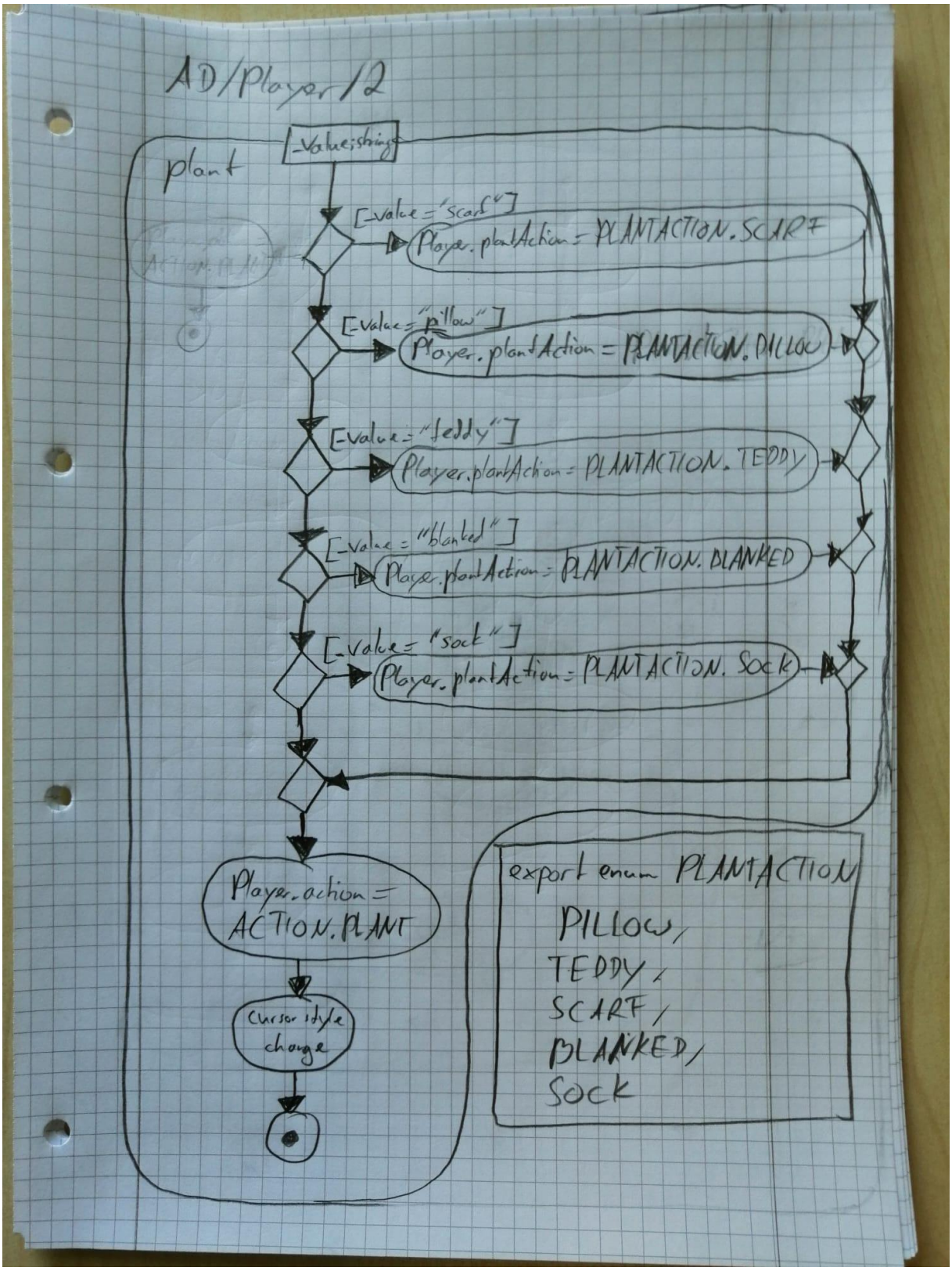

AD/Field/1
Constructor
-positionX:number
-positionY:number
this.positionX = -positionX
this.positionY = -positionY
clear
-plant:Plant
this.plant.playerUpdate(this.plant)
plants.splice(plants.findIndex(
(plant) => -plant = this.plant), 1)
this.isEmpty = true
Simulation.update()
draw
translate(positionX/Y)
draw(ImageAssetField)

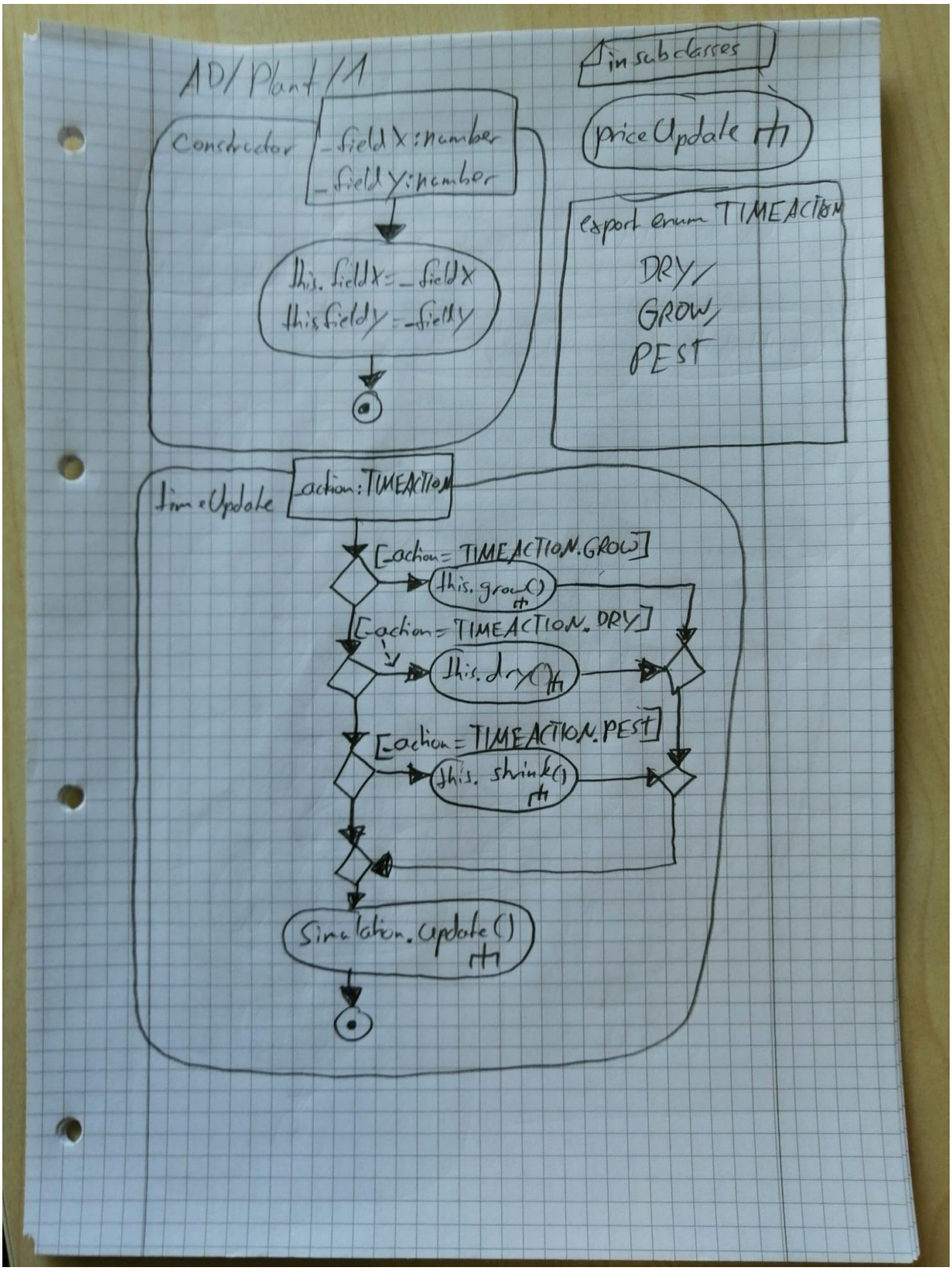

AD/Field/2
clicked
_x:number
_y:number
_x = _x - Field.size
_y = _y - Field.size
[is hit]
[Player.action = ACTION.HARVEST]
this.clear(this.plant)
[Player.action = ACTION.PLANT]
this.plant.playerUpdate(this.plant)
check if correct seed & amount
switch case with for of
plants.push(new Plant(position))
this.plant = plants[plants.length - 1]
this.isEmpty = false
seed.amount - -
Simulation.Update()

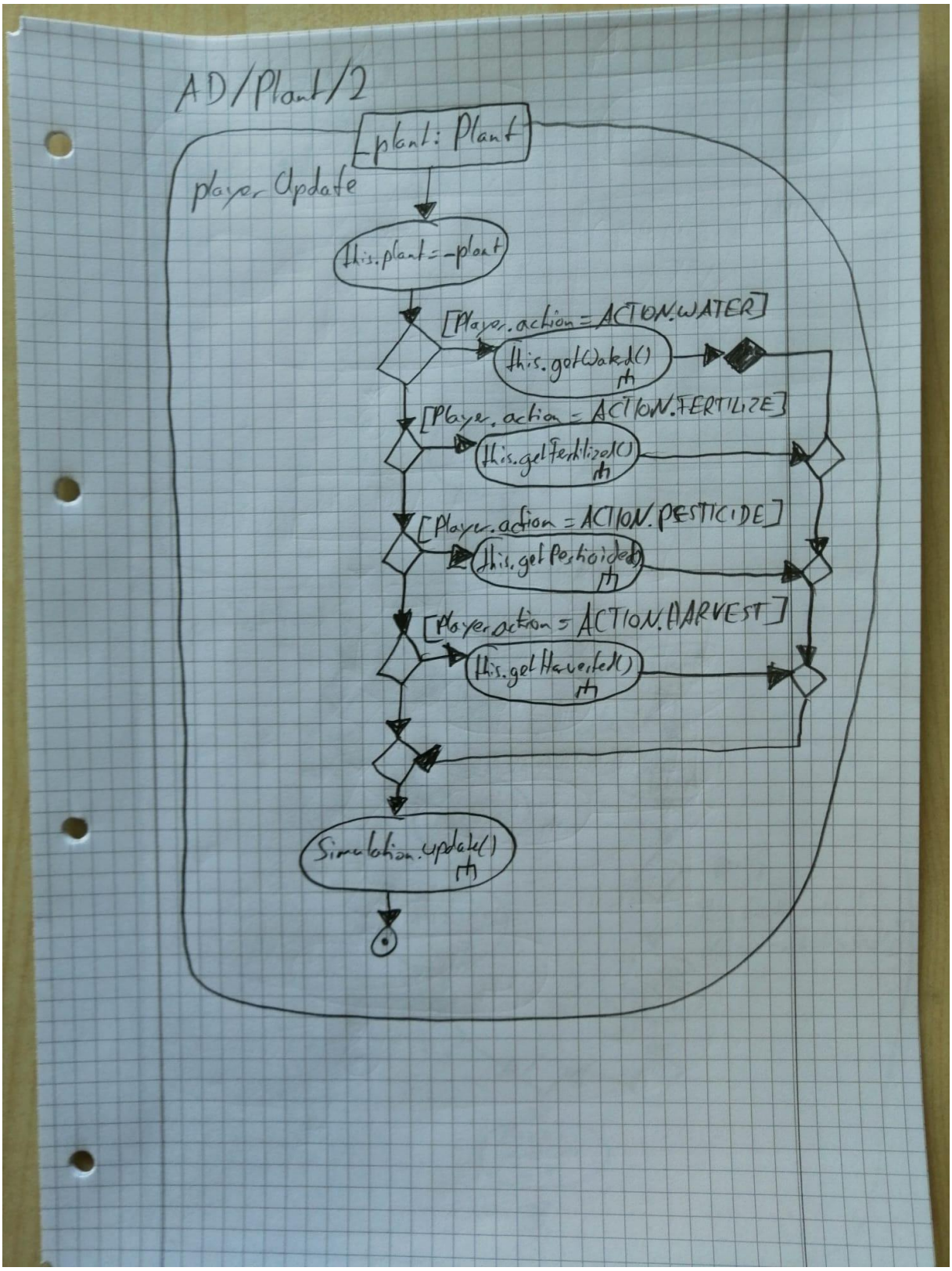

AD/Market
priceUpdate
Market.lastPrice = Market.price
Market.price = Random·Sin+X
Market.drawIt
For every Item You can buy/sell
visualUpdate
change innerHTML to "Name"+Price+"$"
export interface Price
type: string
amount: number
cost
cost
cost
cost
draw
draw a line from the last point to the next one by time
manipulate
create Slopper


AD/Asset
load
Asset.asset = <HTMLImageElement>.getById(elementId)
for every Asset!
AD/Post
Constructor
-fieldX:number
-fieldY:number
this.fieldX = -fieldX
this.fieldY = -fieldY
fly
-fieldX:number
-fieldY:number
ctx.translate(fieldX, fieldY)
move until at planet
draw
-fieldX:number
-fieldY:number
drawImage(MathAsset)
flyMath

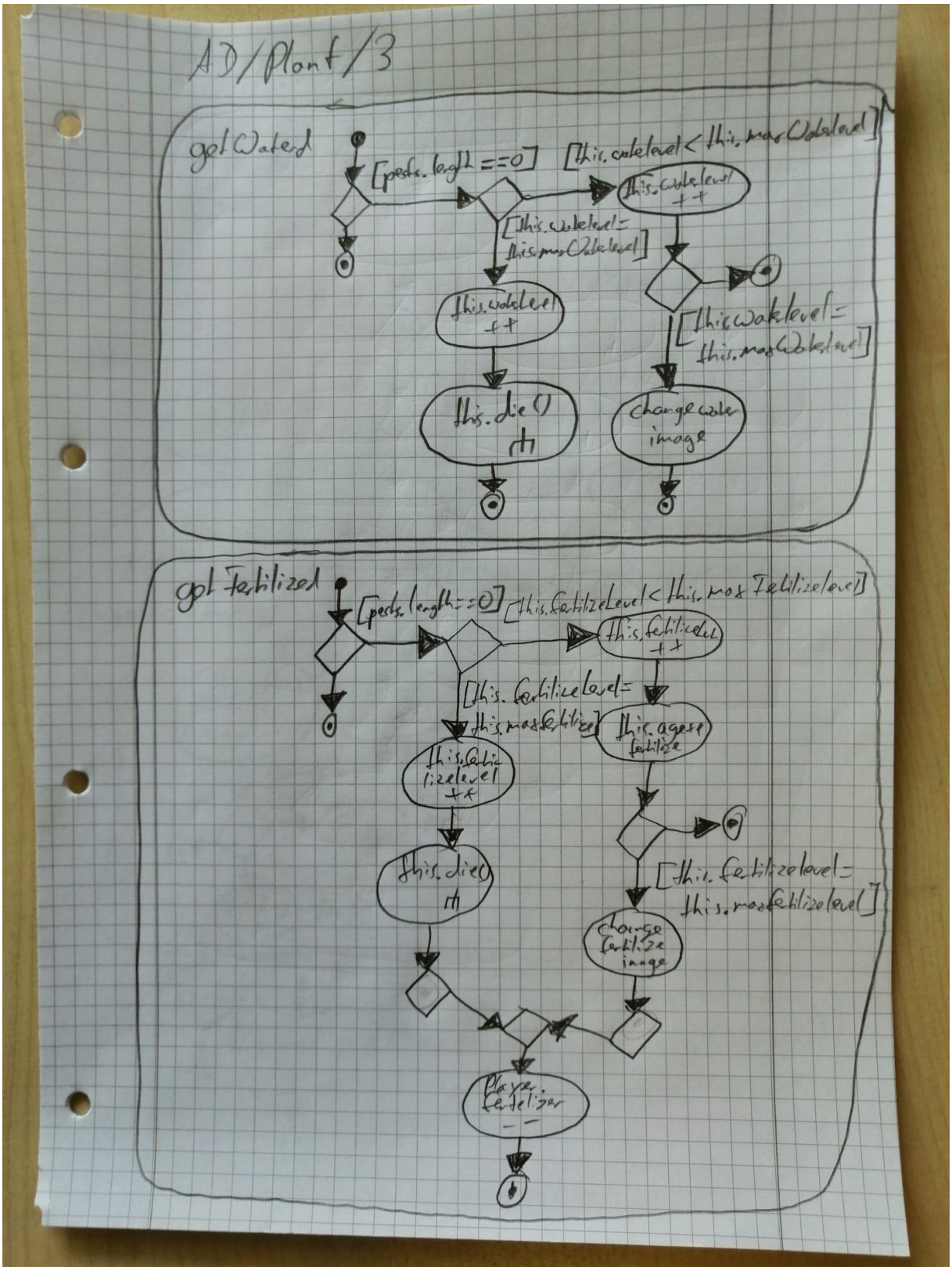

AD/Player/1
fertilize
[Player.fertilizer >0]
Player.action = ACTION.FERTILIZE
cursor style change
water
Player.action = ACTION.WATER
cursor style change
pesticide
[Player.pesticides >0]
Player.action = ACTION.PESTICIDE
cursor style change
harvest
Player.action = ACTION.HARVEST
cursor style change
export enum ACTION
FERTILIZE,
HARVEST,
WATER,
PLANT,
PESTICIDE

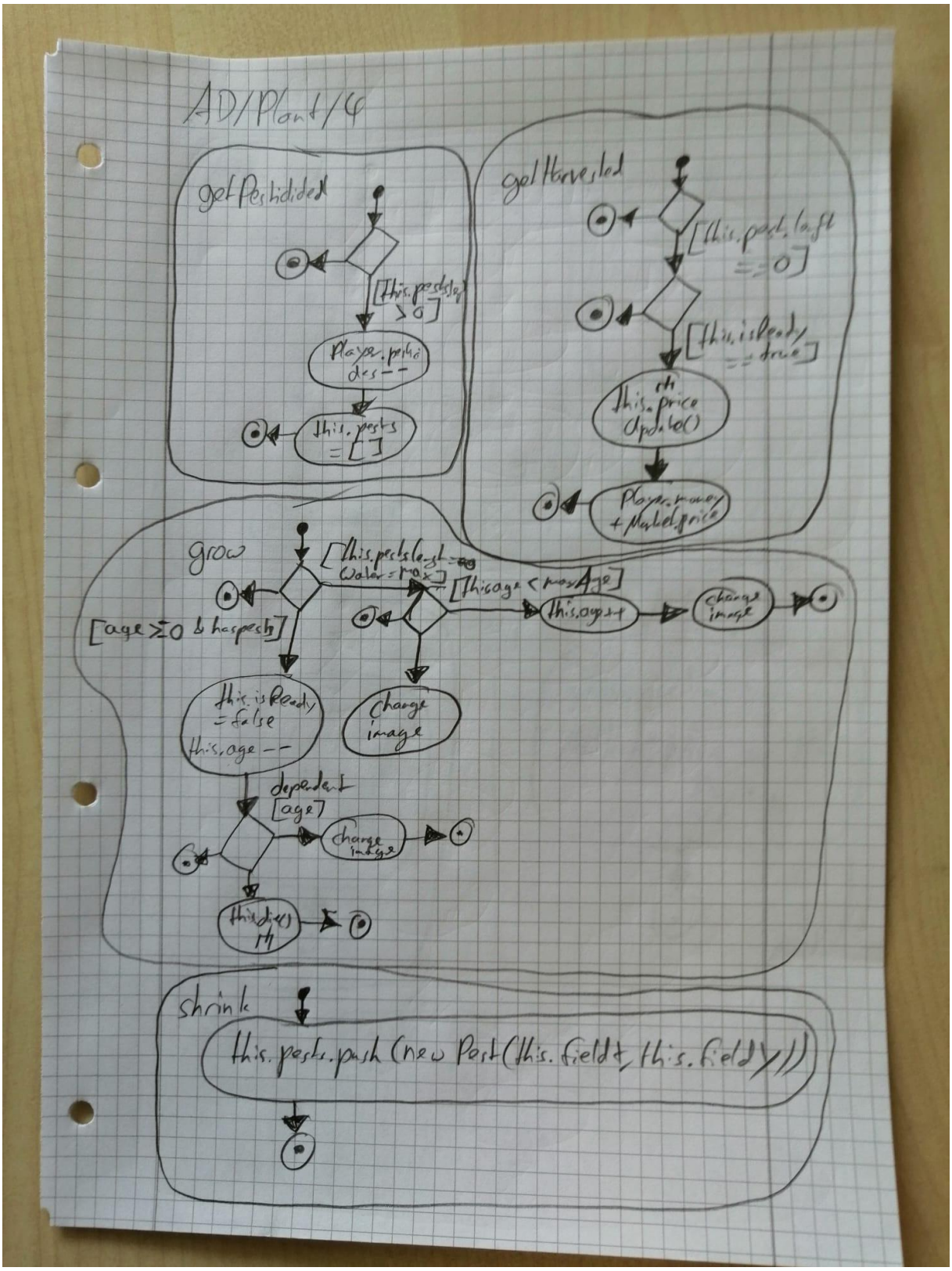

AD/Player/2
plant
_value:string
[_value = "scarf"]
Player.plantAction = PLANTACTION.SCARF
[_value = "pillow"]
Player.plantAction = PLANTACTION.PILLOW
[_value = "teddy"]
Player.plantAction = PLANTACTION.TEDDY
[_value = "blanked"]
Player.plantAction = PLANTACTION.BLANKED
[_value = "sock"]
Player.plantAction = PLANTACTION.Sock
Player.action = ACTION.PLANT
Cursor idle change
export enum PLANTACTION
PILLOW,
TEDDY,
SCARF,
BLANKED,
SOCK

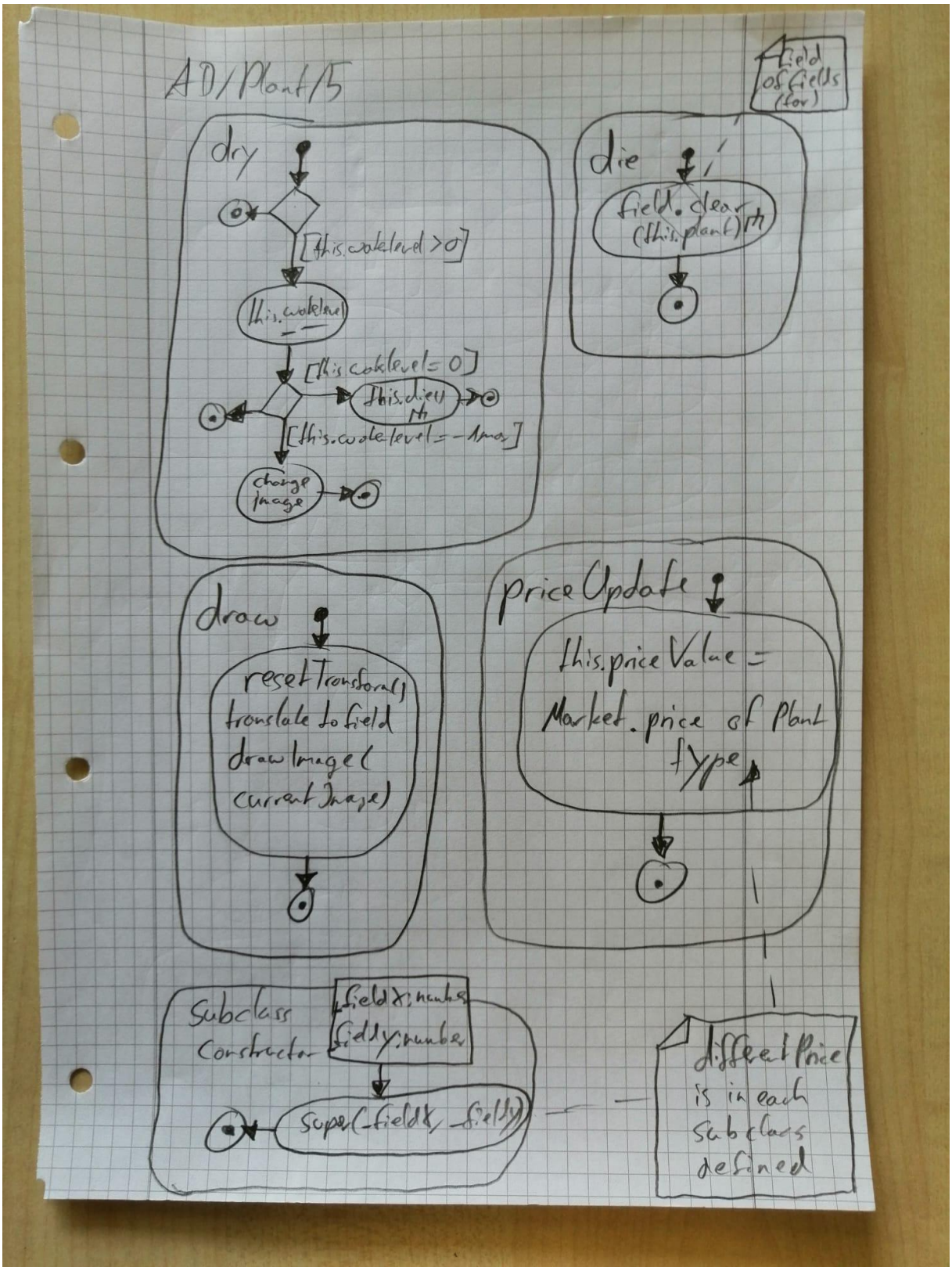

AD/Plant/1
in subclasses
priceUpdate
Constructor
_fieldX:number
_fieldY:number
this.fieldX = _fieldX
this.fieldY = _fieldY
export enum TIMEACTION
DRY,
GROW,
PEST
timeUpdate
_action:TIMEACTION
[_action = TIMEACTION.GROW]
this.grow()
[_action = TIMEACTION.DRY]
this.dry()
[_action = TIMEACTION.PEST]
this.shrink()
Simulation.update()


AD/Plant/2
plant: Plant
player Update
this.plant = plant
[Player.action = ACTION.WATER]
this.getWatered()
[Player.action = ACTION.FERTILIZE]
this.getFertilized()
[Player.action = ACTION.PESTICIDE]
this.getPesticided()
[Player.action = ACTION.HARVEST]
this.getHarvested()
Simulation.update()

AD/Plant/3


getWated
[pests.length ==0]
[this.wateLevel < this.maxWateLevel]
this.wateLevel ++
[this.wateLevel = this.maxWateLevel]
this.wateLevel ++
[this.wakLevel = this.maxWateLevel]
this.dieO
change wate image
getFertilized
[pests.length==0]
[this.fertilizeLevel < this.maxFertilizeLevel]
this.fertilizeLevel ++
[this.fertilizeLevel = this.maxFertilize]
this.agese fertilize
this.fertilizelevel ++
this.dieO
[this.fertilizelevel = this.maxfertilizelevel]
change fertilize image
Player fertelizer --


AD/Plant/G
getPesticided
[this.pests.length > 0]
Player.points -= 5
this.pests = []
getHarvested
[this.pest.left == 0]
[this.isReady == true]
this.price.update()
Player.money += Math.floor(price)
grow
[this.pests.length == 0]
[this.age < maxAge]
this.age++
changeImage
[age >= 0 && haspests]
this.isReady = false
this.age--
changeImage
dependent [age]
changeImage
this.die()
shrink
this.pests.push(new Pest(this.field, this.field))


AD/Plant/5
field of fields (for)
dry
[this.cooklevel > 0]
this.cooklevel --
[this.cooklevel = 0]
this.die()
[this.cooklevel = -1max]
change image
die
field.clear (this.plant)th
draw
resetTransform()
translate to field
drawImage(current Image)
priceUpdate
this.priceValue = Market.price of Plant type
Subclass Constructor
fieldX:number
fieldY:number
super(_fieldX, _fieldY)
different Price is in each subclass defined