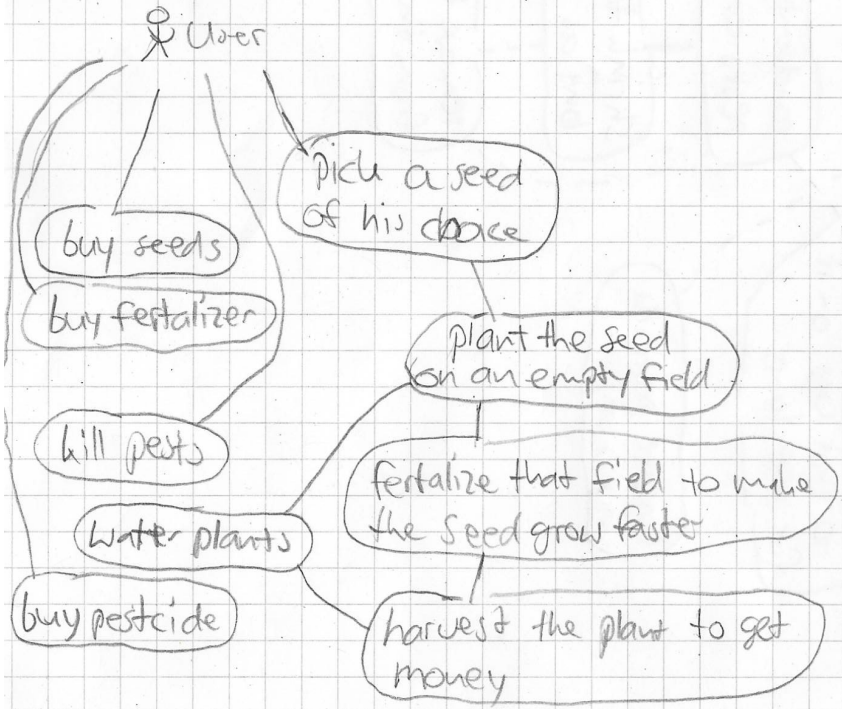
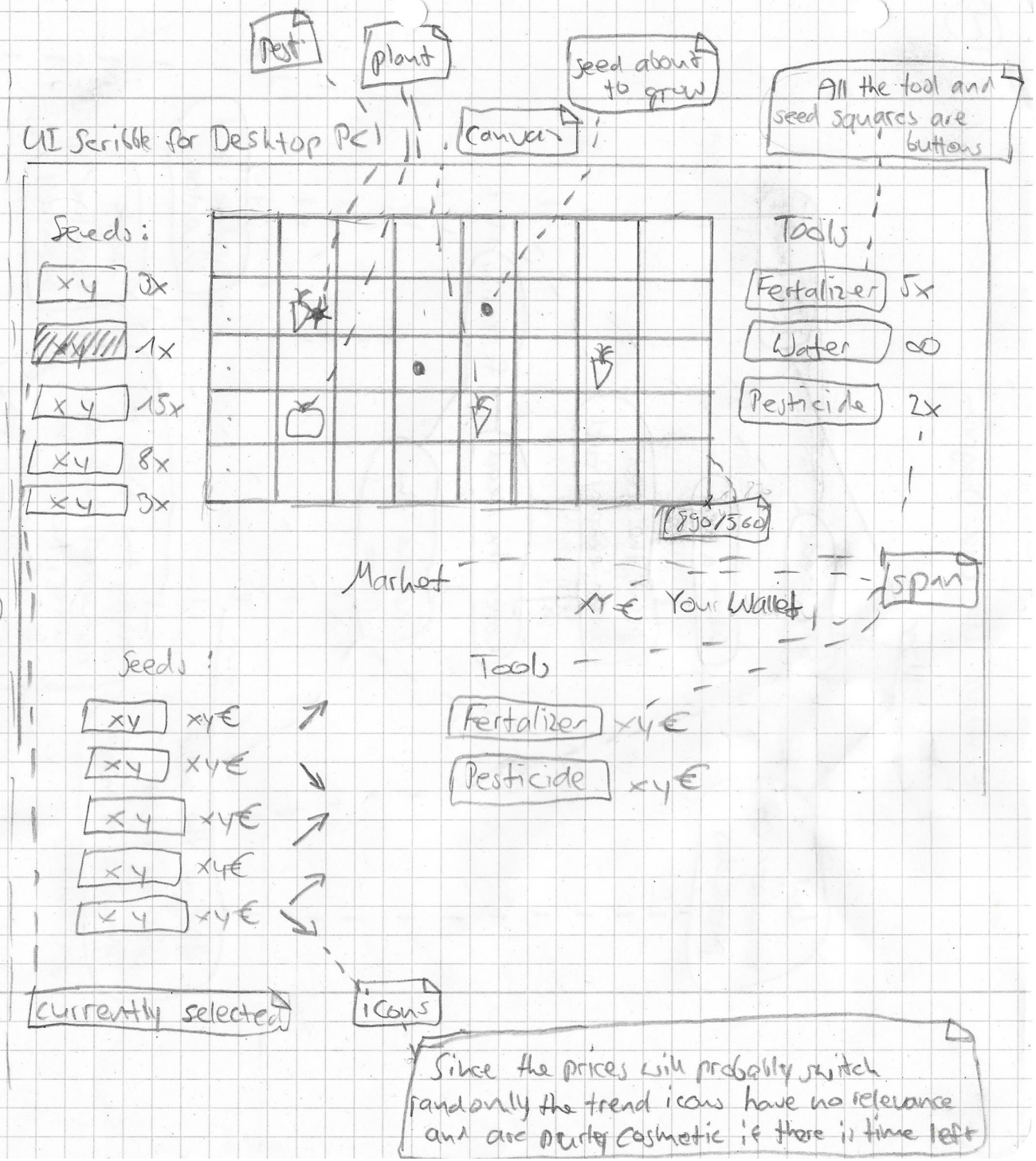
EIA 2 Endabgabe 1. Konzept

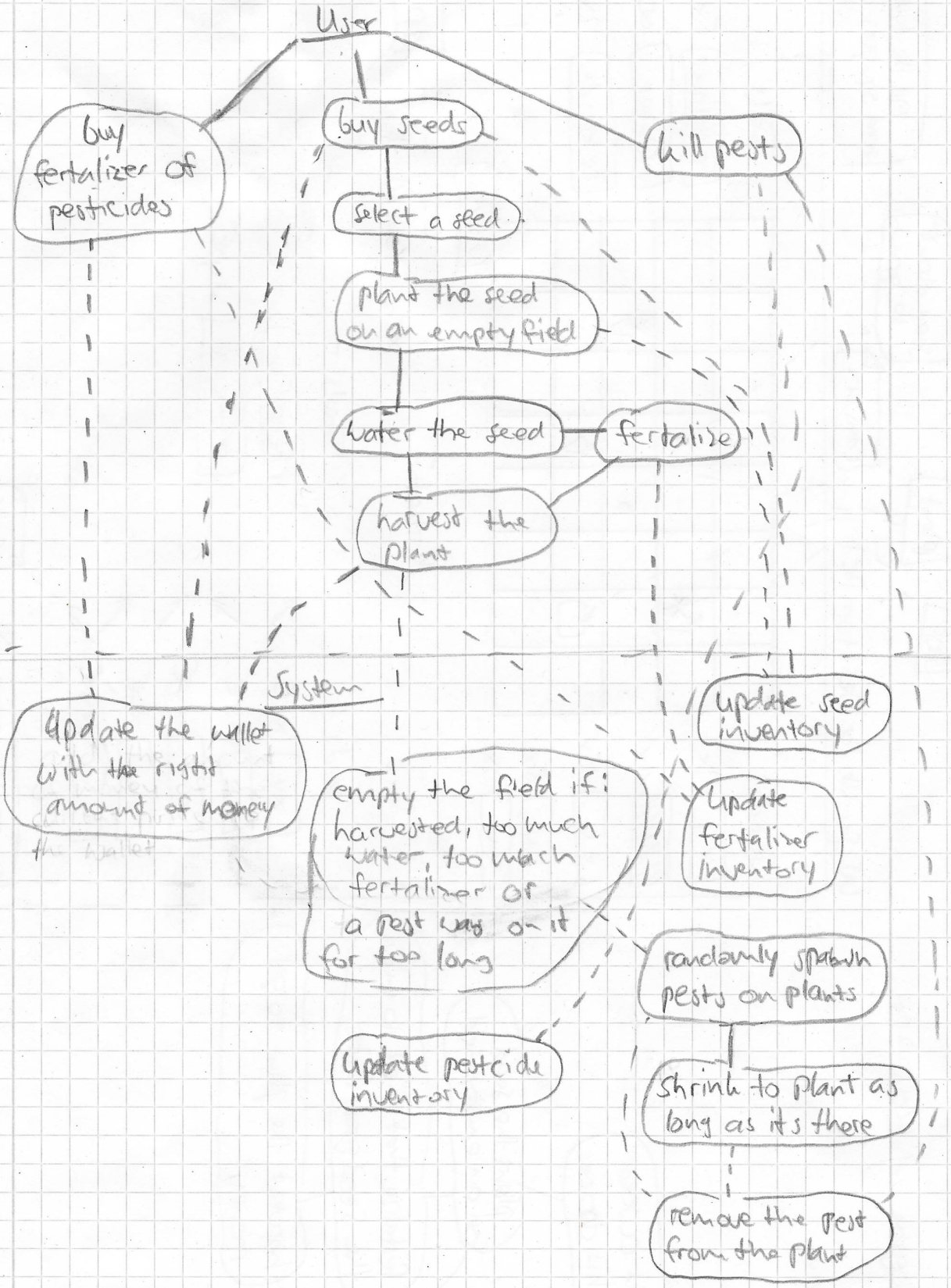
Use Case Diagram



UI Scribble for Desktop PC



Extended Use Case Diagram

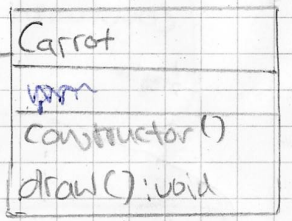


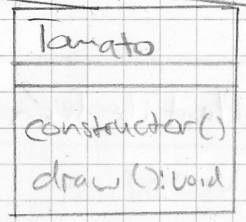
(3)

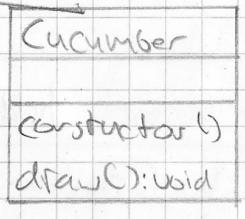
the price needs to be static so it can be accesed with without an object of that class

First class diagram

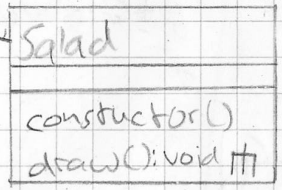
Also not sure yet how I will implement it

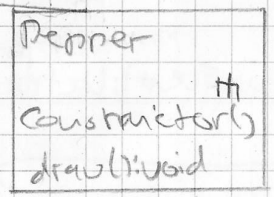




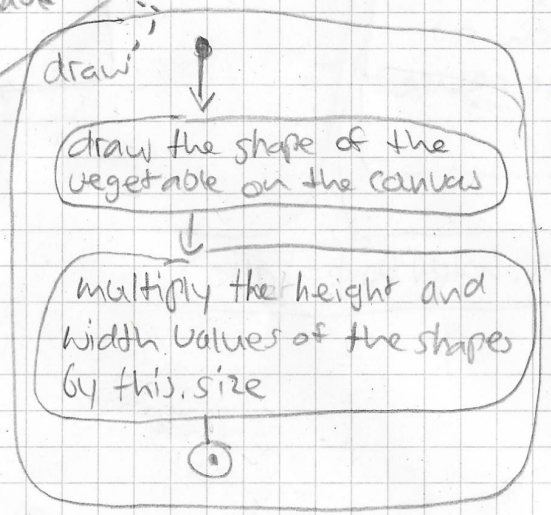


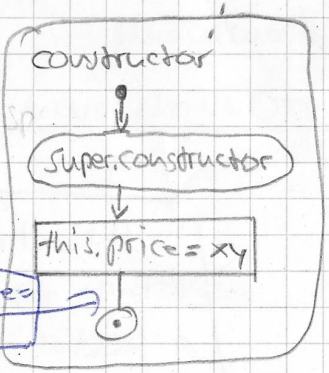
example constructor for all plant-sub-classes

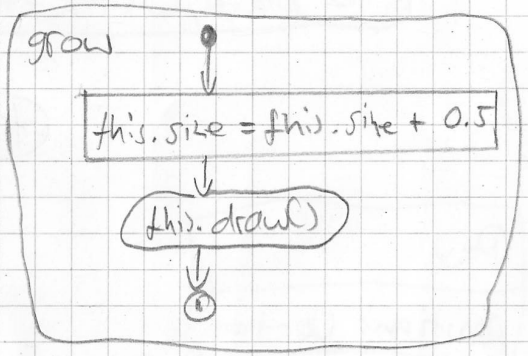


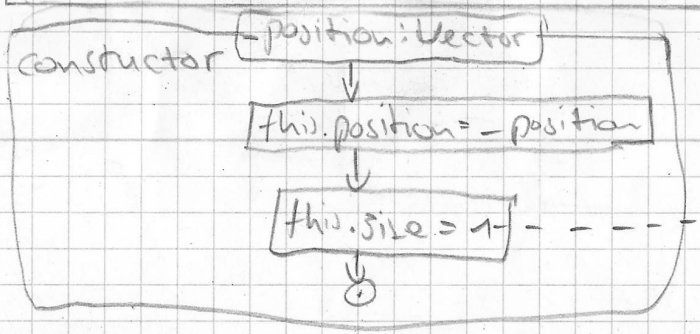


the draw method is the same for all vegetables, they just have diffrent shapes



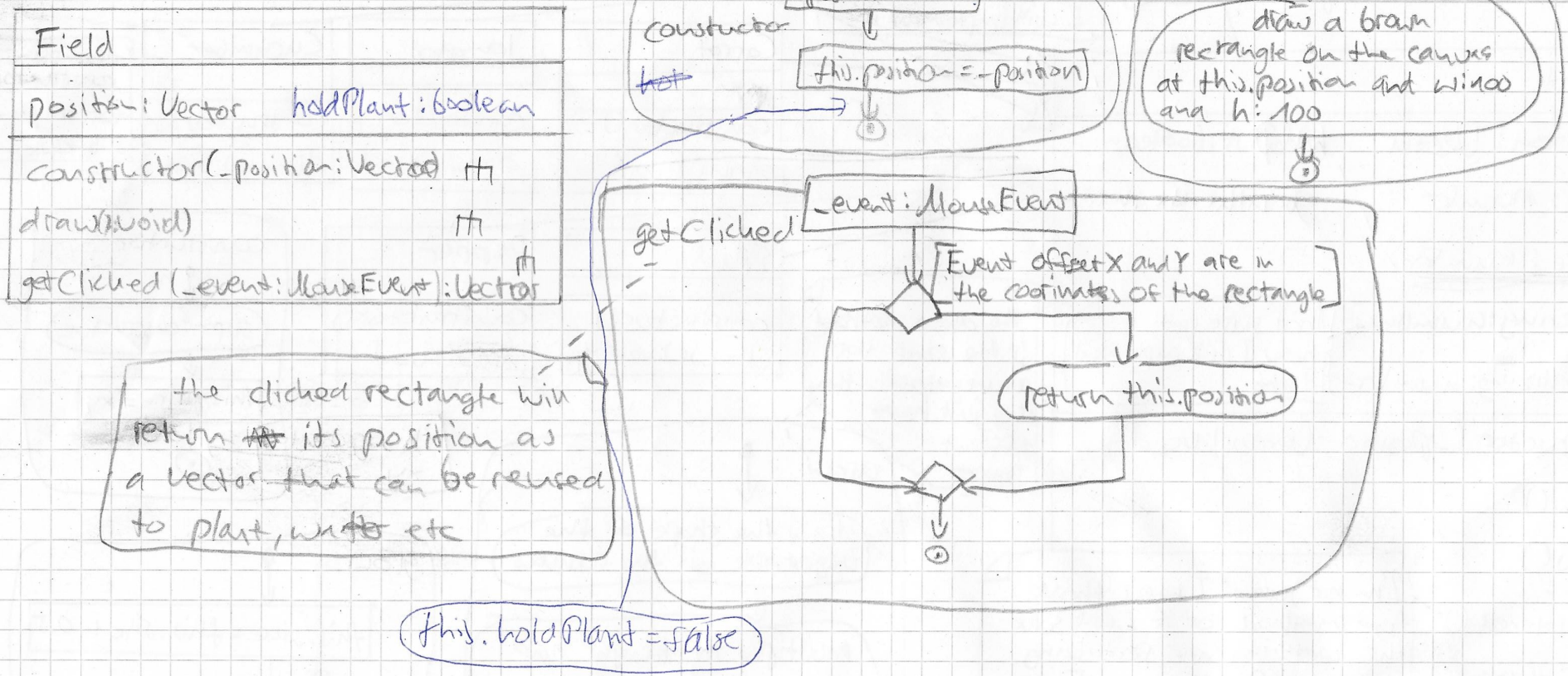


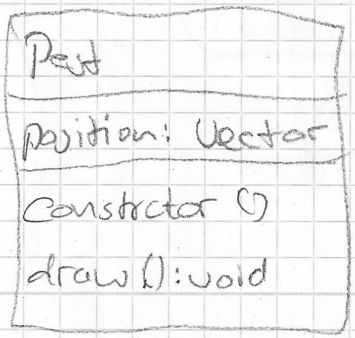




when a plant gets planted the size is always 1

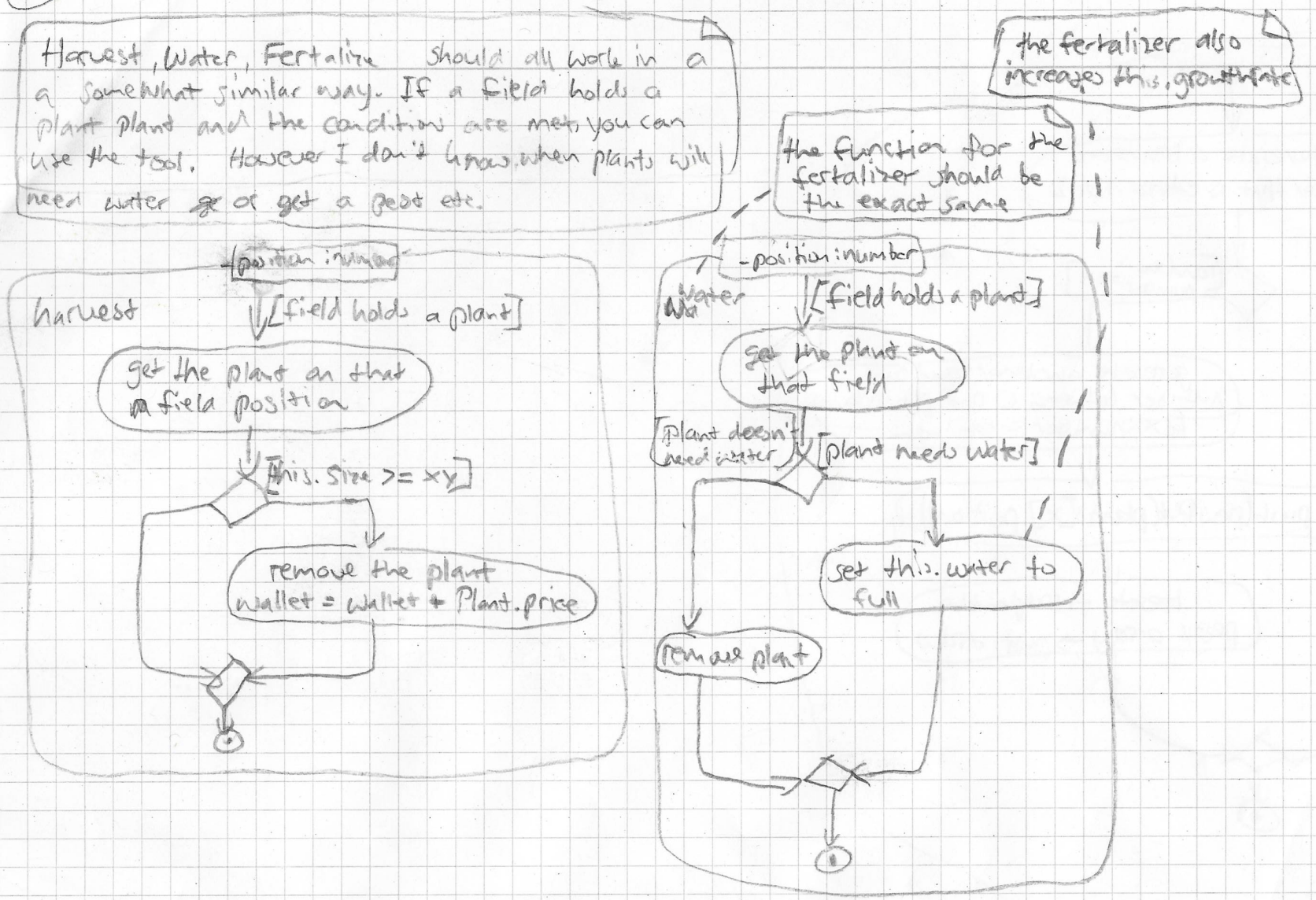
(4)


Field
position: Vector
holdPlant: boolean
constructor(_position:Vector)
draw():void
getClicked(_event:MouseEvent):Vector
constructor
_position:Vector
this.position = _position
draw
draw a brown rectangle on the canvas at this.position and w:100 and h:100
getClicked
_event:MouseEvent
Event offsetX and Y are in the coordinates of the rectangle
return this.position
the clicked rectangle will return its position as a vector that can be reused to plant, water etc
this.holdPlant = false


Pest
position: Vector
constructor()
draw():void

5

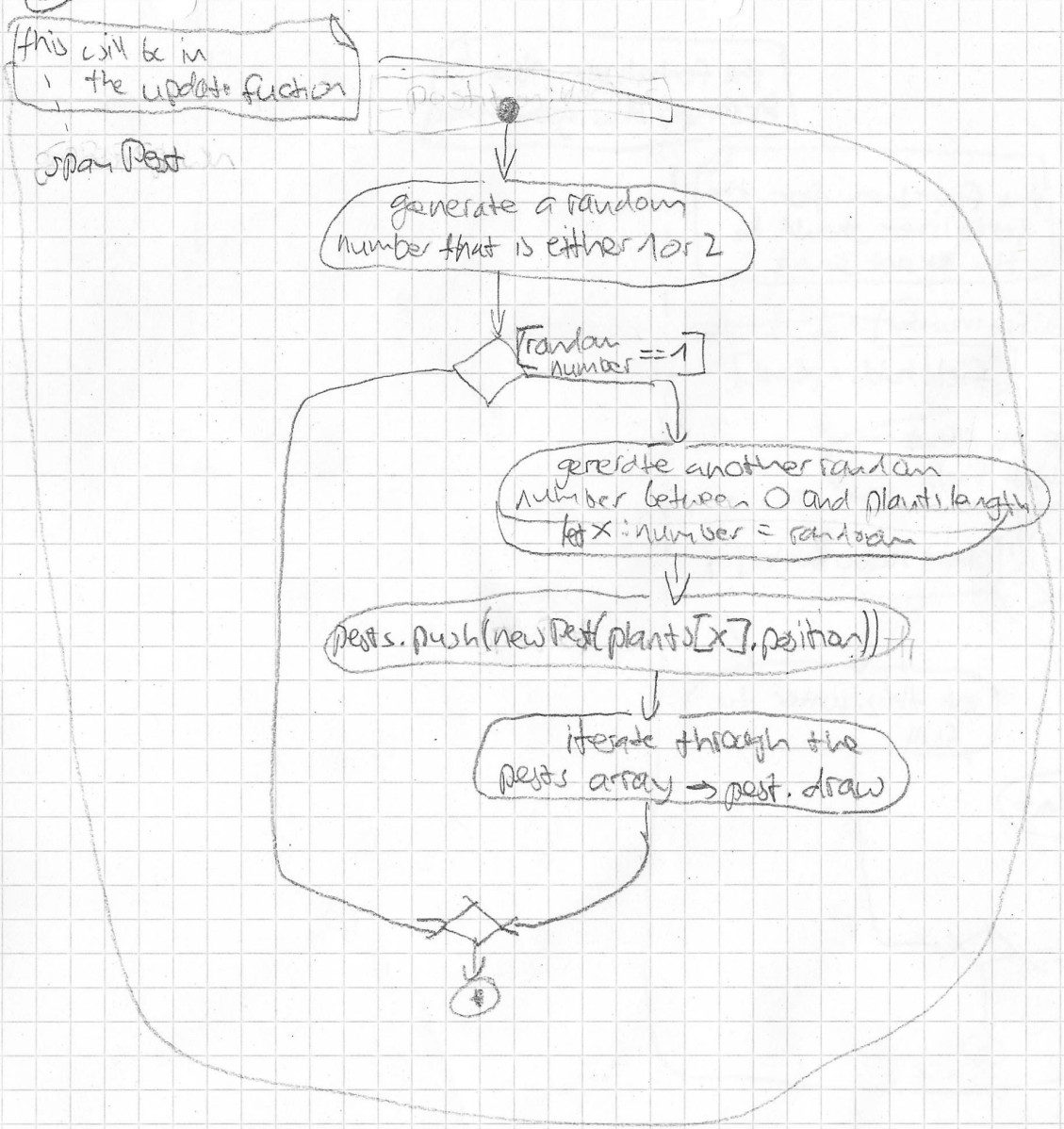
Harvest, Water, Fertalize should all work in a a somewhat similar way. If a field holds a plant plant and the conditions are met, you can use the tool. However I don't know when plants will need water or get a boost etc.



6

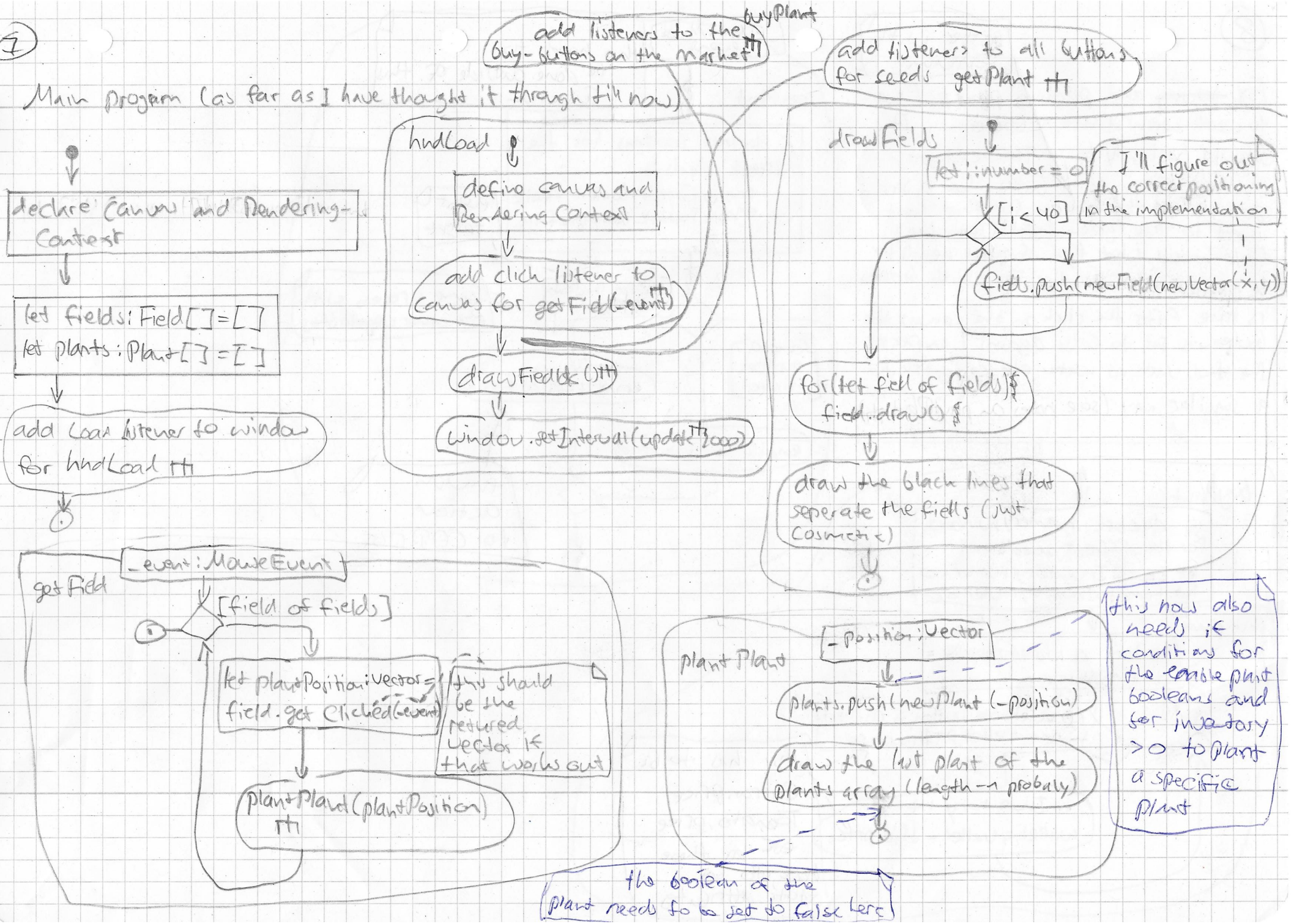
this will be in the update fuction

spawnPest



7

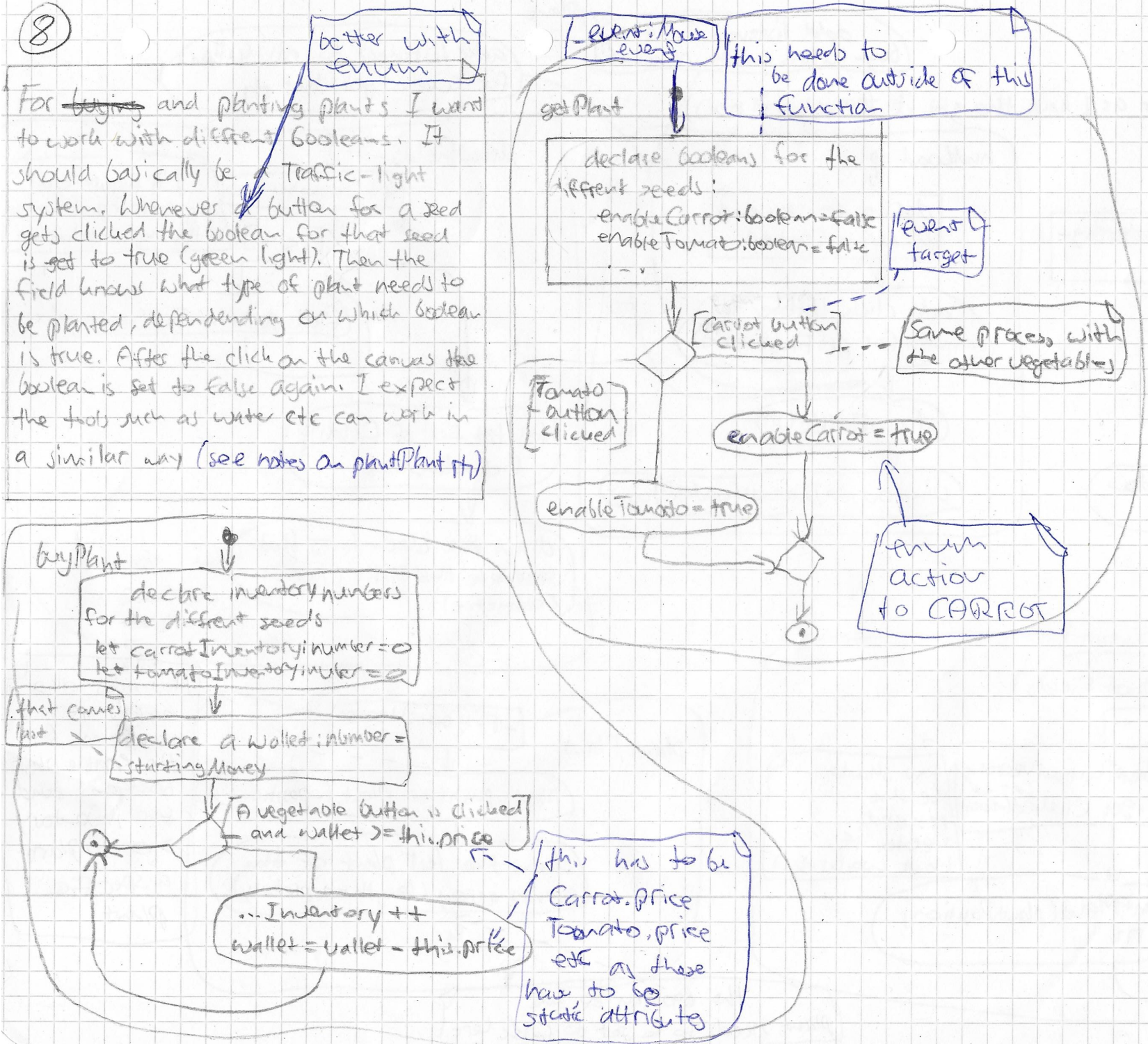
Main Program (as far as I have thought it through till now)



8

better with enum

For ~~buying~~ and planting plants I want to work with different booleans. It should basically be a Traffic-light system. Whenever a button for a seed gets clicked the boolean for that seed is set to true (green light). Then the field knows what type of plant needs to be planted, depending on which boolean is true. After the click on the canvas the boolean is set to false again. I expect the fruits such as water etc can work in a similar way (see notes on plantPlant rn)



9

Überarbeitetes Konzept

Plant

+ price: number

+ Position: Vector
+ size: number
+ growrate: number
+ name: string
+ holdsPest: boolean
+ water: number
+ fertalized: boolean

constructor (_position: Vector)

+ drawHarvestIndicator(_position: Vector): void
+ draw(): void
+ grow(): void
+ recievePest(): void

Carrot

+ price = Math.random() + 1

constructor (_position: Vector)
+ draw(): void

Tomato

+ price: number = Math.random() + 1

constructor (_position: Vector)
+ draw(): void

Salad, Cucumber, Pepper

All Plant subclasses look the same

example constructor for subclasses

constructor
_position: Vector
super.constructor(_position)
this.growthrate = xy
this.name = "..."
this.water = xy

draw
draw the shape of the plant and multiply the coordinates by this.size

grow
this.size = this.size + this.growrate

Constructor
_position: Vector
this.position = _position
this.size = 1
this.holdsPest = false
this.fertalized = false

drawHarvestIndicator
_position: Vector
draw a golden rectangle at the right position to show that the plant is ready to be harvested

10

Field

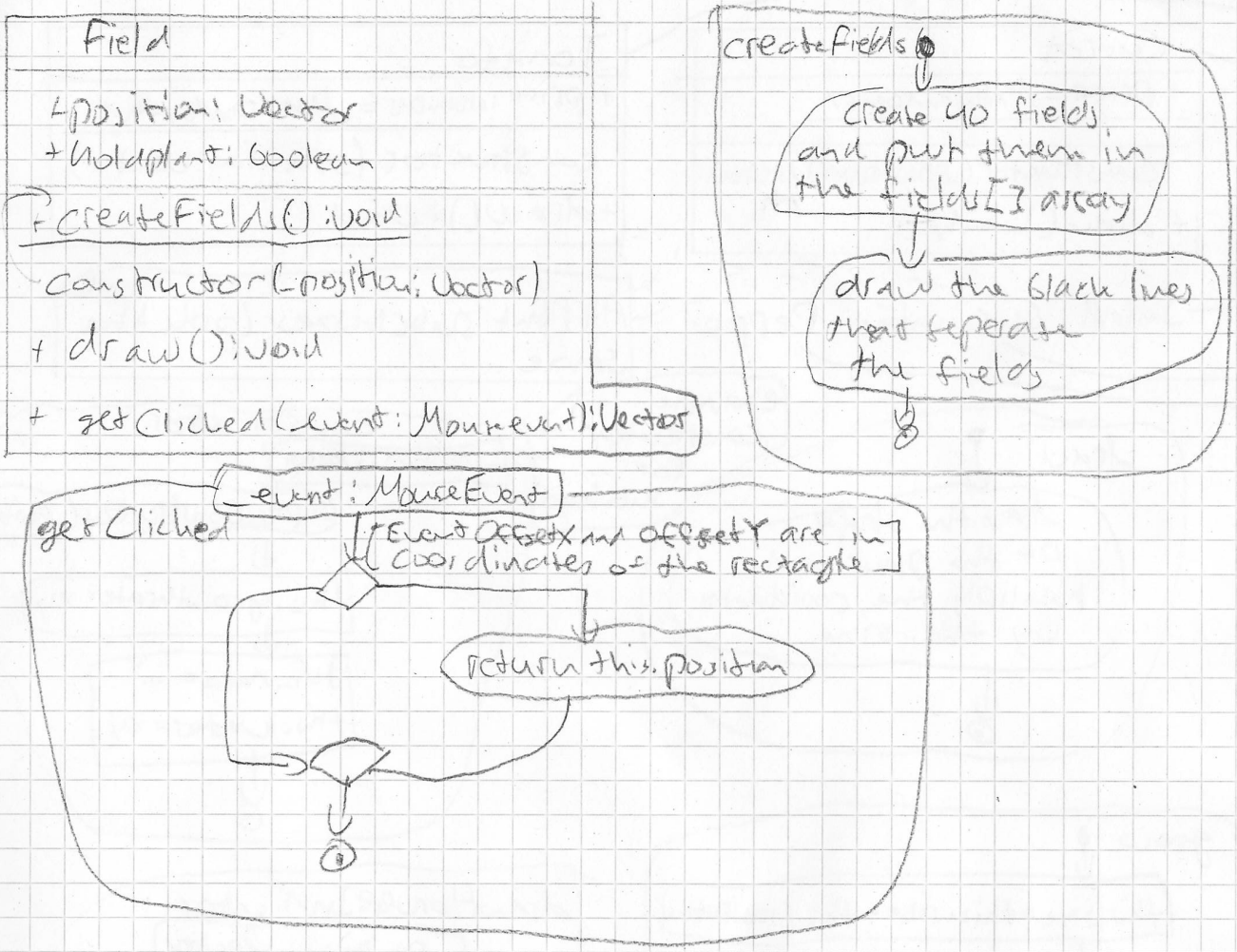
-position: Vector
+holdsplant: boolean

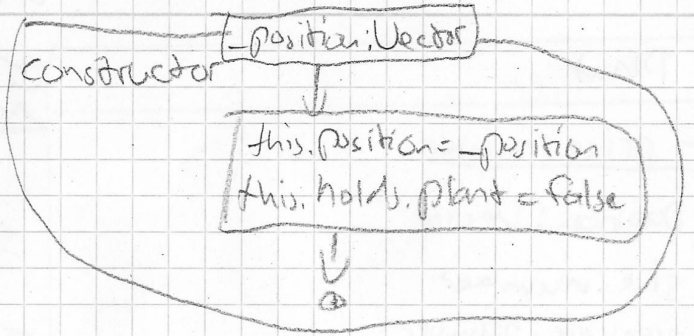
+createFields(): void

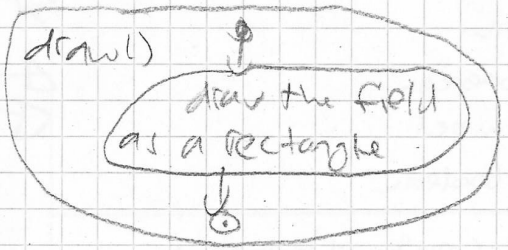
-constructor(_position: Vector)

+draw(): void

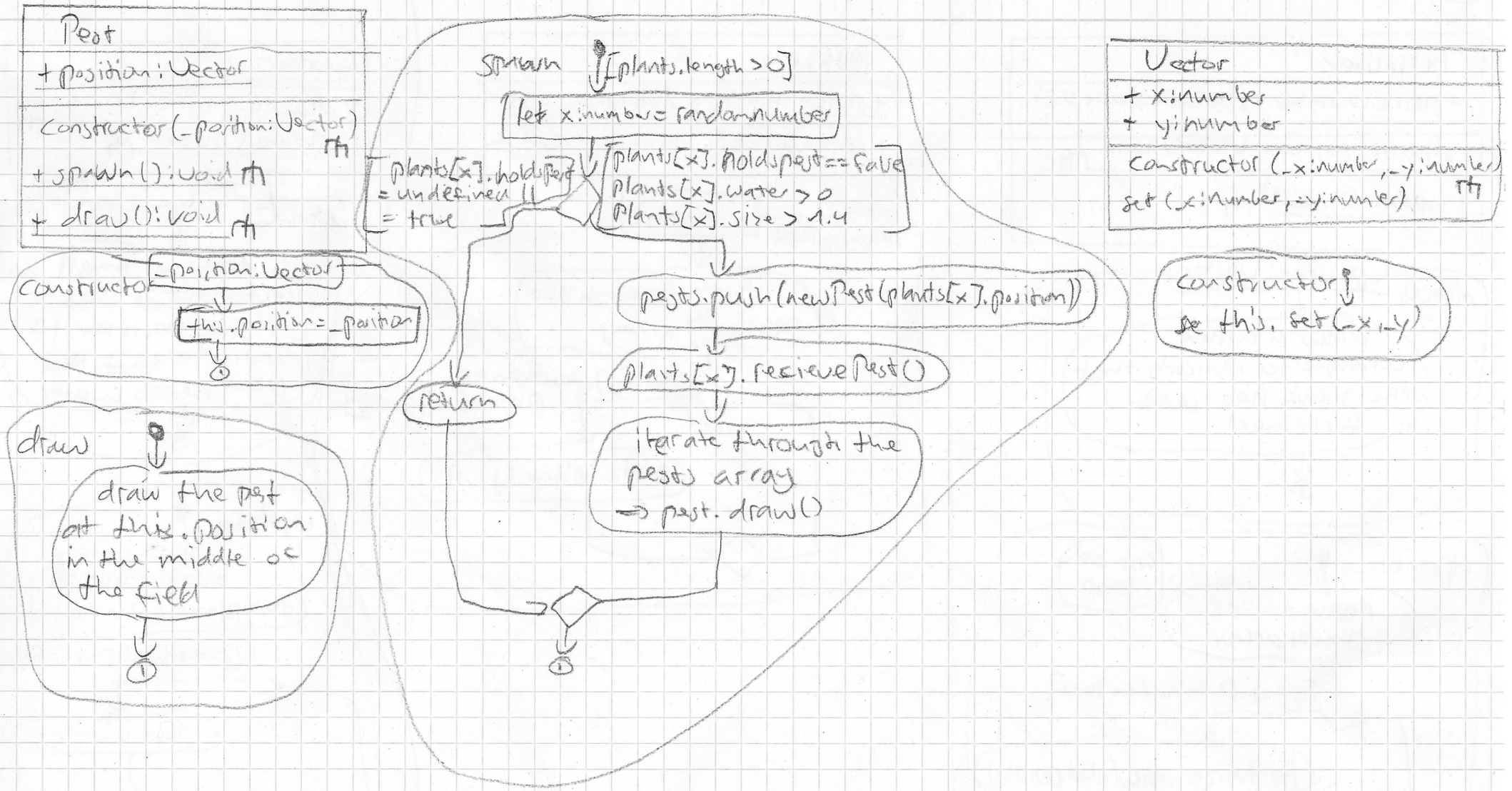
+getClicked(event: MouseEvent): Vector





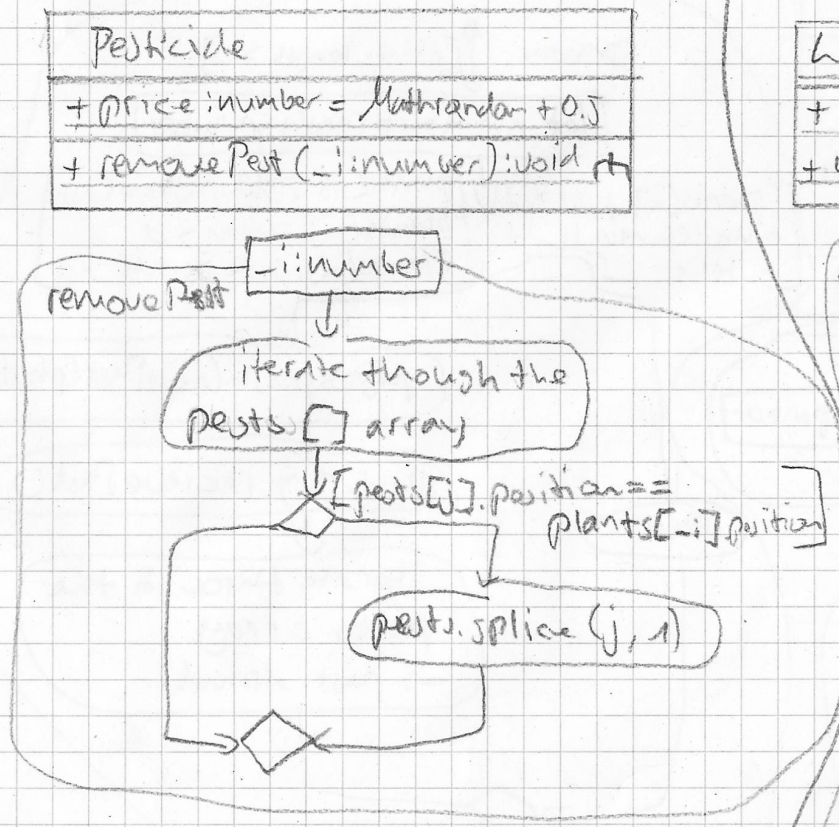


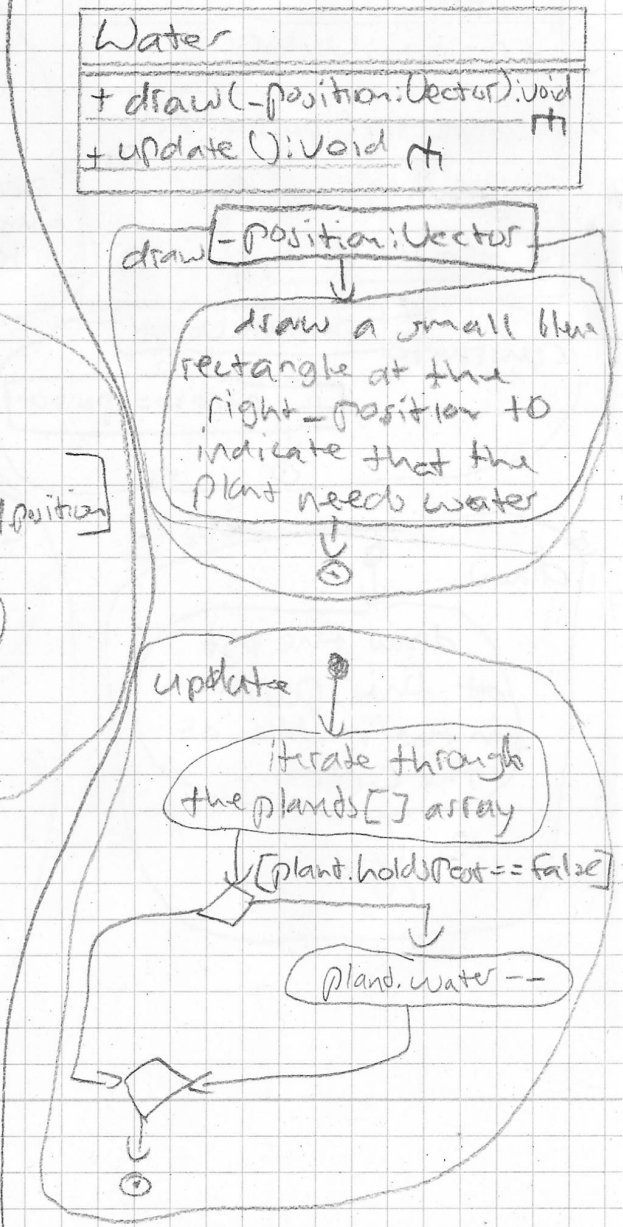
11


Pest
+position: Vector
constructor(_position:Vector)
+spawn():void
+draw():void
constructor
_position:Vector
this.position=_position
draw
draw the pest at this.position in the middle of the field
spawn
[plants.length > 0]
let x:number= randomNumber
plants[x].holdsPest = undefined || = true
plants[x].holdsPest == false
plants[x].water > 0
plants[x].size > 1.4
return
pests.push(new Pest(plants[x].position))
plants[x].recievePest()
iterate through the pests array
-> pest.draw()
Vector
+ x:number
+ y:number
constructor(_x:number,_y:number)
set(_x:number,_y:number)
constructor
se this.set(_x,_y)

17


Fertalizer
+price :number = (Math.random() + 0.5)
+draw(_position: Vector): void
+update(): void
draw
_position: Vector
draw a white rectangle to indicate that the plant has been fertalized
update
iterate through the plants array
for of loop
[plant.fertalized == true]
Fertalizer.draw(plant.position)


Pesticide
+price :number = Math.random() + 0.5
+removePest(_i: number): void
removePest
_i: number
iterate through the pests[] array
[pests[j].position == plants[_i].position]
pests.splice(j, 1)


Water
+draw(_position: Vector): void
+update(): void
draw
_position: Vector
draw a small blue rectangle at the right _position to indicate that the plant needs water
update
iterate through the plants[] array
[plant.holdsPest == false]
plant.water--

13

Market classes

Market

+ buy(_event : MouseEvent) : void
+ updatePrices() : void

Inventory

+ carrotAmount : number = 0
+ tomatoAmount : number = 0
+ cucumberAmount : number = 0
+ salladAmount : number = 0
+ pepperAmount : number = 0
+ fertalizerAmount : number = 0
+ pesticideAmount : number = 0

+ update() : void

Wallet

+ money : number = 0

+ update() : void

buy

_event : MouseEvent

_event.target = document.querySelector(#buyCarrots)
Wallet.money >= Carrot.price

this goes for all vegetables if the event.target is their button. In the example here, the carrots are clicked

Inventory.carrotAmount++

Wallet.money = Wallet.money - Carrot.price

updatePrices

Carrot.price = Math.random() * pVariationNum + 1
Tomato.price = Math.random() * pVariationNum + 1
etc. for all vegetables...
Fertalizer.price = Math.random * pVariationNum + 0.5
Pesticide.price = Math.random * pVariationNum + 0.5

document.querySelector("#carrotprice").innerHTML =
Carrot.price.toFixed(2) + "€"
etc for all vegetables and tools

Fill in the HTML spans where the prices show

update

document.querySelector("#inventory").innerHTML =
"Your Inventory" +
Inventory.carrotAmount + "x" +
Inventory.tomatoAmount + "x" +
Inventory.cucumberAmount + "x" +
Inventory.salladAmount + "x" +
Inventory.pepperAmount + "x" +
Inventory.fertalizerAmount + "x" +
Inventory.pesticideAmount + "x"

update

document.querySelector("#wallet").innerHTML = "Your Wallet" +
Wallet.money.toFixed(2) + "€"

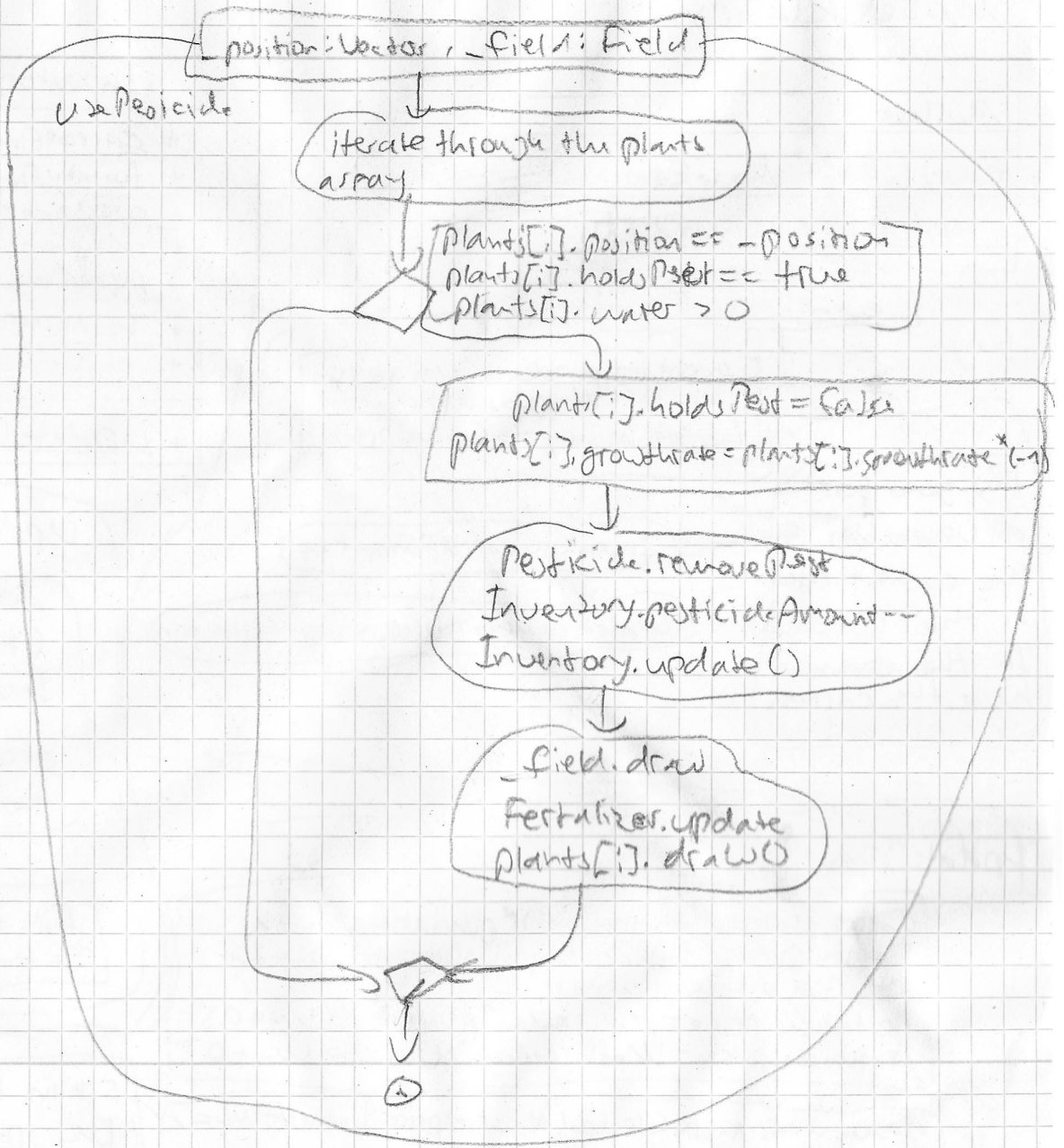
Fill in the HTML spans for the inventory

14

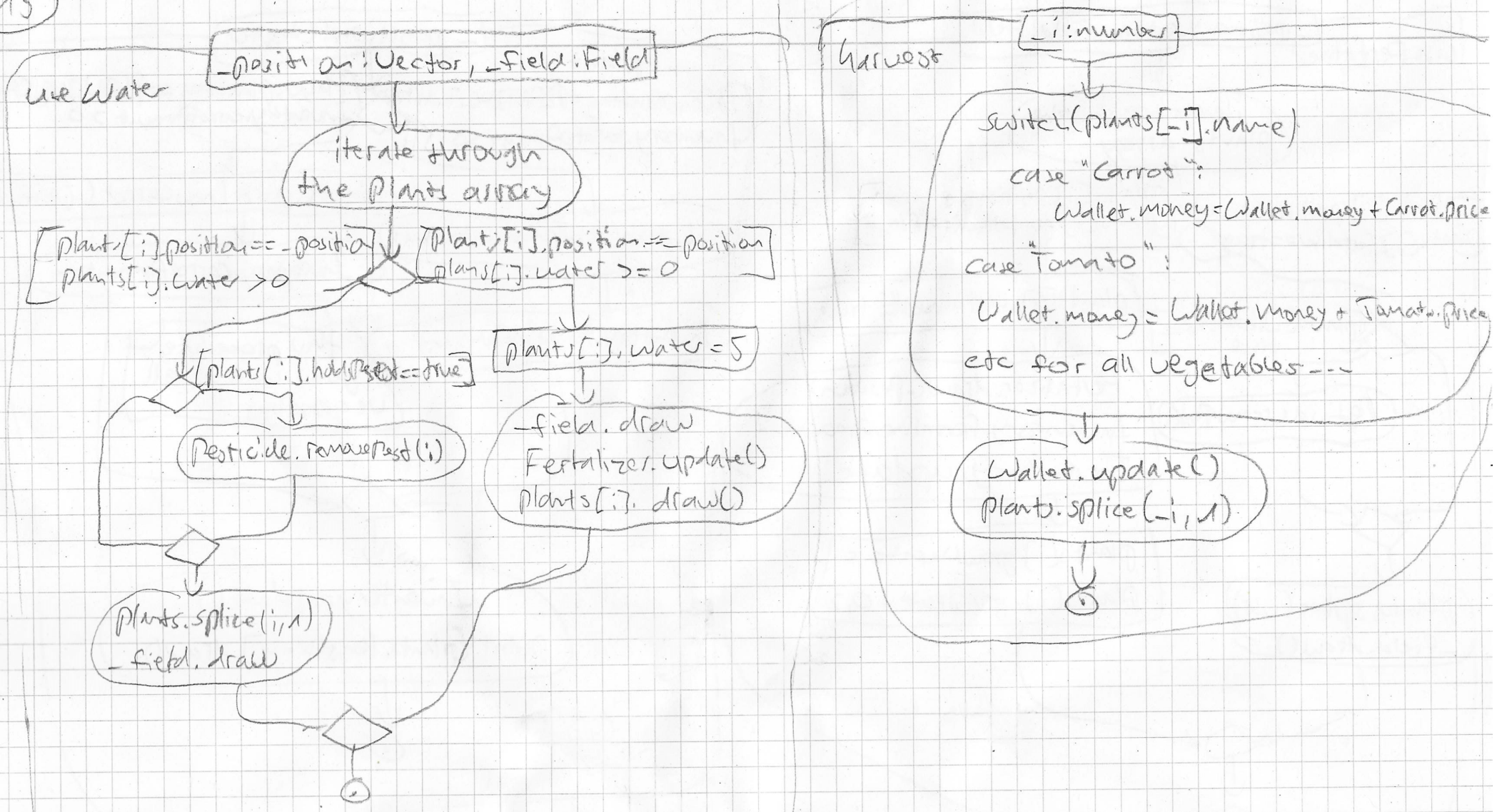
enum TOOLACTION
- FERTALIZE
- HARVEST
- WATER
- PLANT
- PESTICIDE

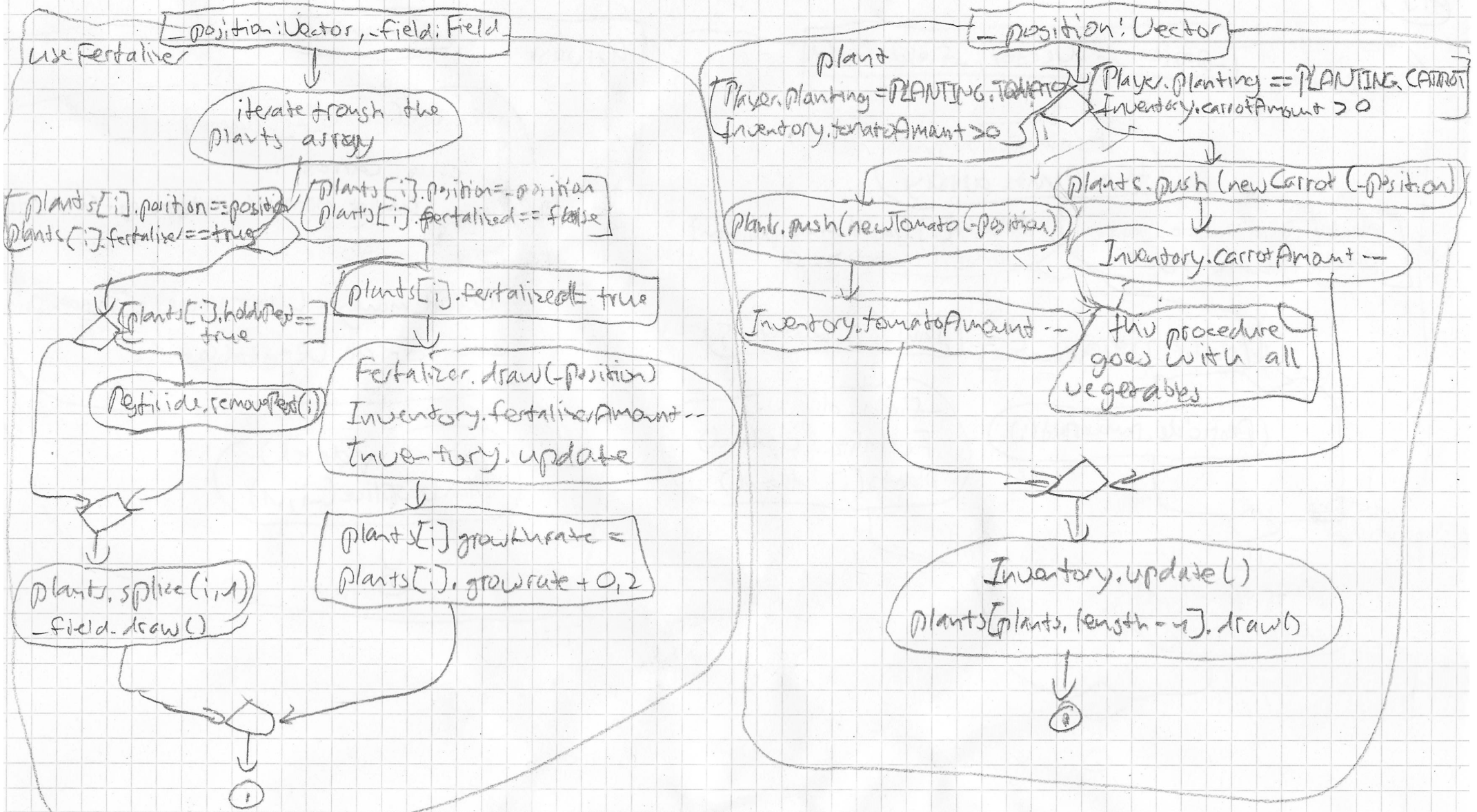
enum PLANTING
- CARROT
- TOMATO
- CUCUMBER
- SALAD
- PEPPER

Player
+ toolAction : TOOLACTION + planting : PLANTING
- usePesticide(_position:Vector, _field:Field):void + useWater(_position:Vector, _field:Field):void + useFertalizer(_position:Vector, _field:Field):void + harvest(_i:number):void() - plant(_position:Vector):void()



15


useWater
_position:Vector, _field:Field
iterate through the plants array
plants[i].position==_position
plants[i].water>=0
plants[i].position==_position
plants[i].water>=0
plants[i].water=5
plants[i].hasPest==true
Pesticide.removePest(i)
_field.draw
Fertalizer.update()
plants[i].draw()
plants.splice(i,1)
_field.draw
harvest
_i:number
switch(plants[_i].name)
case "Carrot":
Wallet.money=Wallet.money+Carrot.price
case "Tomato":
Wallet.money=Wallet.money+Tomato.price
etc for all vegetables---
Wallet.update()
Plants.splice(_i,1)


_position:Vector, _field:Field
useFertalizer
iterate trough the plants array
plants[i].position==position
plants[i].fertalized==true
plants[i].position=_position
plants[i].fertalized==false
plants[i].holdsPest==true
plants[i].fertalized=true
Pestilide.removePest(i)
Fertalizer.draw(_position)
Inventory.fertalizerAmount--
Inventory.update
plants[i].growthrate=plants[i].growrate+0,2
plants.splice(i,1)
_field.draw()
1
_position:Vector
plant
Player.planting=PLANTING.TOMATO
Inventory.tomatoAmount>0
Player.planting==PLANTING.CARROT
Inventory.carrotAmount>0
plants.push(newTomato(_position))
plants.push(newCarrot(_position))
Inventory.tomatoAmount--
Inventory.carrotAmount--
the procedure goes with all vegerables
Inventory.update()
plants[plants.length-1].draw()
1

17

Main Program

Variablen:

export let fieldCanvas: HTMLCanvasElement
export let ctxC: CanvasRenderingContext2D
let currentUse: HTMLSpanElement
let formData: FormData
let moneyString: string
let pVariationString: string
export let pVariationNum: Number
export let fields: Field[] = []
export let plants: Plant[] = []
export let posts: Post[] = []

window.

add Load listener to window for hndLoad ††

hndLoad

get the value from formData("variation") and save it in pVariationNum

add change listener to settings hndSettingChange †† ††
add click listener to start button hndSimulationLoad

hndSettingsChange

get the value from formData("money") and save it in moneyString

get the value from formData("variation") and save it in pVariationNum

Field.createFields
window.setInterval(update, 3000)

add click Listener to the canvas for getAction ††

hndSimulationLoad

set #format and #market visible and delete the settings-elements

Wallet.money = parseInt(moneyString)

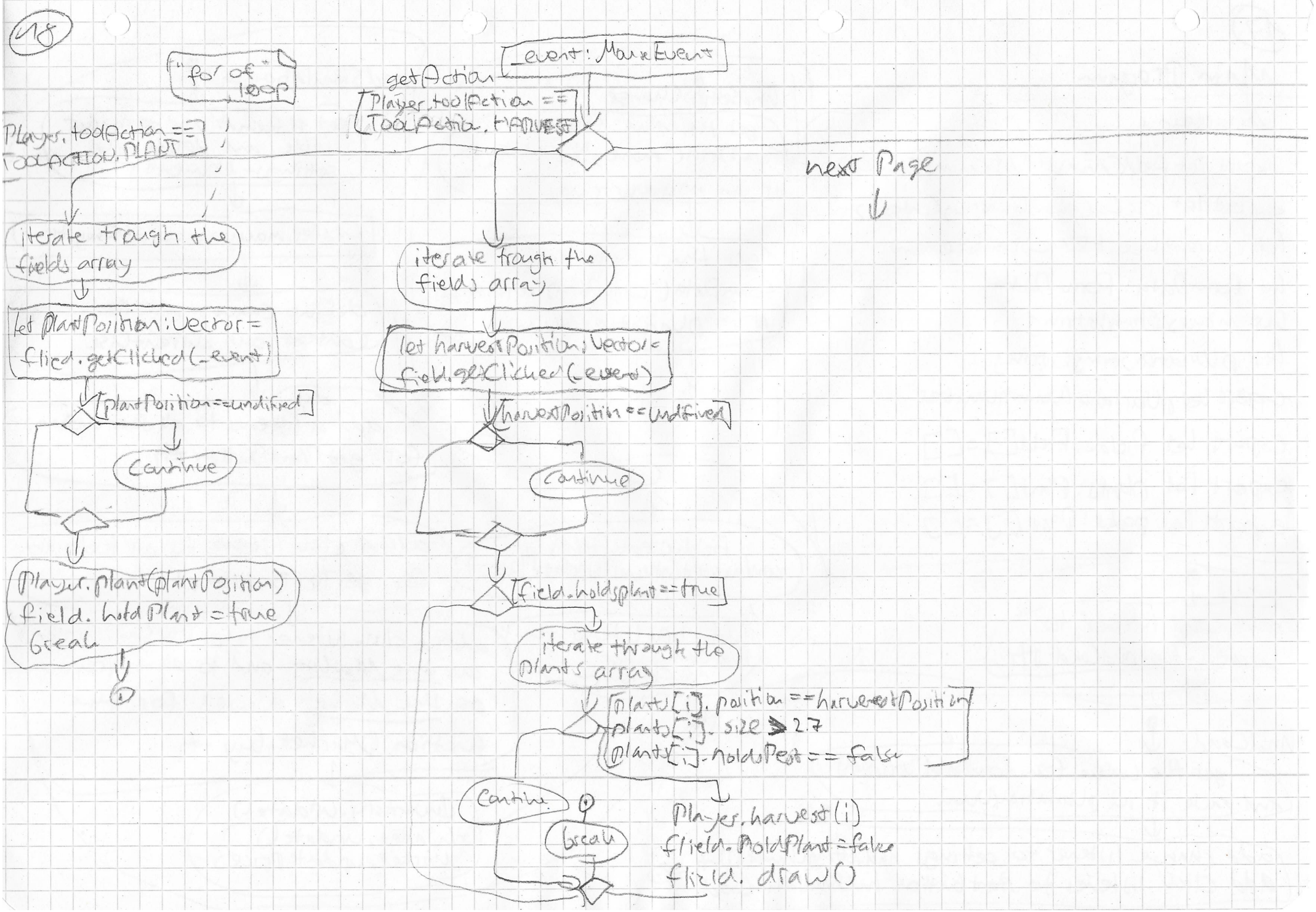
define canvas and RenderingContext and currentUse

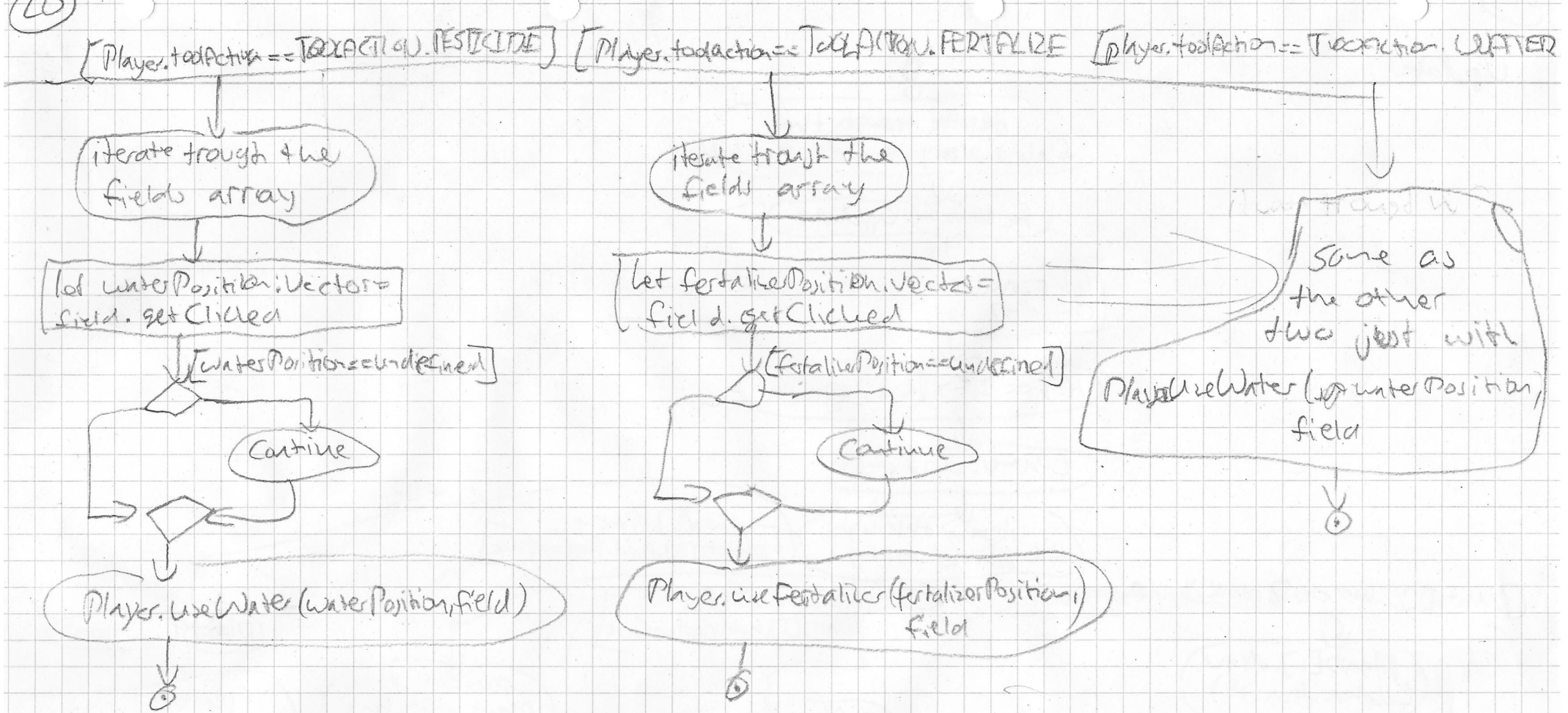
click
add a listener to all seedbuttons for getPlantButton ††

add a click listener to all toolButtons for getToolButton ††

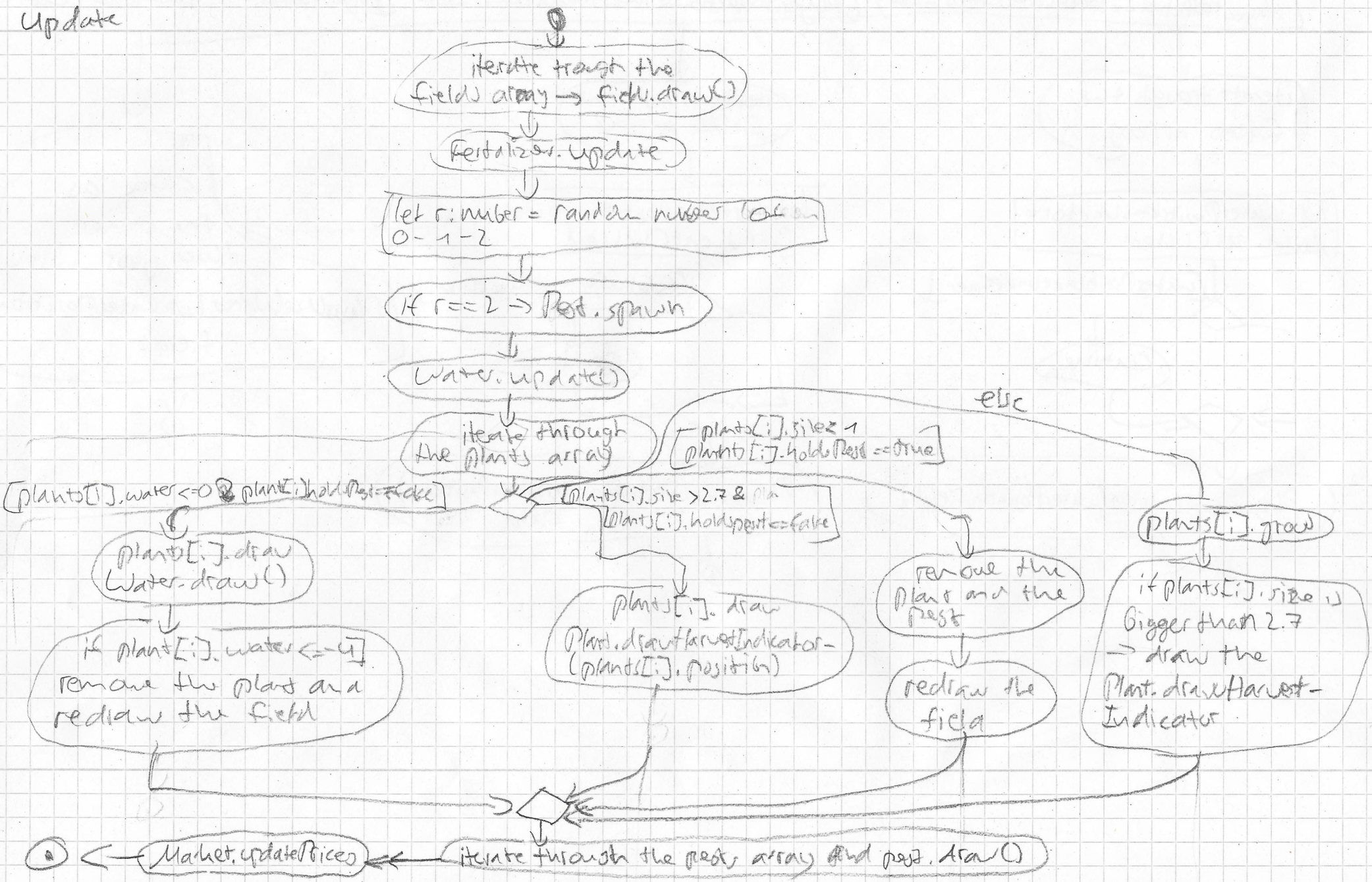
add a click listener to all the seedbuttons on the Market and to all toolbuttons on the Market as well for the function Market.buy ††

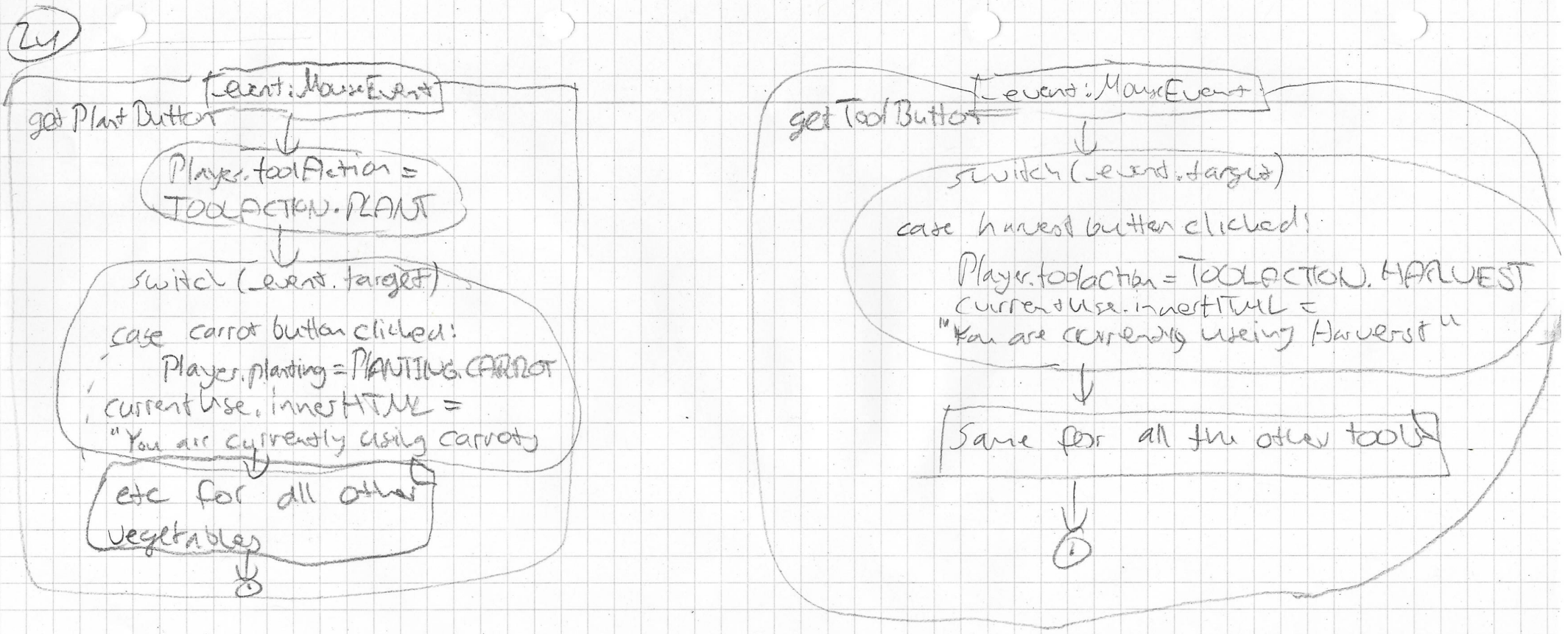
Inventory.update()
Wallet.update()
Market.updatePrices()


"for of" loop
getAction
_event: MouseEvent
Player.toolAction == ToolAction.HARVEST
Player.toolAction == TOOLACTION.PLANT
next Page
iterate trough the fields array
let plantPosition: Vector = field.getClicked(_event)
plantPosition==undifined
continue
Player.plant(plantPosition)
field.holdPlant = true
break
iterate trough the fields array
let harvestPosition: Vector = field.getClicked(_event)
harvestPosition == undifined
continue
field.holdsplant == true
iterate trough the plants array
plants[i].position == harvestPosition
plants[i].size >= 2.7
plants[i].holdsPest == false
continue
break
Player.harvest(i)
field.holdPlant = false
field.draw()


[Player.toolAction == TOOLACTION.PESTICIDE]
[Player.toolAction == TOOLACTION.FERTILIZE
[Player.toolAction == TOOLACTION.WATER
iterate trough the fields array
let waterPosition: Vector = field.getClicked
[waterPosition == undefined]
Continue
Player.useWater(waterPosition, field)
iterate trough the fields array
let fertalizerPosition: Vector = field.getClicked
[fertalizerPosition == undefined]
Continue
Player.useFertalizer(fertalizerPosition, field)
same as the other two just with
Player.useWater(waterPosition, field

Update


iterate trough the fields array -> field.draw()
fertalizer.update
let r:nuber = random number 0-1-2
if r==2 -> Pest.spawn
water.update()
iterate through the plants array
plants[i].size<1 plants[i].holdsPest==true
else
plants[i].size>2.7 & pla plants[i].holdspest==false
plants[i].water<=0 & plant[i].holdPest==false
plants[i].draw water.draw()
if plant[i].water<=0 remove the plant and redraw the field
plants[i].draw Plant.drawHarvestIndicator- (plants[i].position)
remove the plant and the pest
redraw the field
plants[i].grow
if plants[i].size is bigger than 2.7 -> draw the Plant.drawHarvest-Indicator
iterate through the pest, array and pest.draw()
Market.updatePrices()


24
getPlantButton
event: MouseEvent
Player.toolAction =
TOOLACTION.PLANT
switch (event.target)
case carrot button clicked:
Player.planting = PLANTING.CARROT
currentUse.innerHTML =
"You are currently using carrots"
etc for all other
vegetables
getToolButton
event: MouseEvent
switch (event.target)
case harvest button clicked:
Player.toolAction = TOOLACTION.HARVEST
currentUse.innerHTML =
"You are currently using Harvest"
Same for all the other tools