

UI-Scribble

Mittwoch, 29. Juni 2022 16:05

Startscreen 1. Html-Seite

Gemüsegarten

min. Schwankungen: 0 — 1

max. Schwankungen: 2 — 3

Let's go

Startkapital:

- ☐ 500
- ☒ 1000
- ☐ 1500

Annotations:

- `<h1>`
- background picture of a Farm
- `<input>` slider min
- `<input>` slider max
- `<input>` radioButton
- `<button>` class: startButton

2. Html Seite ---> Übertragung mit "get request"

Geldbeutel: 250 €

Annotations:

- `<div>` wallet
- For-Schleife generiert 40 Spans
- `<div>` Gamefield
- `` field

Geldbeutel: 250 €

T	P	L	Ca	Cu
-2€	-4€	-1€	-3€	-2€

Annotations:

- `<div>` name: "plant-buttons"
- `<button>`

Geldbeutel: 246 € → +5 €

Wachstum -1

Wachstum -3

geben -1€

fangen -3€

ernien +5€

Schlaflos -10€

Annotations:

- `<div>` name: "plant-buttons"
- `<div>` name: "plant-screens"
- shows information what the plant needs

Fields
gameDiv: HTMLDivElement recentVegetable: Vegetable infoContainer: HTMLDivElement
constructor (gameDivGiven)

Wallet
static instance: Wallet urlParam: URLSearchParams minPrice: number maxPrice: number seedMoney: number inflationRatio: number
constructor()

Vegetable
uiField: Fields growthStatus1: string growthStatus2: string damageStatus: string bugStatus: string growthTime: number neededFertilize: number fertilizePrice: number neededWater: number plantPrice: number income: number healPrice: number
constructor (-uiField, -growthStatus2, -growthTime, -neededFertilize, -fertilizePrice, -neededWater, -plantPrice, -income, -healPrice)

Tomato
static informationInstance: Tomato = new Tomato
constructor (-uiField)

Potato
static informationInstance: Potato = new Potato
constructor (-uiField)

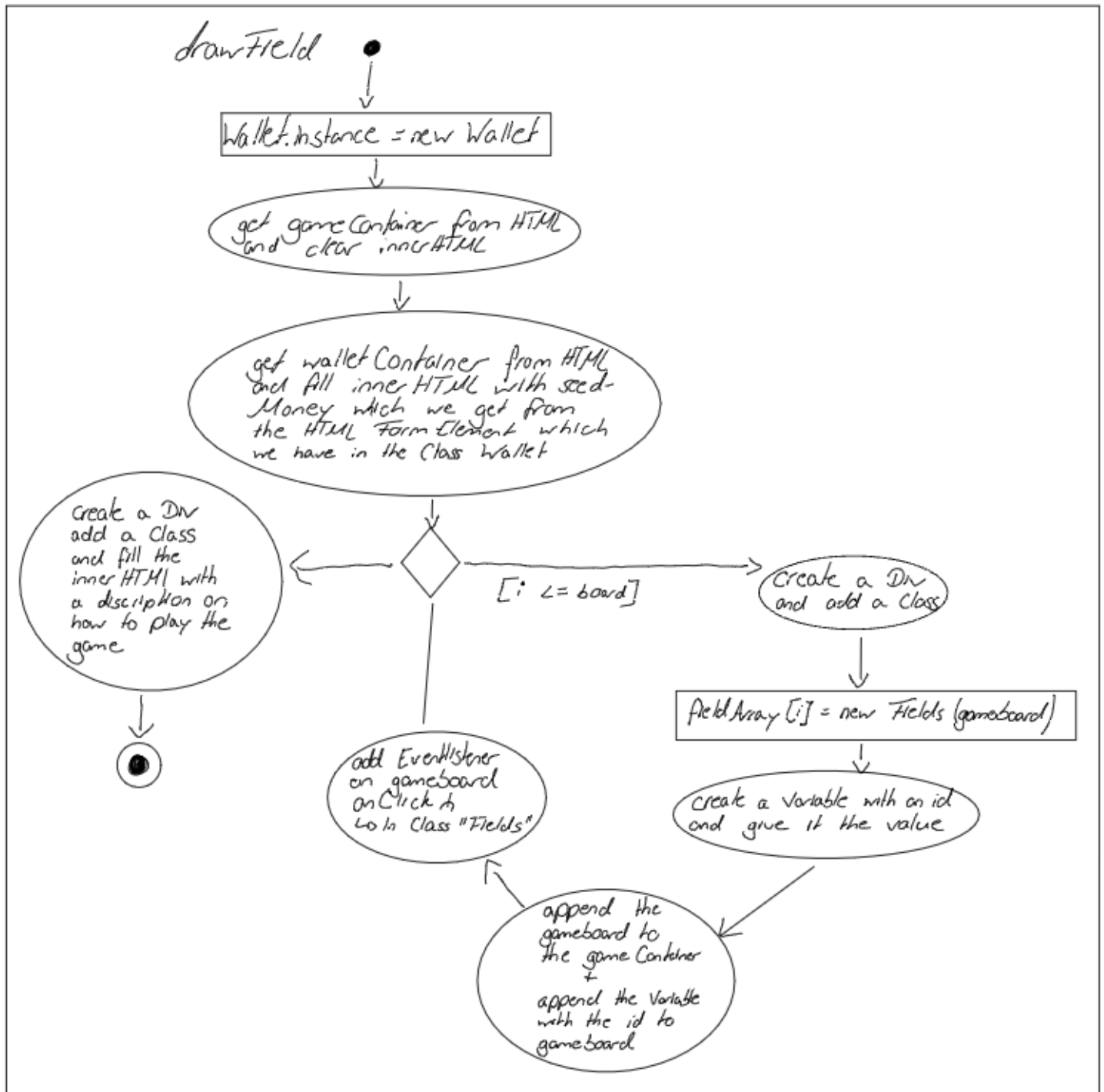
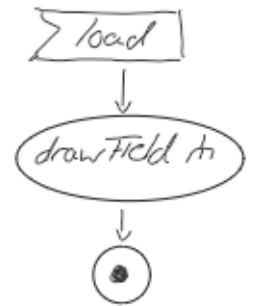
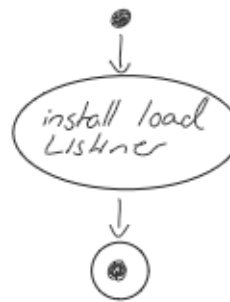
Lettuce
static informationInstance: Lettuce = new Lettuce
constructor (-uiField)

Carrots
static informationInstance: Carrots = new Carrots
constructor (-uiField)

Cucumber
static informationInstance: Cucumber = new Cucumber
constructor (-uiField)

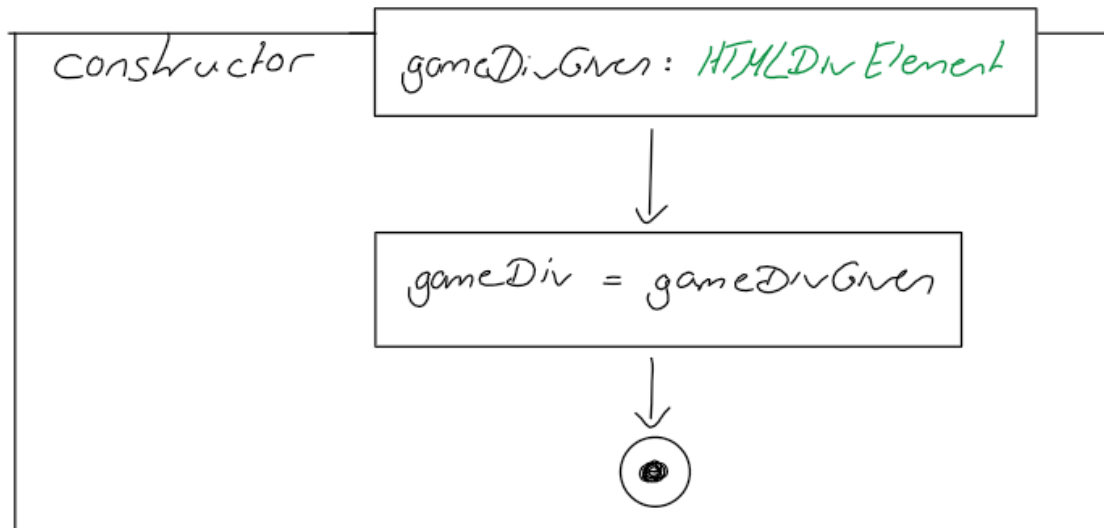
Activity Diagram - Main

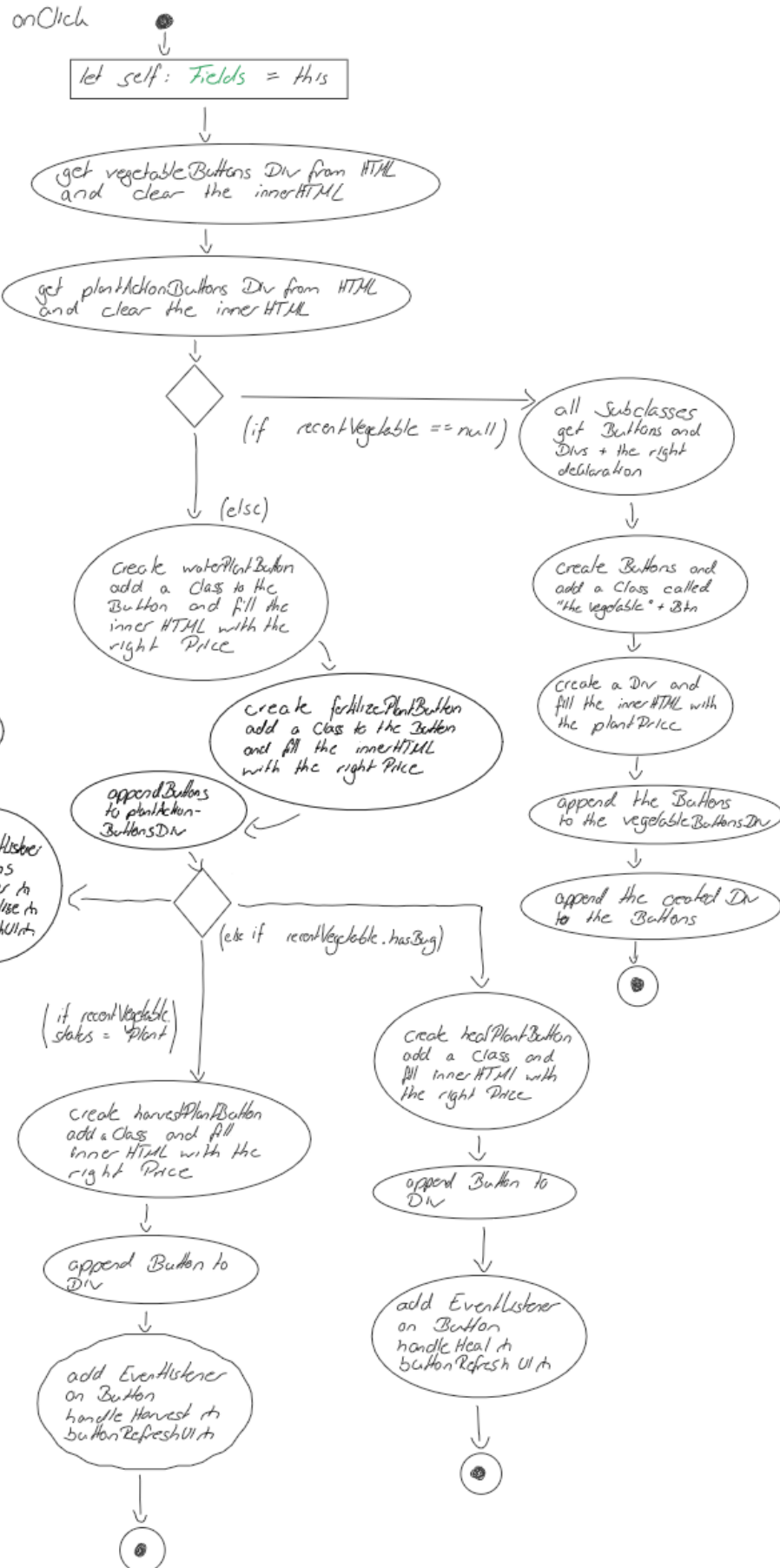
```
let board: number
let gameboard: HTMLDivElement
let gameContainer: HTMLDivElement
let fieldArray: Fields[] = []
let walletContainer: HTMLDivElement
let textContainer: HTMLDivElement
```

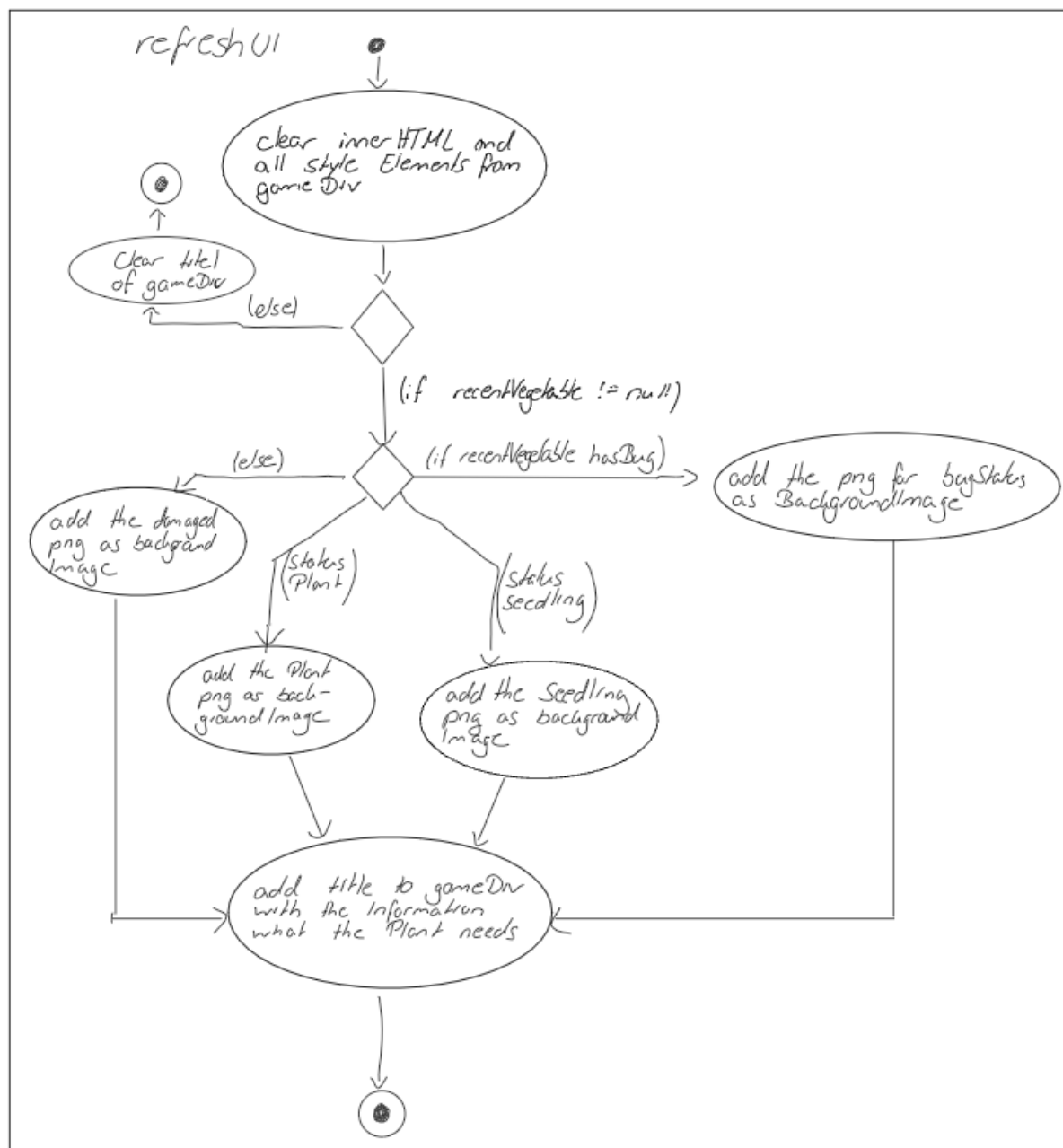
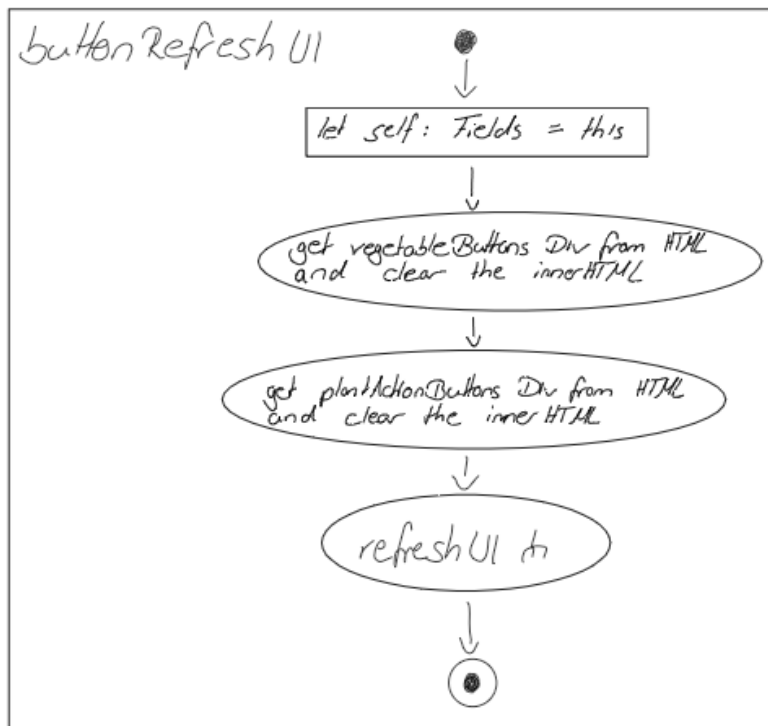


Class - Fields

gameDiv : HTMLDivElement
recentVegetable : Vegetable | null
infoContainer : HTMLDivElement







Activity Diagram - Super Class Vegetable

constructor

```
- uiField : Fields  
- growthStatus2 : string  
- growthTime : number  
- fertize : number  
- fertizePrice : number  
- water : number  
- price : number  
- income : number  
- health : number
```

set this. Variables to
_variables

```
this.growthStatus1 = "plant"  
this.damageStatus = "dead"  
this.bugStatus = "bugs"
```

Set beginning status of the
plant on seedling



(if uiField != null)

```
this.uiField = _uiField
```

handle Growth A

make a variable that counts
how often a bug can appear
on a plant
↳ Math.random
min : 1 , max : 5
name : possibleBugCount

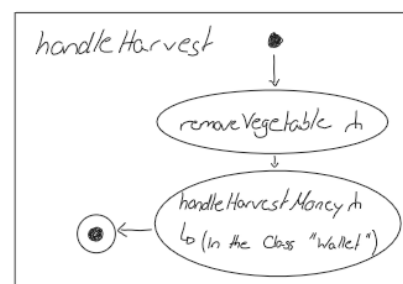
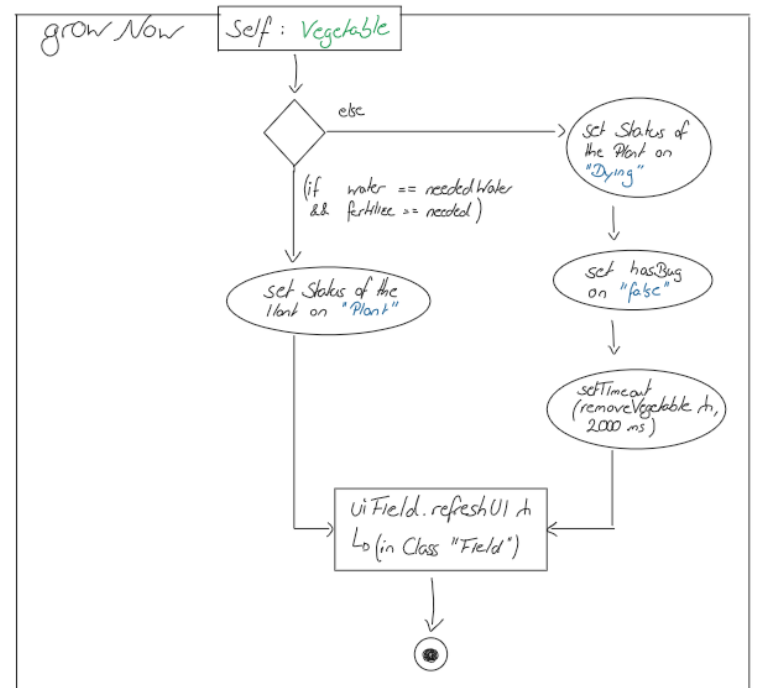
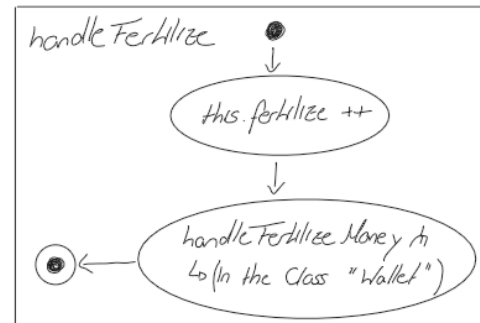
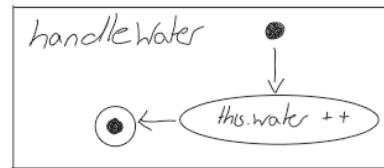
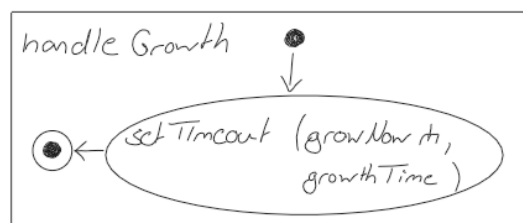
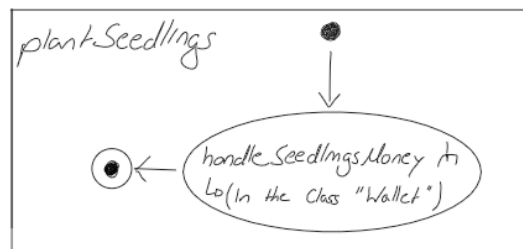
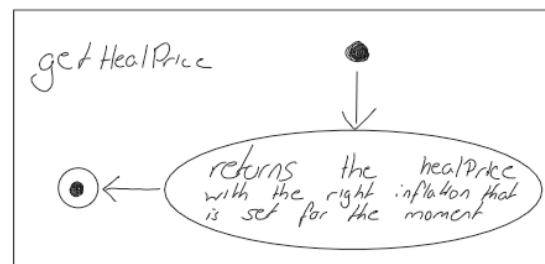
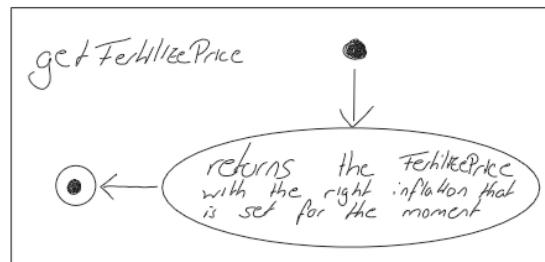
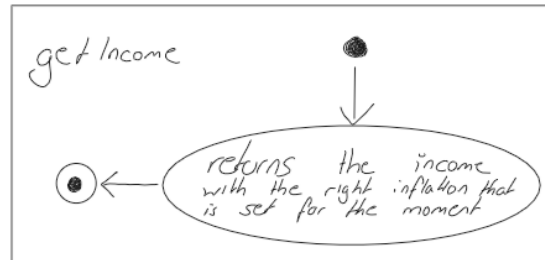
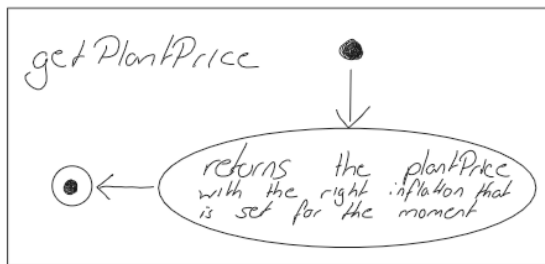


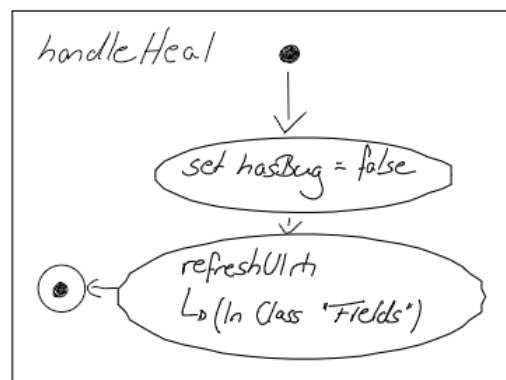
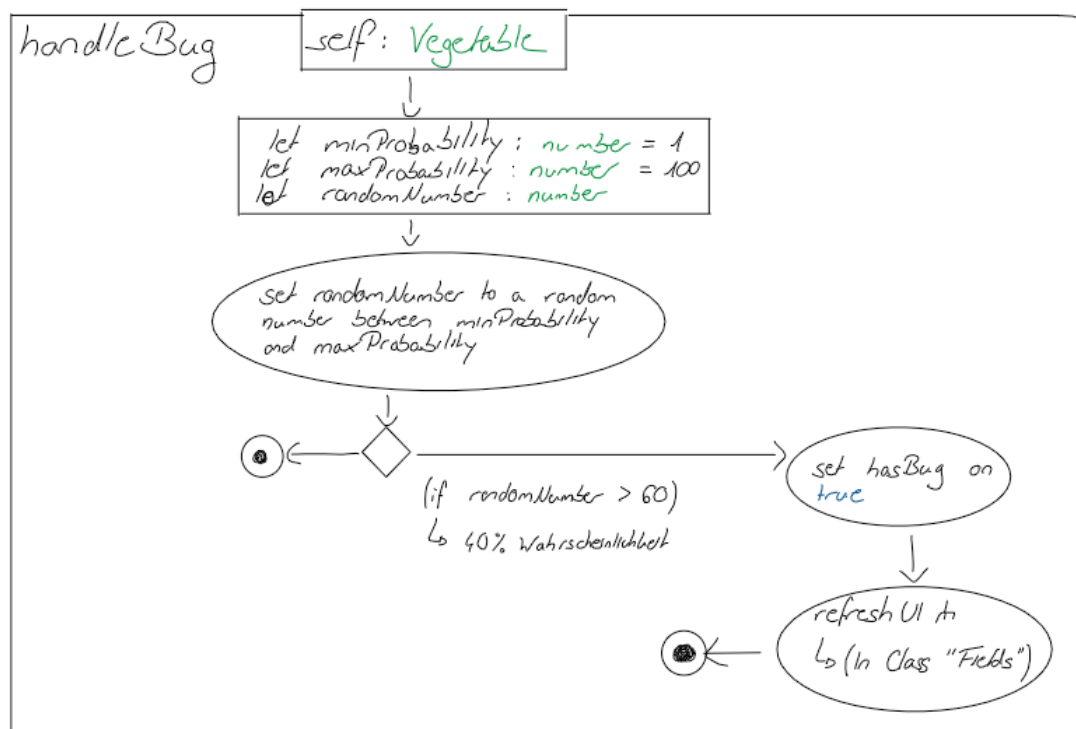
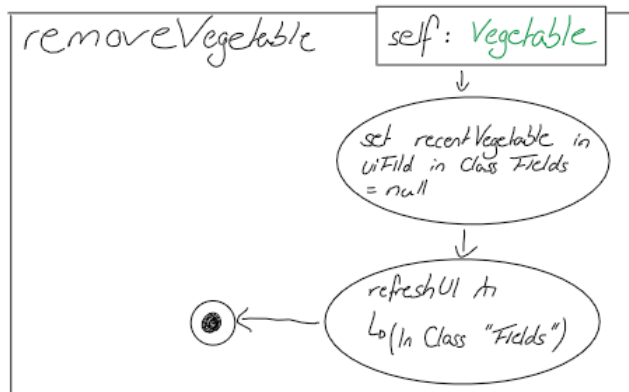
(for i < possibleBugCount)

make a variable that takes
the growthTime of a plant
as maximum and give a
random number in the
growthTime back
name = getRandomNumber

Set timeout (handleBug,
time = getRandomNumber

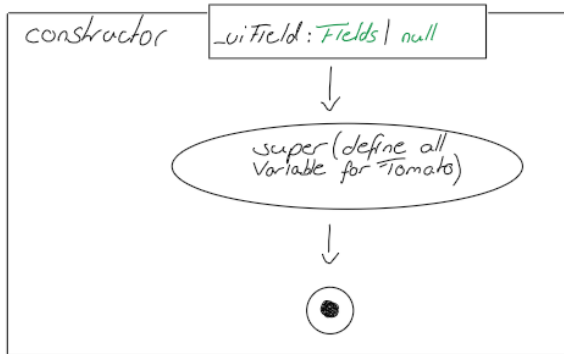






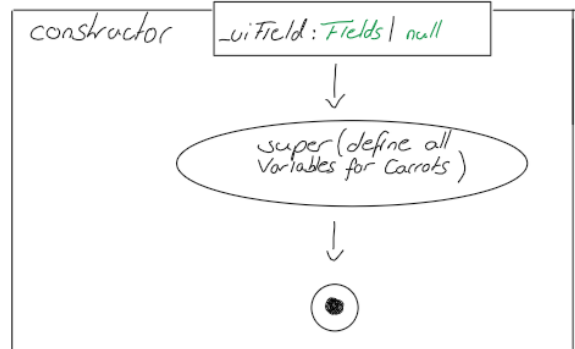
Subclass - Tomato

Static informationInstance: Tomato = new Tomato (null)



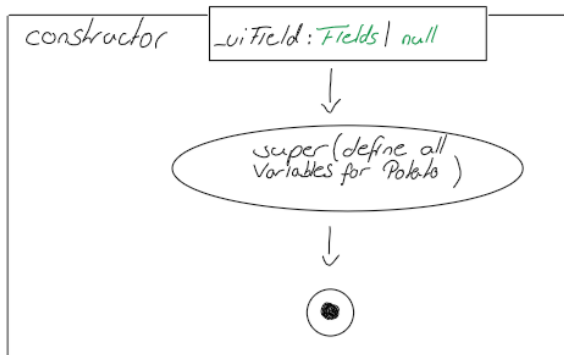
Subclass - Carrots

Static informationInstance: Carrots = new Carrots (null)



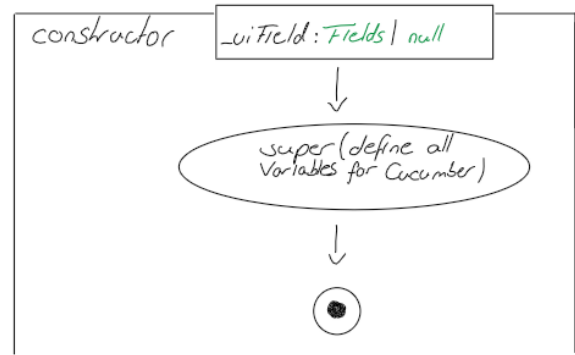
Subclass - Potato

Static informationInstance: Potato = new Potato (null)



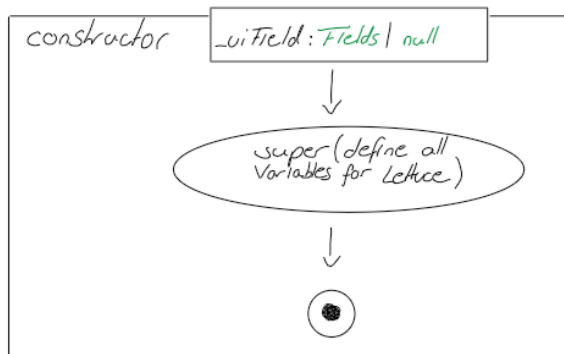
Subclass - Cucumber

Static informationInstance: Cucumber = new Cucumber (null)



Subclass - Lettuce

Static informationInstance: Lettuce = new Lettuce (null)



Class - Wallet

static instance: `Wallet`

alletContainer: `HTMLDivElement` = `document.querySelector`

urlParam: `URLSearchParams` = `new URLSearchParams(window.location.search)`

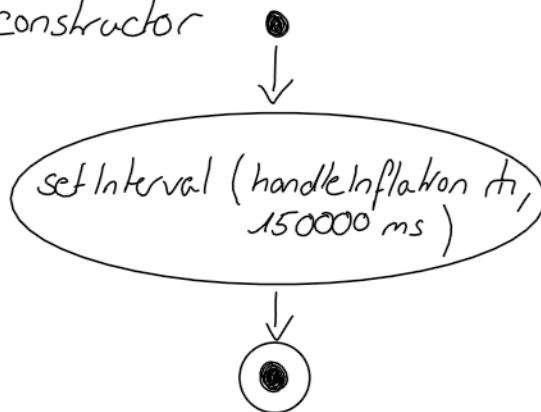
minPrice: `number` = get value from Form Element via urlParam

maxPrice: `number` = "

seedMoney: `number` = "

inflationRatio: `number` = 1

constructor



handle Inflation



```
let vegetableButtons: HTMLDivElement  
let plantActionButtons: HTMLDivElement
```



get vegetableButtons Div from HTML
and clear the innerHTML



get plantActionButtons Div from HTML
and clear the innerHTML



make a new Variable that gets
it's definition from another function
↳ random Int from Interval (min, max) at
name = changedInflation



```
inflationRatio = changedInflation / 100
```



alert, so the user sees
the difference



