Konzepte EIA2 Endabgabe

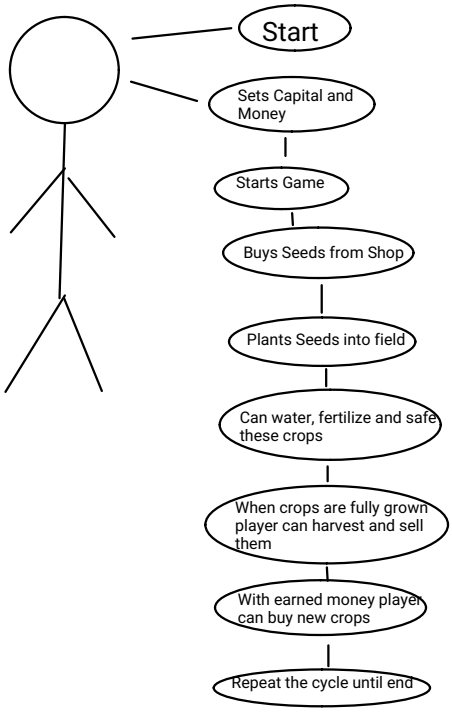
- 1.Konzept ist das auf dem karierten Blatt (diente nur zur Verständlichkeit und Übersicht)
- Endgültiges Konzept ist das folgende mit dem UI-Scribble, Klassendiagrammen und Aktivitätsdiagrammen

EIA 2

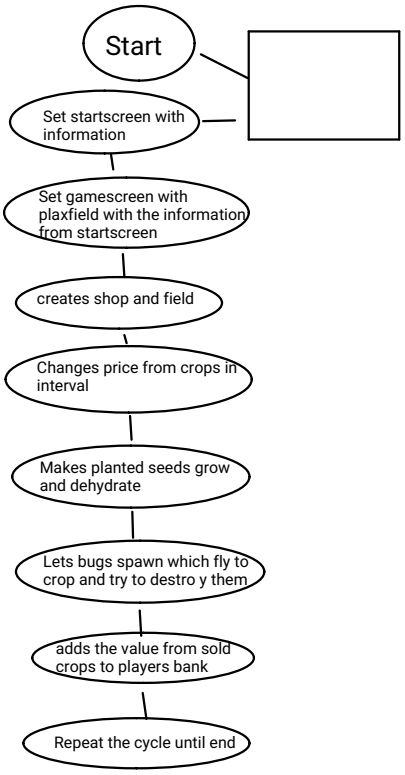
- [ ] Konzept (Klassendiagram, Aktivitätsd., ...)
- [ ] Anwendung
  - [ ] 40 Felder → jeweils eine Pflanze
  - [ ] Jedes Feld kann bepflanzt werden
  - [ ] Pflanze stellt Wachsstadium da (klein, mittel, groß, dürr, ...)
  - [ ] Schädling → kann Pflanze zerstören (auch mehrere)
  - [ ] Aktionen: wird durch Interaktion mit Feld ausgelöst
    - [ ] gießen
    - [ ] düngen
    - [ ] Schadling bekämpfen
    - [ ] anbauen
    - [ ] ernten
    - [ ] Pflanze löschen
    - [ ] kaufen
  - [ ] 5 Gemüse
    - ↳ sehen unters. aus, brauchen unters. lang, brauchen unters. viel Wasser, ...
  - [ ] Kapital wird angezeigt (wird bei Start angegeben)
  - [ ] Marktpreise für:
    - [ ] Gemüse
    - [ ] Pestizide
    - [ ] Dünger

    } ändern sich
  - [ ] Pflanze stirbt wenn:
    - → nicht gegossen → übergossen
    - → überdüngt → Schädling
  - [ ] Pflanze tot = Feld wird leer
  - [ ] geerntet = Feld leer + Kapital steigt (aktueller Preis)
  - [ ] Nutzer kann vom Kapital kaufen:
    - [ ] Setzlinge
    - [ ] Dünger
    - [ ] Pestizide
  - [ ] Pestizide werden nur auf Schädling angewendet
    - → nicht aufs Feld
  - [ ] Simulation läuft beschleunigt
  - [ ] Schädlinge in Realzeit

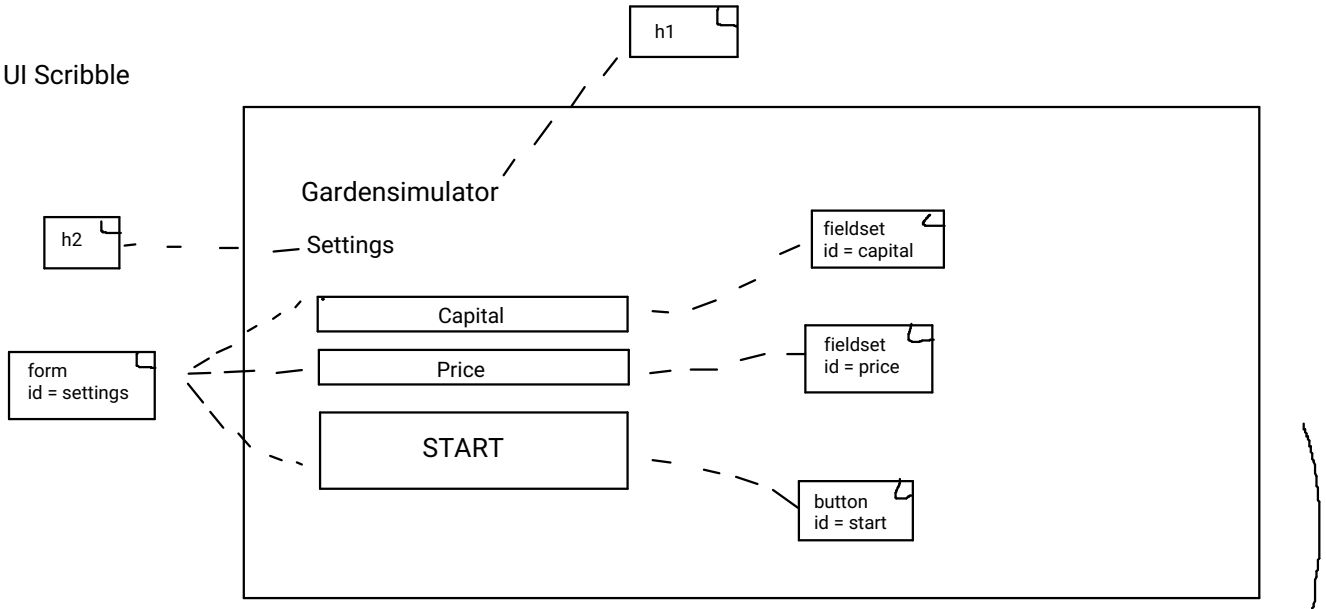
Player

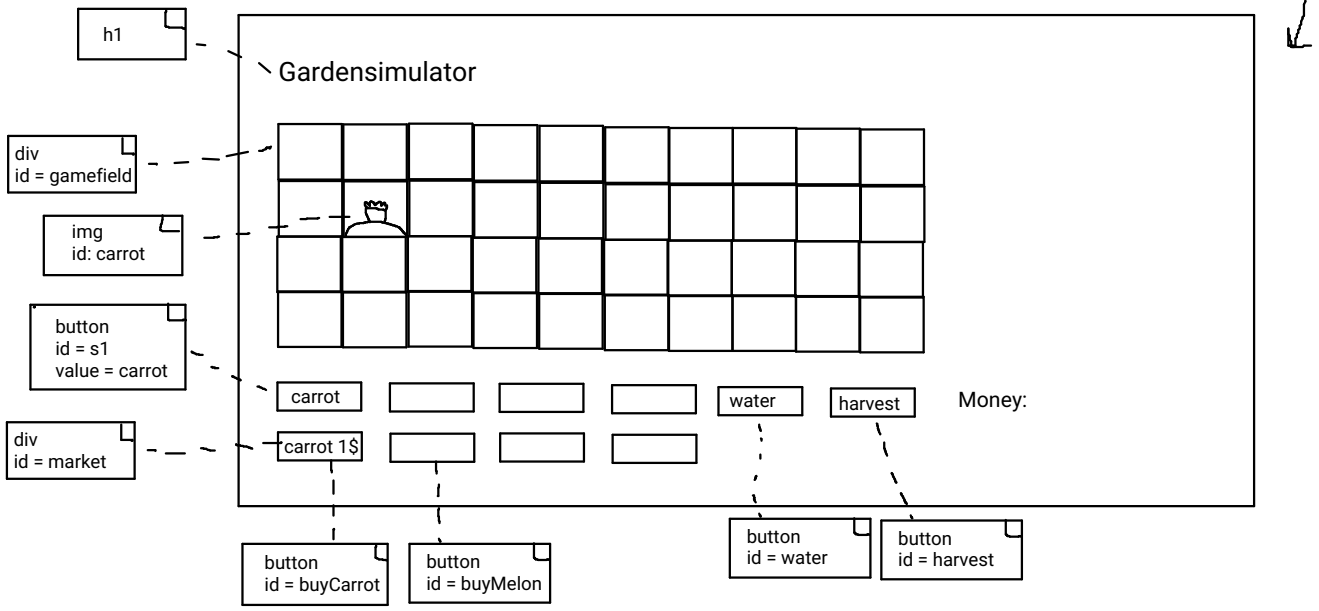


Computer

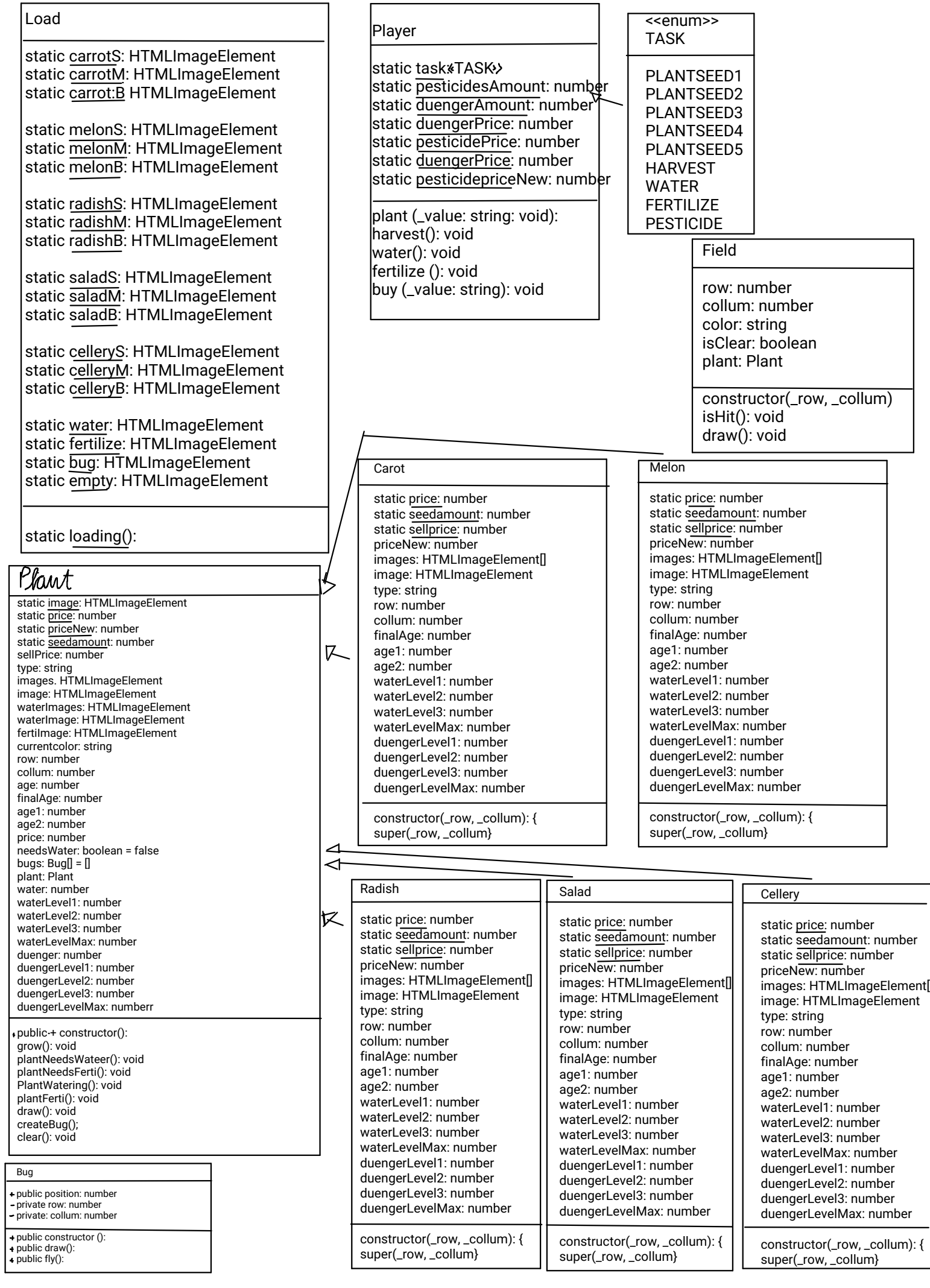


UI Scribble





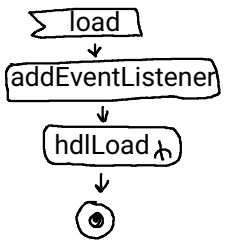
Class Diagram

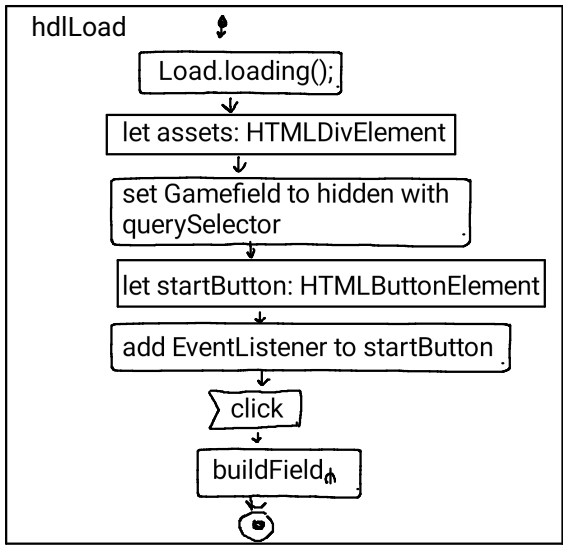


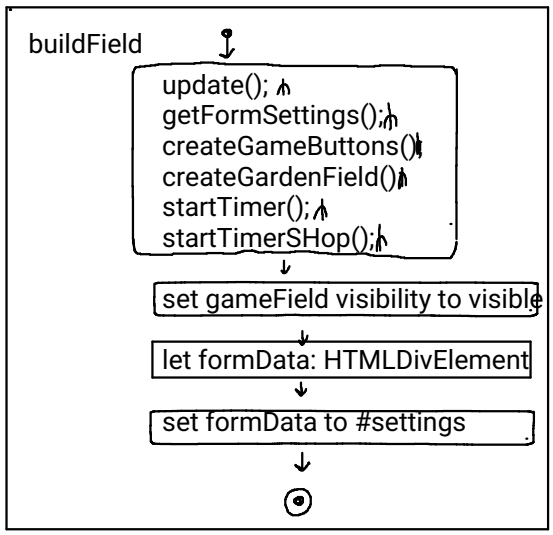
Acitivity-Diagram

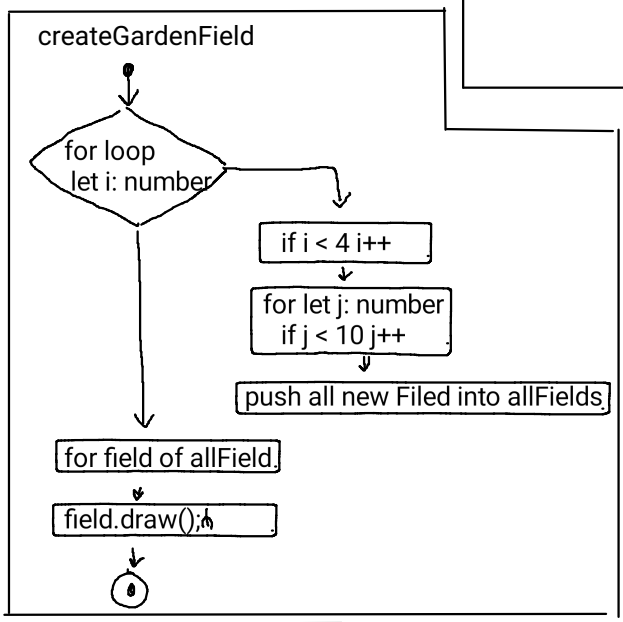
main

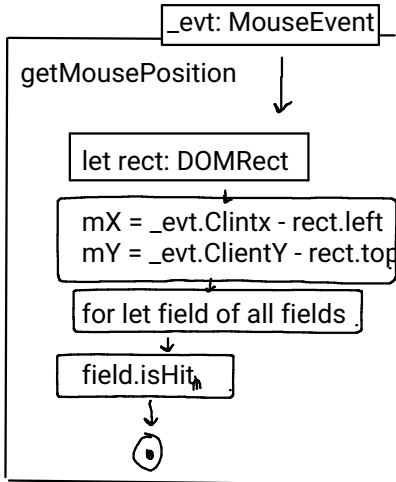
```
export let canvas: HTMLCanvasElement
export let cr2: CanvasRenderingContext2D
export let allFields: Field[] = []
export let allPlants: Plant[] = []
export let player: Player = new Player()
export let mX: number
export let mY: number
export let time: number
let gameField: HTMLDivElement
let formValues: FormData
export let money: number
let moneyChange: number
```

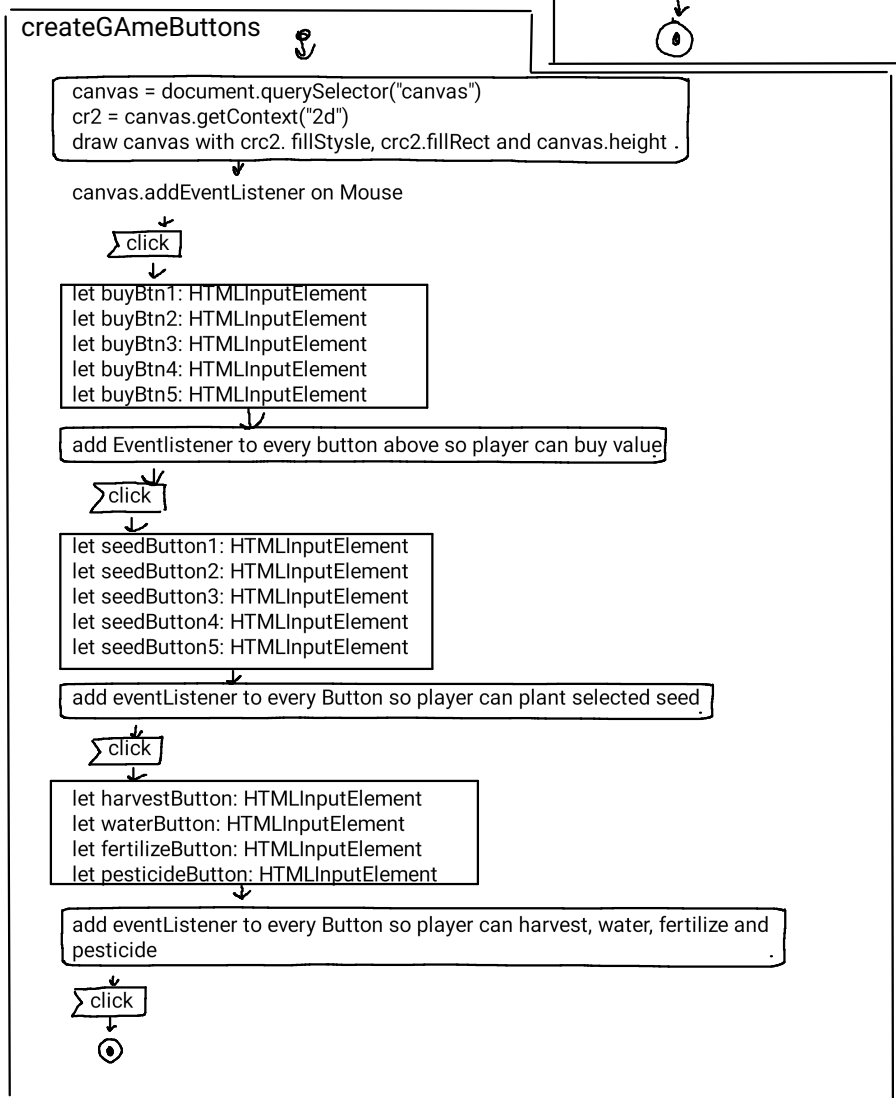


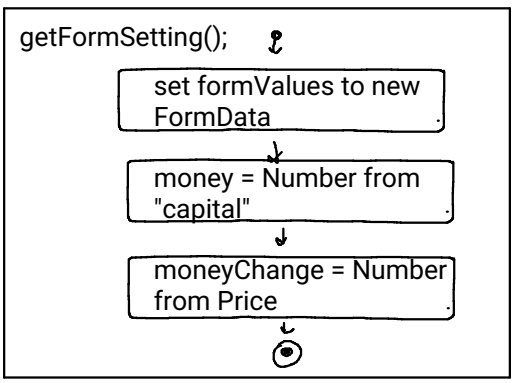


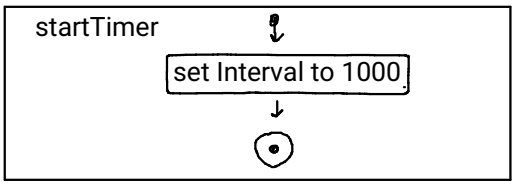


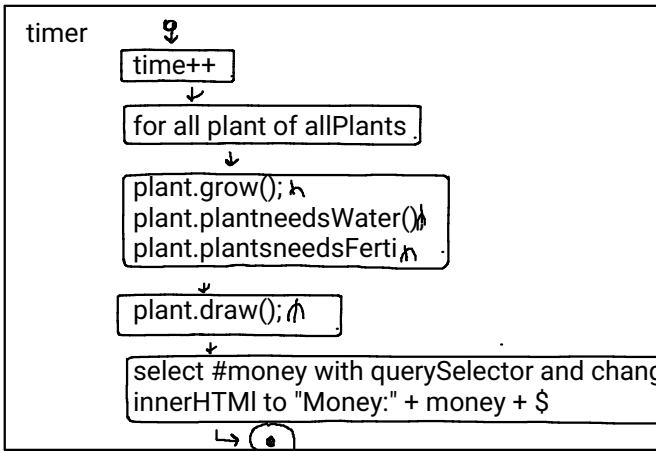


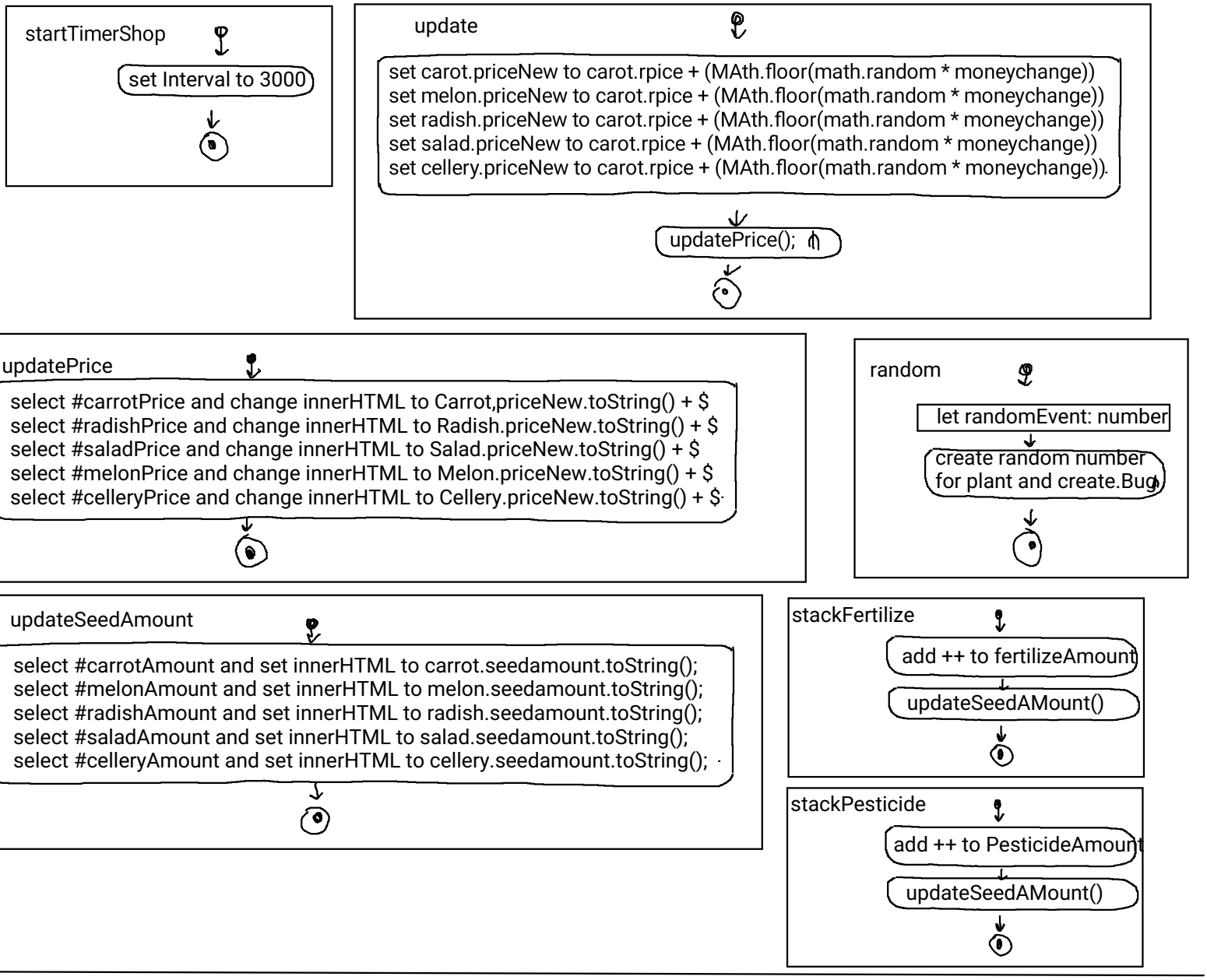












load

static carrotS: HTMLImageElement
static carrotM: HTMLImageElement
static carrot:B HTMLImageElement

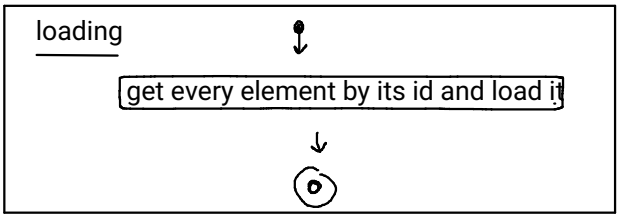
static melonS: HTMLImageElement
static melonM: HTMLImageElement
static melonB: HTMLImageElement

static radishS: HTMLImageElement
static radishM: HTMLImageElement
static radishB: HTMLImageElement

static saladS: HTMLImageElement
static saladM: HTMLImageElement
static saladB: HTMLImageElement

static celleryS: HTMLImageElement
static celleryM: HTMLImageElement
static celleryB: HTMLImageElement

static water: HTMLImageElement
static fertilize: HTMLImageElement
static bug: HTMLImageElement
static empty: HTMLImageElement

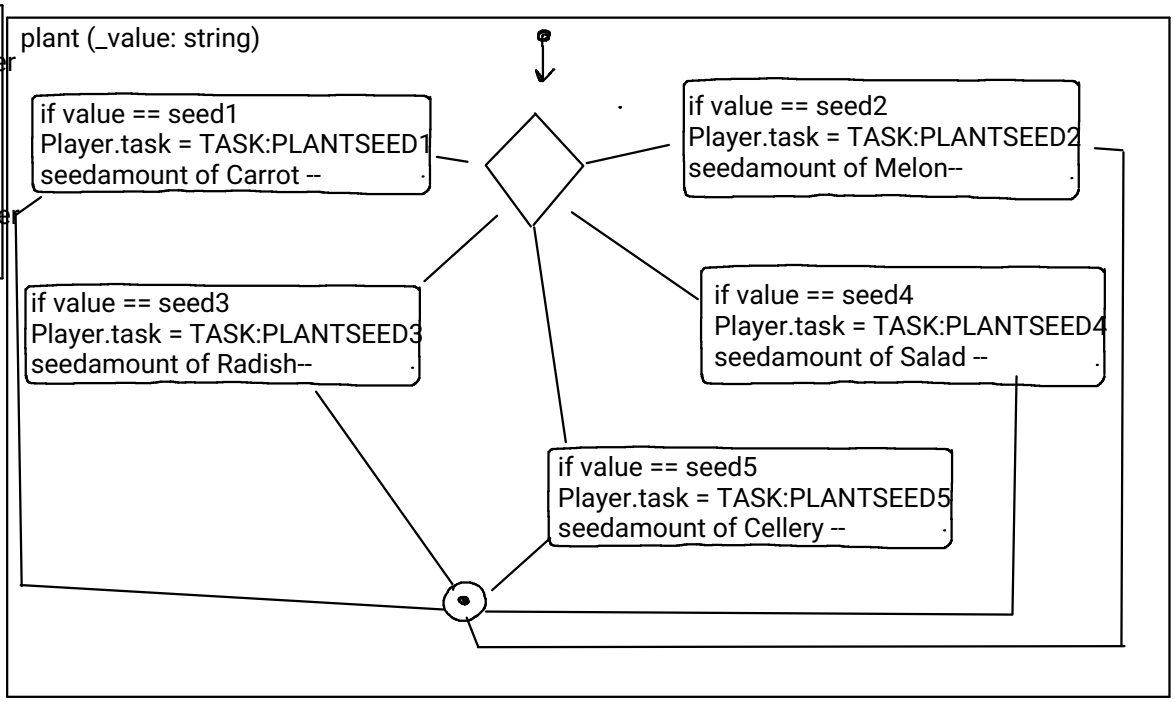


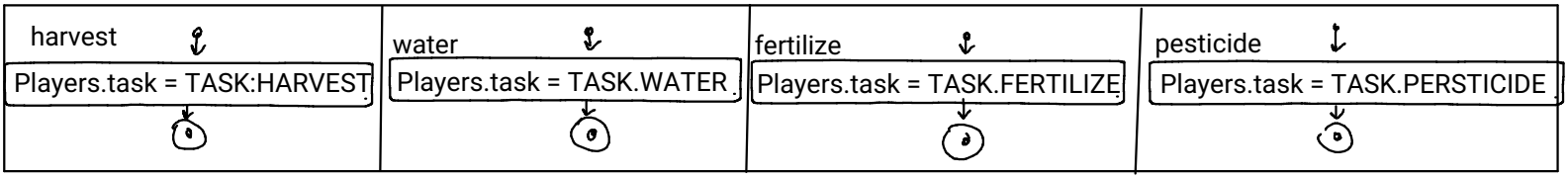
Player

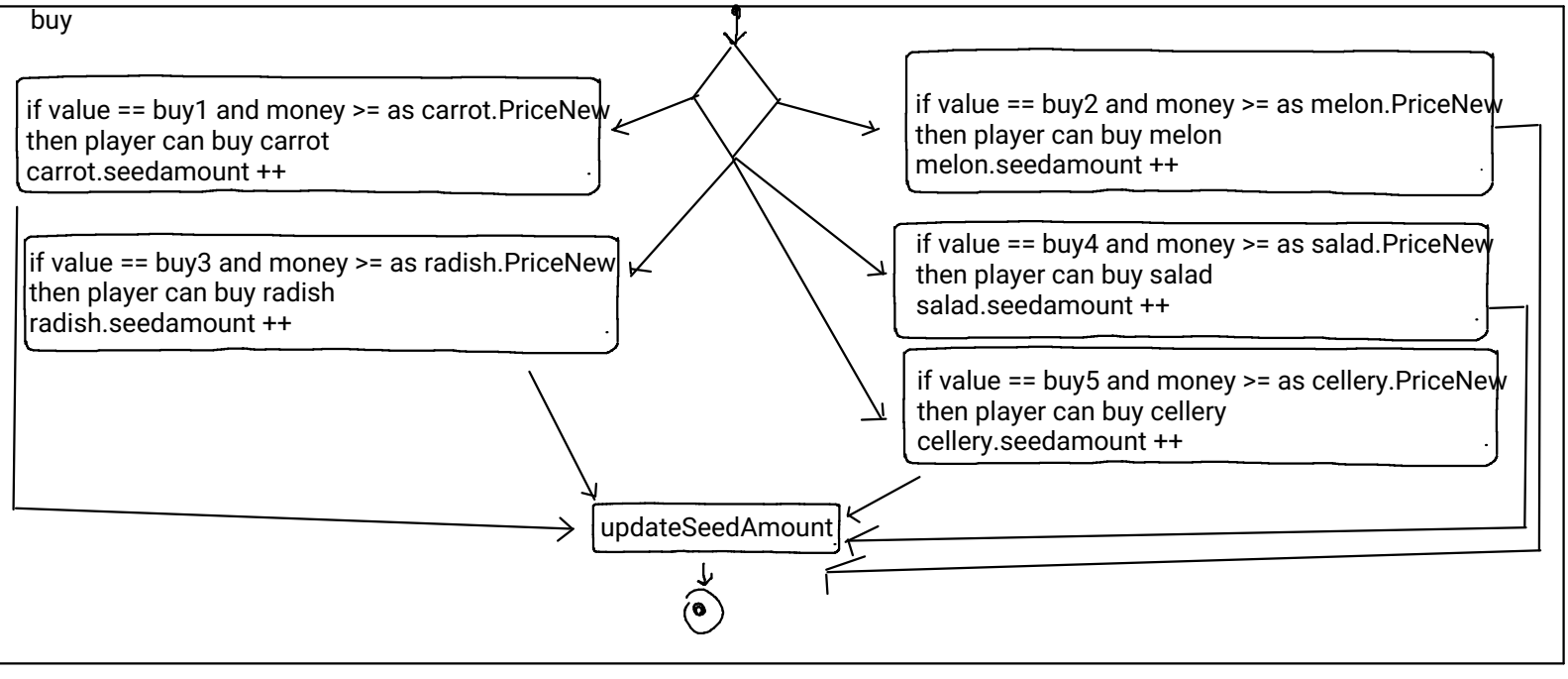
static task: TASK
static pesticidesAmount: number
static duengerAmount: number
static duengerPrice: number
static pesticidePrice: number
static duengerPrice: number
static pesticidepriceNew: number

<<enum>>
TASK

PLANTSEED1
PLANTSEED2
PLANTSEED3
PLANTSEED4
PLANTSEED5
HARVEST
WATER
FERTILIZE
PESTICIDE







Field

row: number
collum: number
color: string
isClear: boolean
plant: Plant

constructor

this.row = _row
this.collum = _collum

isHit

_mX, _mY

if _mX and _mY have special coordinates

switch to Player.task

case TASK.PlantSeed1

if carot.seedAmount >= 0 push all new carrots into this.row and this.collum
this plant = allPlants.length - 1

updateSeedAmount();

case TASK.PlantSeed4

if salad.seedAmount >= 0 push all new salad into this.row and this.collum
this plant = allPlants.length - 1

updateSeedAmount();

case TASK.PlantSeed2

if melon.seedAmount >= 0 push all new melons into this.row and this.collum
this plant = allPlants.length - 1

updateSeedAmount();

case TASK.PlantSeed1

if cellery.seedAmount >= 0 push all new cellery into this.row and this.collum
this plant = allPlants.length - 1

updateSeedAmount();

case TASK.PlantSeed3

if radish.seedAmount >= 0 push all new radish into this.row and this.collum
this plant = allPlants.length - 1

updateSeedAmount();

caste TASK.HARVESt

if this.plant reaches over age 2 and under final age

plant can be sold allPlants.slice (allPlants.findIndex e => e == this.plant)

timer ();
startTimerSHop();

case TASK.PESITIZIDE

if this.isClear == false

this.plant.plantkillBug();

case TASK.FERTILIZE

if this.isClear == false

this.plant.plantFertilize();

case TASK.WATER

if this.isClear == false

this.plant.plantWatering();

this.draw

for all Plants

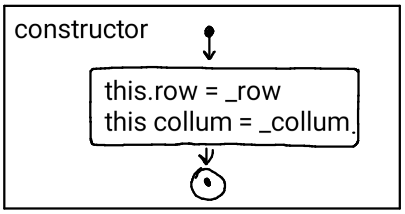
plant.draw();

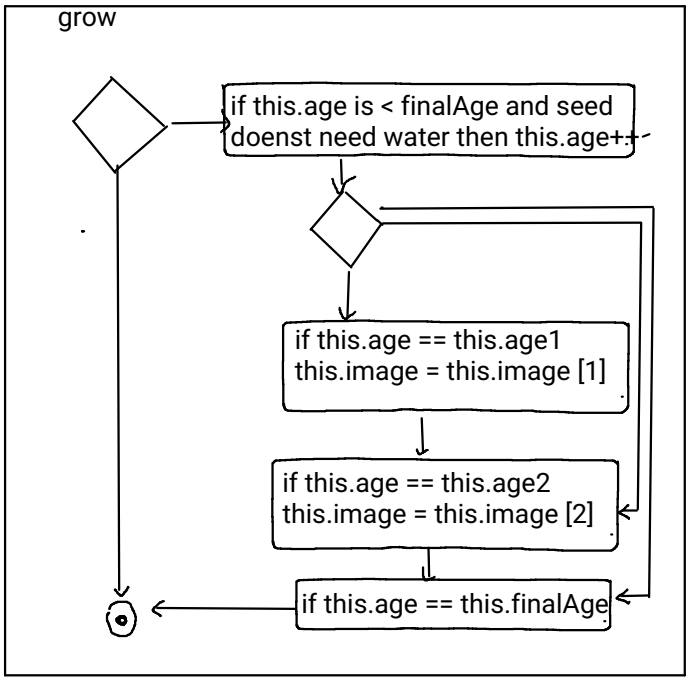
draw

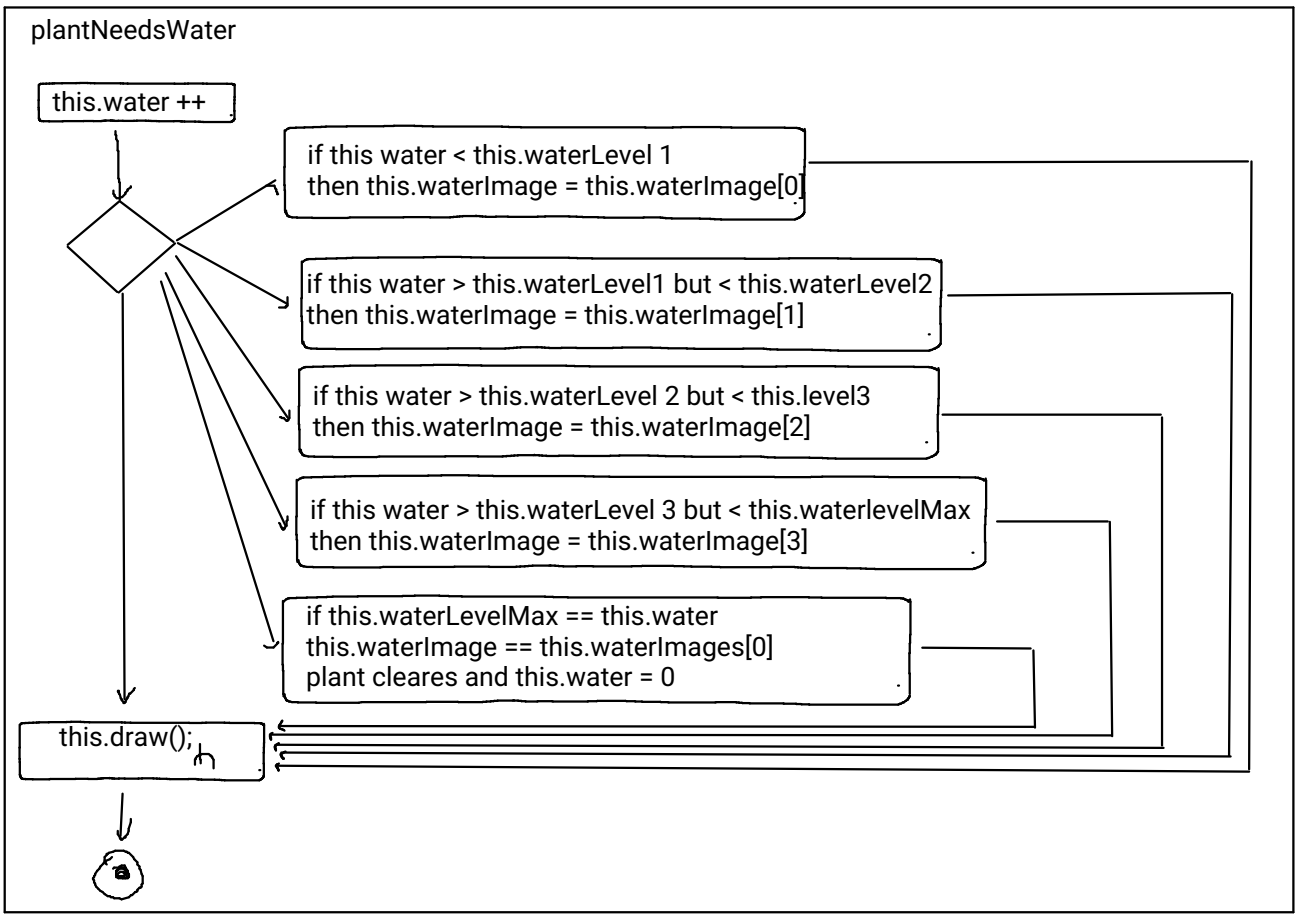
draw with crc2. resetTransform, crc2.fillStyle and crc2.fillRect

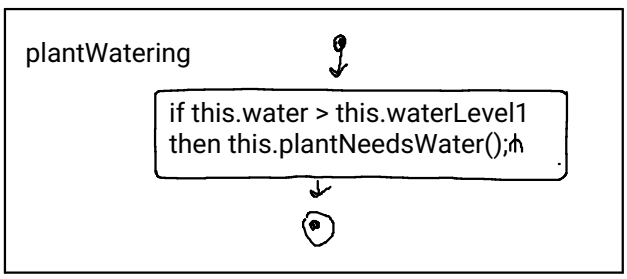
Plant

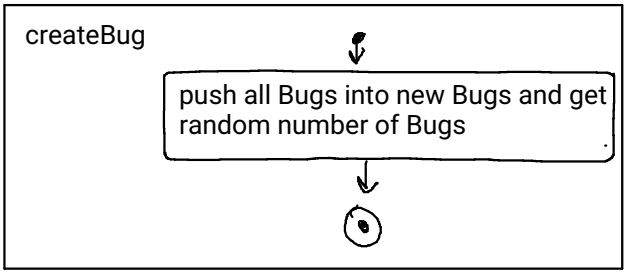
static image: HTMLImageElement
static price: number
static priceNew: number
static seedamount: number
sellPrice: number
type: string
images. HTMLImageElement
image: HTMLImageElement
waterImages: HTMLImageElement
waterImage: HTMLImageElement
fertiImage: HTMLImageElement
currentcolor: string
row: number
collum: number
age: number
finalAge: number
age1: number
age2: number
price: number
needsWater: boolean = false
bugs: Bug[] = []
plant: Plant
water: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duenger: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: numberr

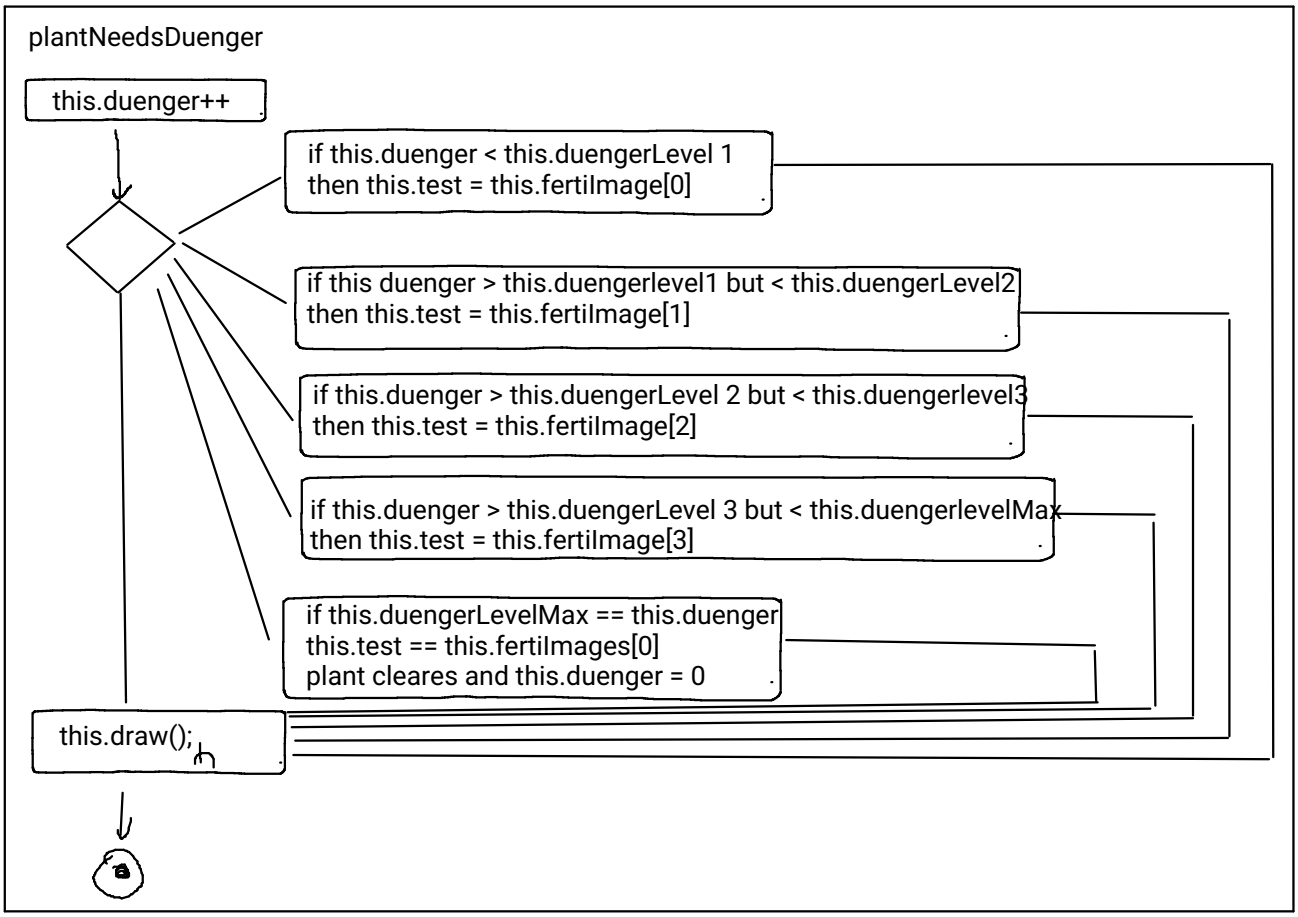


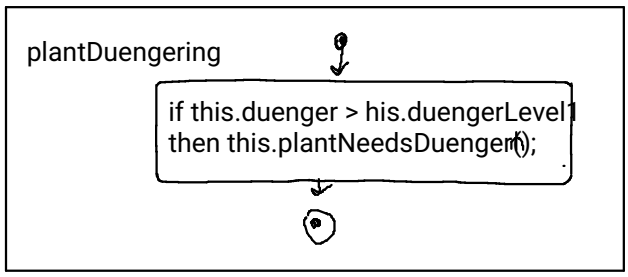


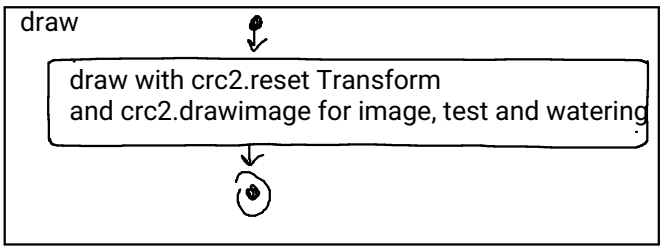


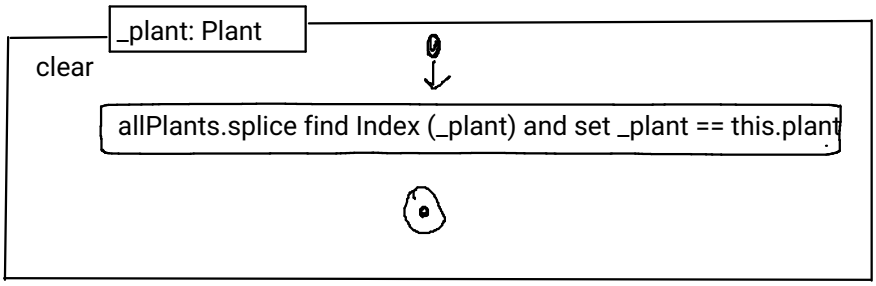


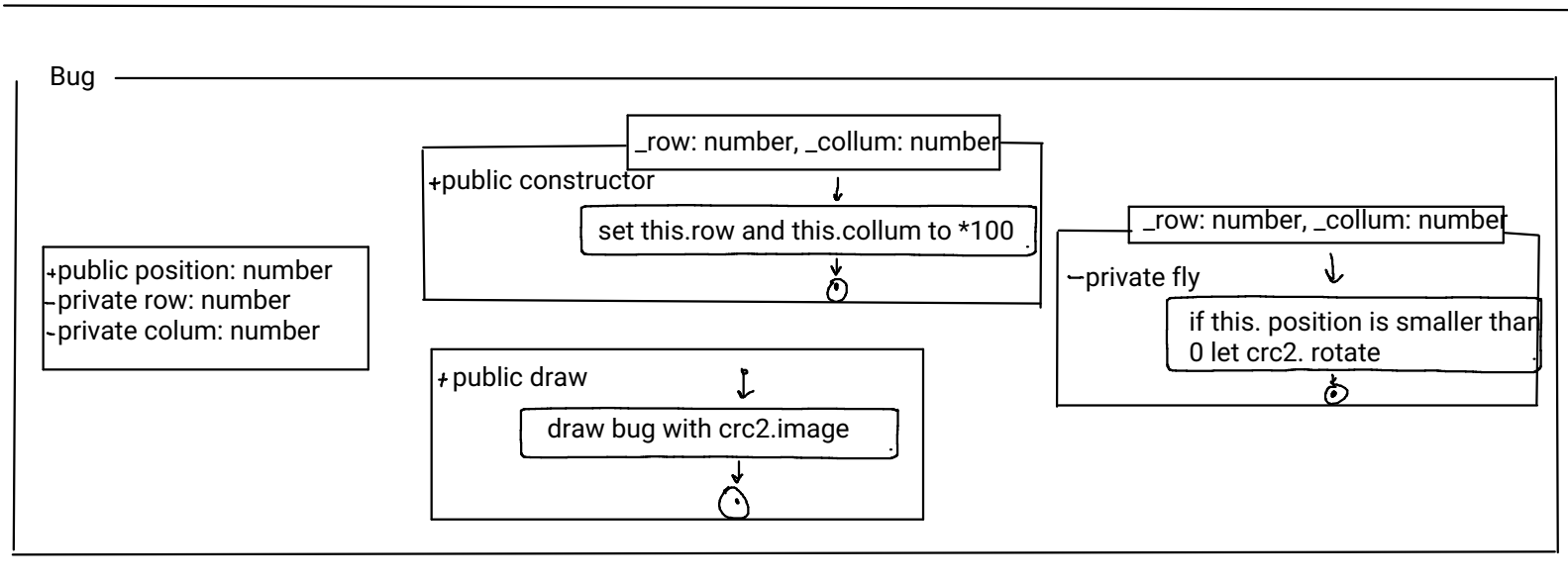


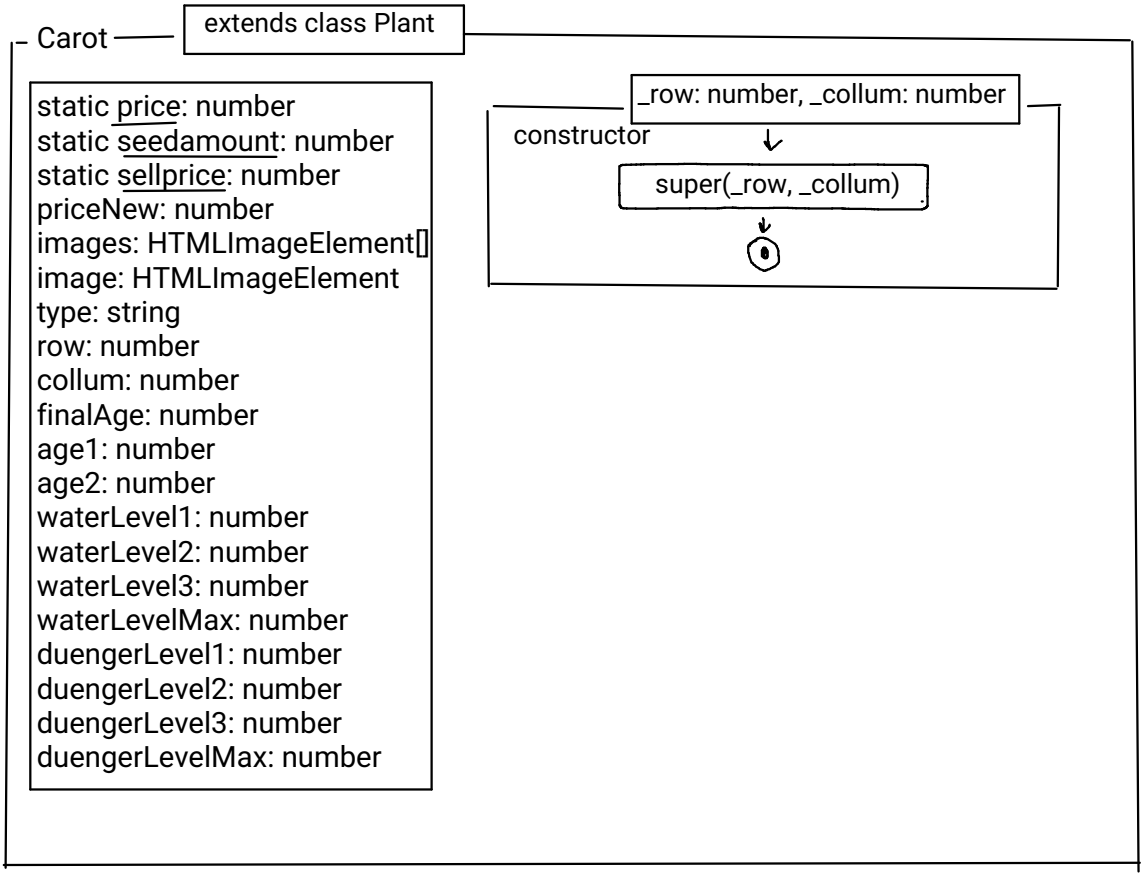

plantNeedsDuenger
this.duenger++
if this.duenger < this.duengerLevel 1
then this.test = this.fertiImage[0]
if this duenger > this.duengerlevel1 but < this.duengerLevel2
then this.test = this.fertiImage[1]
if this.duenger > this.duengerLevel 2 but < this.duengerlevel3
then this.test = this.fertiImage[2]
if this.duenger > this.duengerLevel 3 but < this.duengerlevelMax
then this.test = this.fertiImage[3]
if this.duengerLevelMax == this.duenger
this.test == this.fertiImages[0]
plant cleares and this.duenger = 0
this.draw();

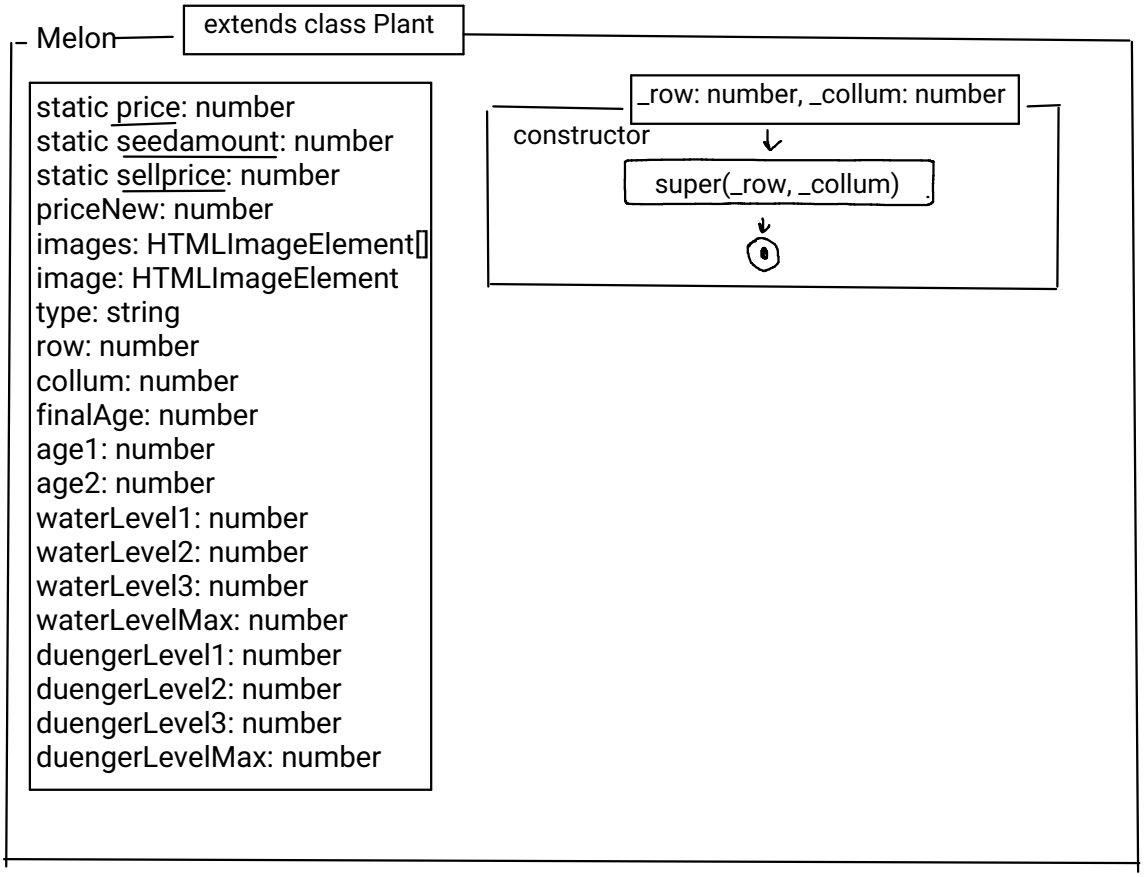

plantDuengering
if this.duenger > his.duengerLevel1
then this.plantNeedsDuenger();

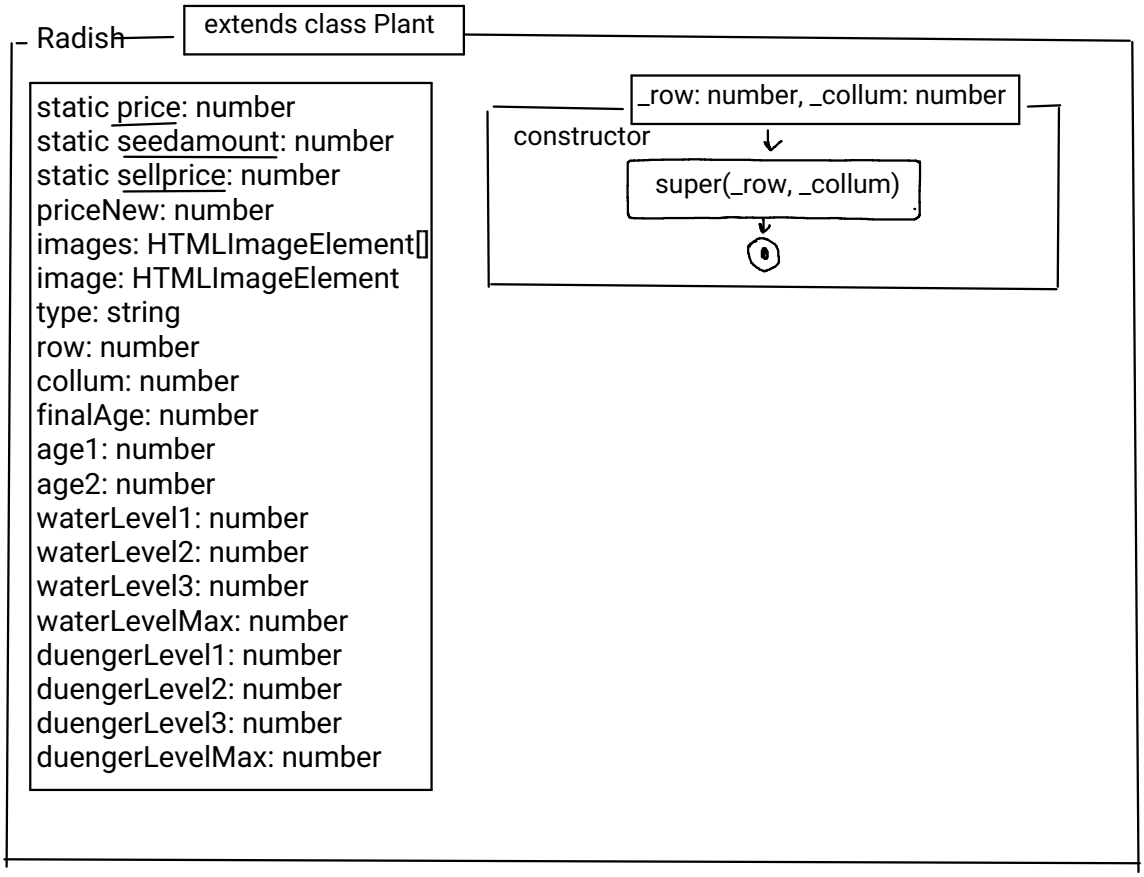

draw
draw with crc2.reset Transform
and crc2.drawimage for image, test and watering

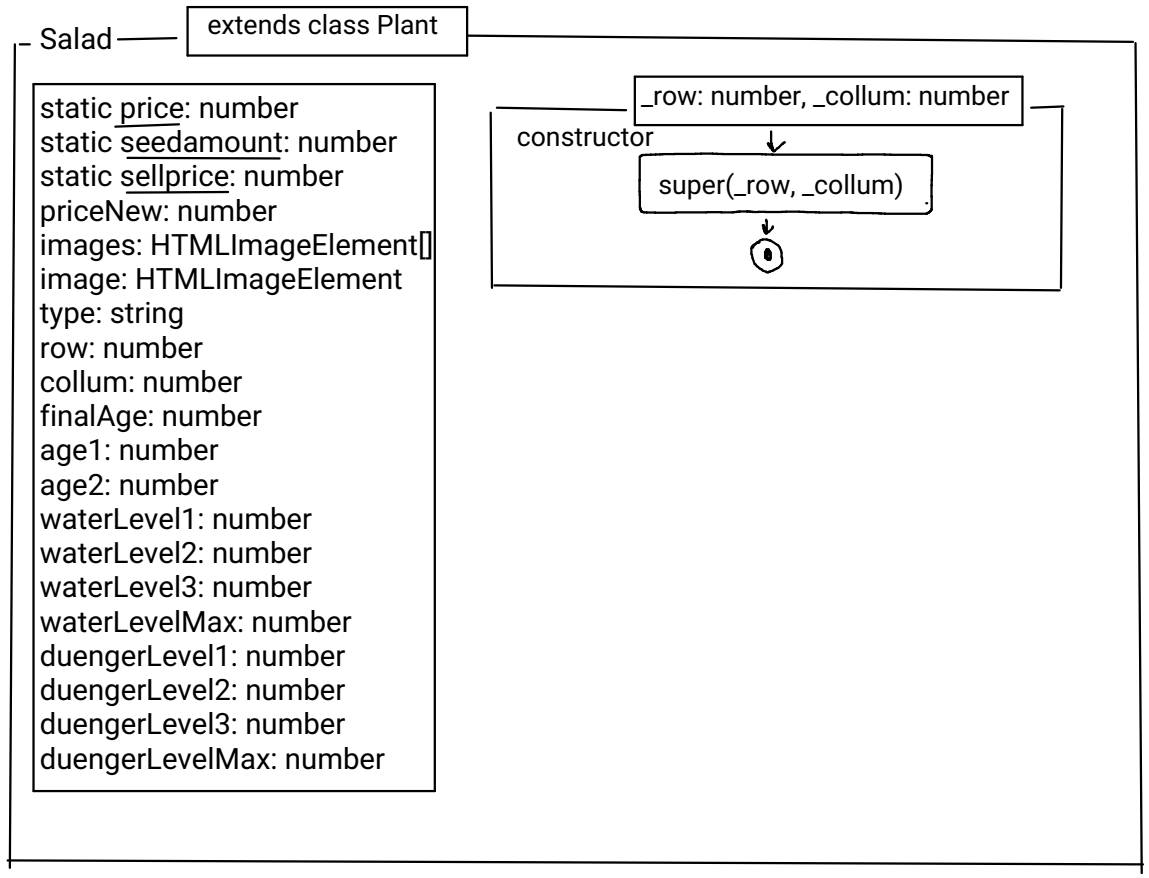

_plant: Plant
clear
allPlants.splice find Index (_plant) and set _plant == this.plant

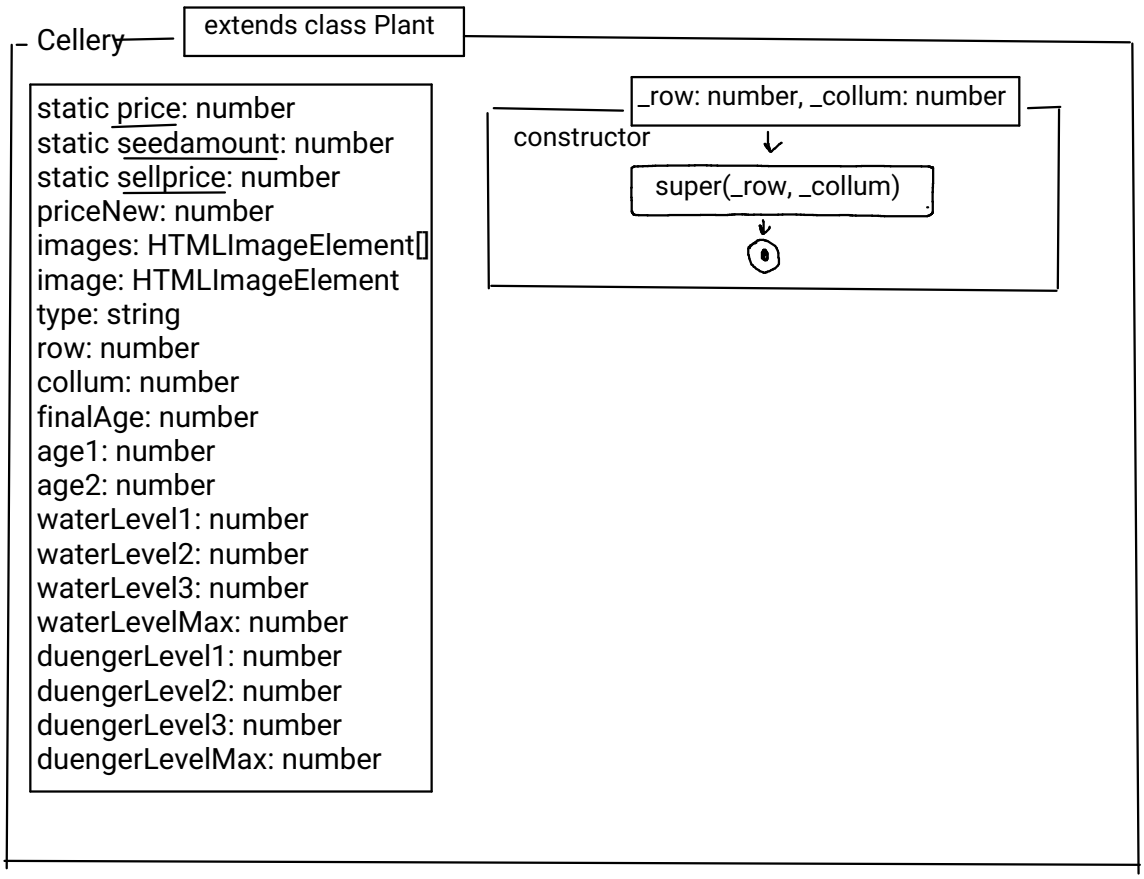

Bug
+public position: number
-private row: number
-private colum: number
_row: number, _collum: number
+public constructor
set this.row and this.collum to *100
_row: number, _collum: number
-private fly
if this. position is smaller than
0 let crc2. rotate
+ public draw
draw bug with crc2.image


Carot
extends class Plant
static price: number
static seedamount: number
static sellprice: number
priceNew: number
images: HTMLImageElement[]
image: HTMLImageElement
type: string
row: number
collum: number
finalAge: number
age1: number
age2: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: number
_row: number, _collum: number
constructor
super(_row, _collum)


Melon
extends class Plant
static price: number
static seedamount: number
static sellprice: number
priceNew: number
images: HTMLImageElement[]
image: HTMLImageElement
type: string
row: number
collum: number
finalAge: number
age1: number
age2: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: number
_row: number, _collum: number
constructor
super(_row, _collum)


Radish
extends class Plant
static price: number
static seedamount: number
static sellprice: number
priceNew: number
images: HTMLImageElement[]
image: HTMLImageElement
type: string
row: number
collum: number
finalAge: number
age1: number
age2: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: number
_row: number, _collum: number
constructor
super(_row, _collum)


Salad
extends class Plant
static price: number
static seedamount: number
static sellprice: number
priceNew: number
images: HTMLImageElement[]
image: HTMLImageElement
type: string
row: number
collum: number
finalAge: number
age1: number
age2: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: number
_row: number, _collum: number
constructor
super(_row, _collum)


Cellery
extends class Plant
static price: number
static seedamount: number
static sellprice: number
priceNew: number
images: HTMLImageElement[]
image: HTMLImageElement
type: string
row: number
collum: number
finalAge: number
age1: number
age2: number
waterLevel1: number
waterLevel2: number
waterLevel3: number
waterLevelMax: number
duengerLevel1: number
duengerLevel2: number
duengerLevel3: number
duengerLevelMax: number
_row: number, _collum: number
constructor
super(_row, _collum)