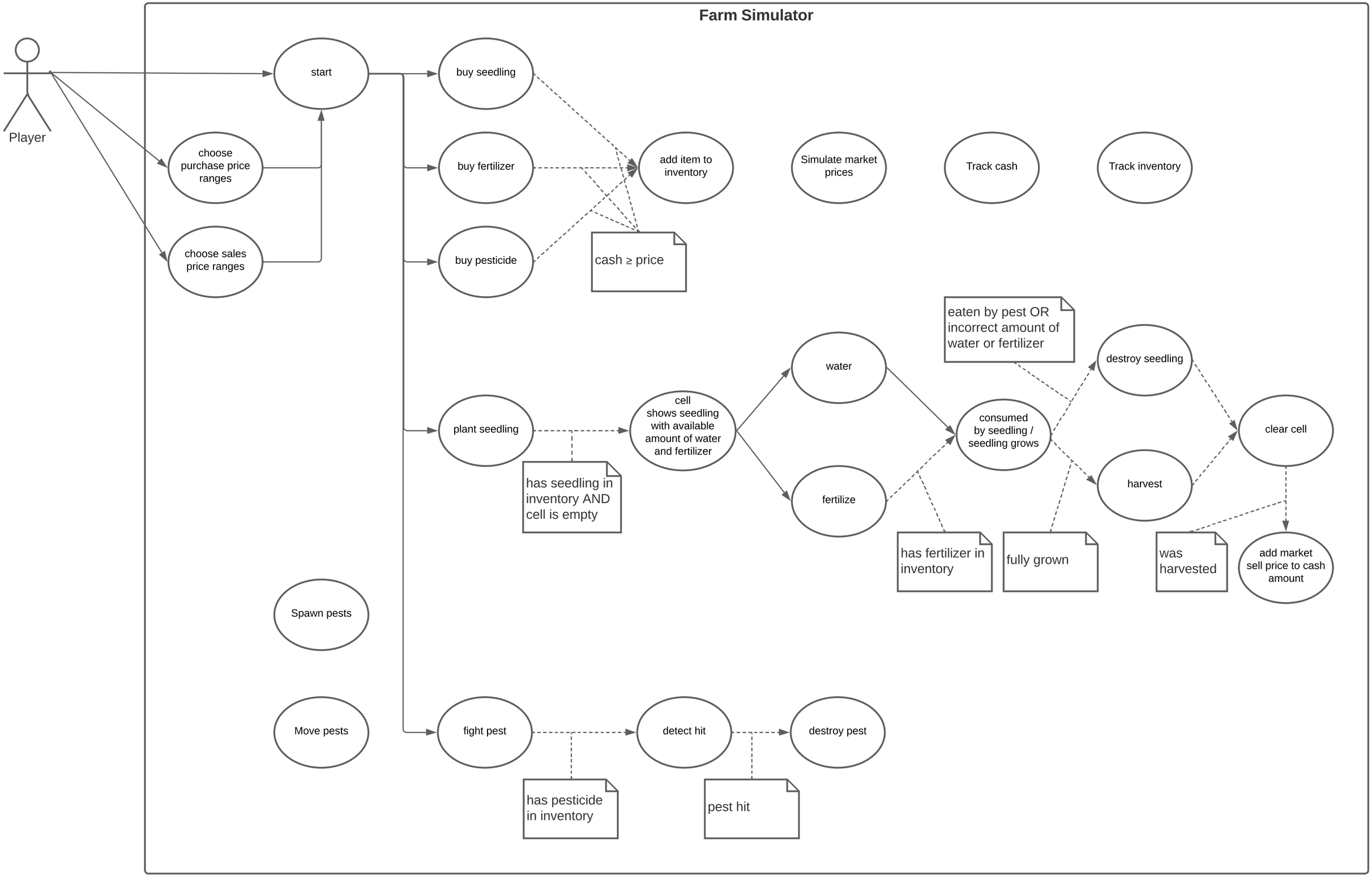
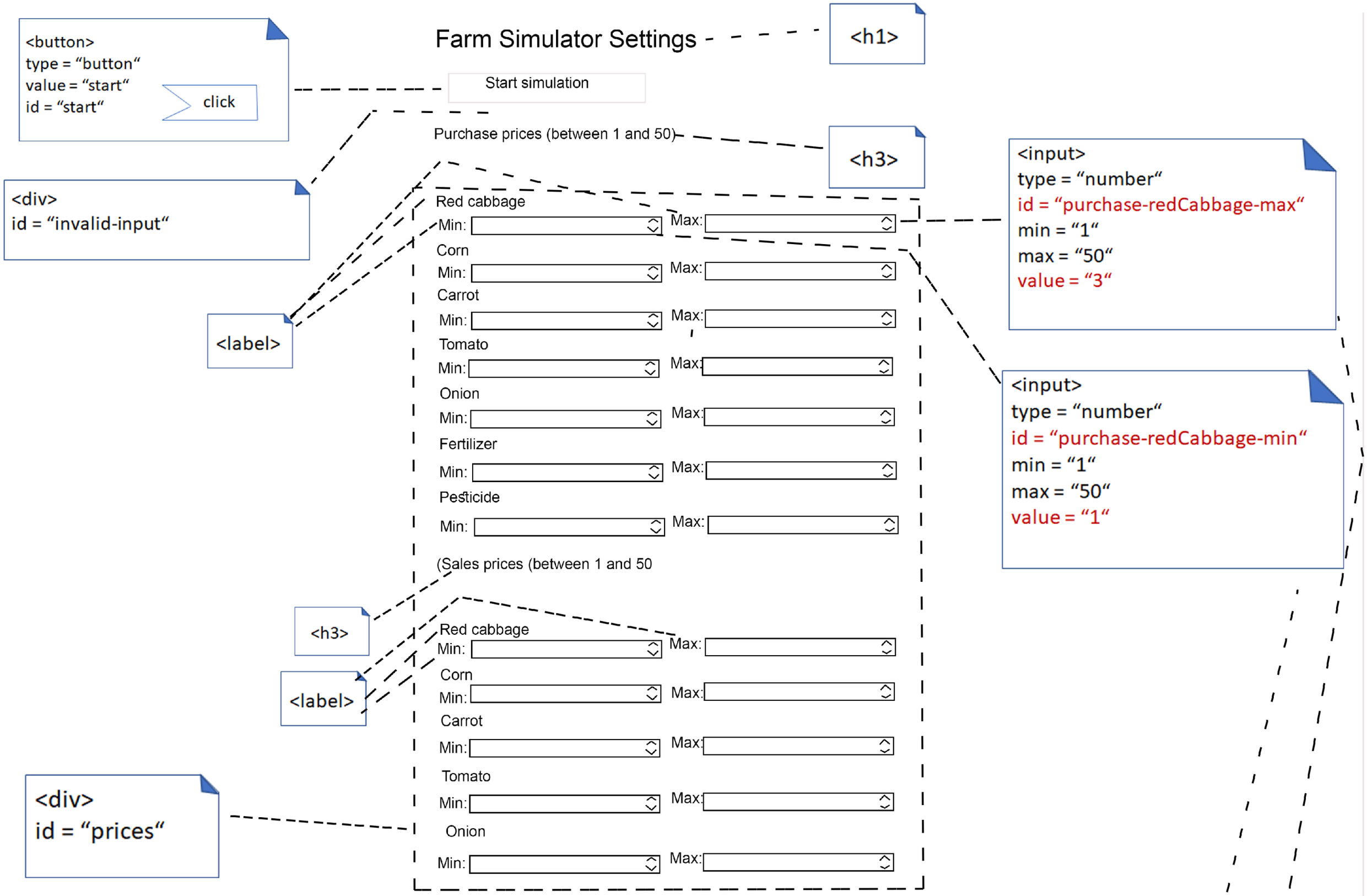
Design Document
EIA2 – Farm Simulator

Manuel Kasperzack

Use Case Diagram

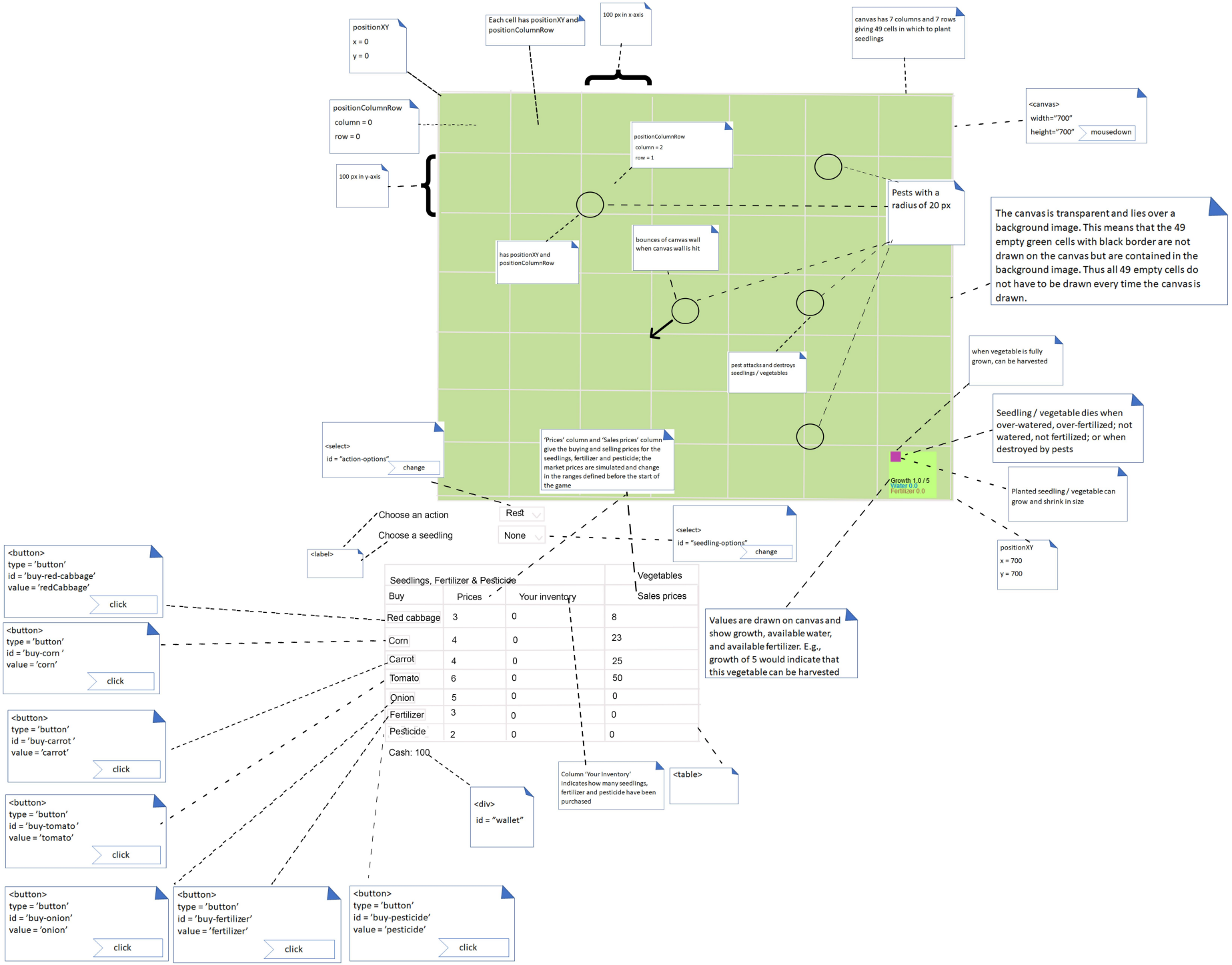

Farm Simulator
Player
start
buy seedling
buy fertilizer
buy pesticide
choose purchase price ranges
choose sales price ranges
add item to inventory
cash ≥ price
Simulate market prices
Track cash
Track inventory
plant seedling
has seedling in inventory AND cell is empty
cell shows seedling with available amount of water and fertilizer
water
fertilize
has fertilizer in inventory
eaten by pest OR incorrect amount of water or fertilizer
consumed by seedling / seedling grows
fully grown
destroy seedling
harvest
was harvested
clear cell
add market sell price to cash amount
Spawn pests
Move pests
fight pest
has pesticide in inventory
detect hit
pest hit
destroy pest

User Interface Scribble


<button>
type = “button“
value = “start“
id = “start“
click
Farm Simulator Settings
<h1>
Start simulation
Purchase prices (between 1 and 50)
<h3>

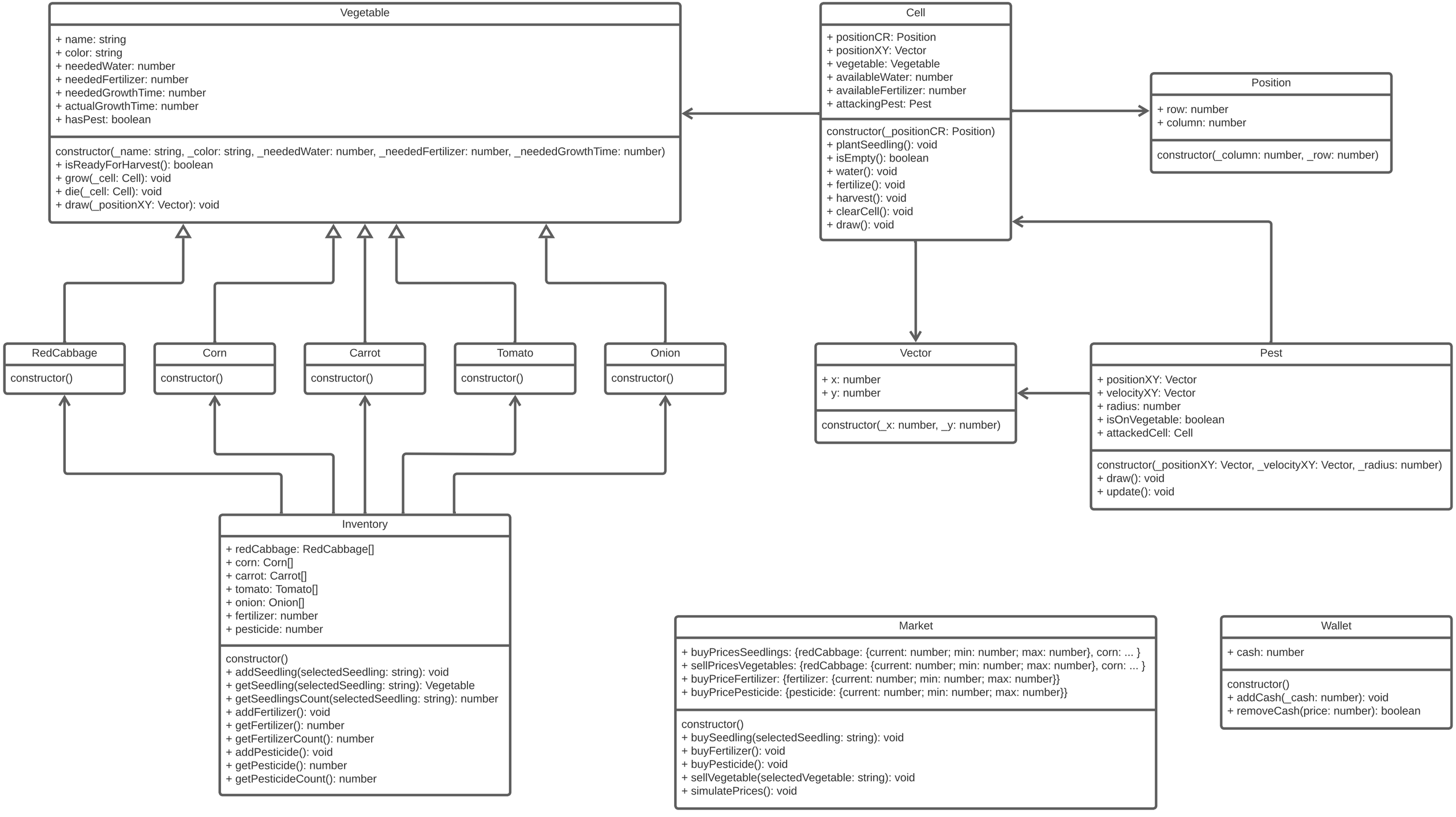
id = “invalid-input“
<label>
Red cabbage
Min:
Max:
Corn
Min:
Max:
Carrot
Min:
Max:
Tomato
Min:
Max:
Onion
Min:
Max:
Fertilizer
Min:
Max:
Pesticide
Min:
Max:
<input>
type = “number“
id = “purchase-redCabbage-max“
min = “1“
max = “50“
value = “3“
<input>
type = “number“
id = “purchase-redCabbage-min“
min = “1“
max = “50“
value = “1“
(Sales prices (between 1 and 50
<h3>
<label>
Red cabbage
Min:
Max:
Corn
Min:
Max:
Carrot
Min:
Max:
Tomato
Min:
Max:
Onion
Min:
Max:

id = “prices“
Note:
Always the same logic as above, only the values for the HTML attributes id and value change.
The remaining ids are:
"purchase-corn-min", "purchase-corn-max"
"purchase-carrot-min", "purchase-carrot-max"
"purchase-tomato-min", "purchase-tomato-max"
"purchase-onion-min", "purchase-onion-max"
"sell-redCabbage-min", "sell-redCabbage-max"
"sell-corn-min", "sell-corn-max"
"sell-carrot-min", "sell-carrot-max"
"sell-tomato-min", "sell-tomato-max"
"sell-onion-min", "sell-onion-max"
And the values for the HTML attribute value are the numbers written in the boxes.


positionXY
x = 0
y = 0
Each cell has positionXY and positionColumnRow
100 px in x-axis
canvas has 7 columns and 7 rows giving 49 cells in which to plant seedlings
positionColumnRow
column = 0
row = 0
100 px in y-axis
positionColumnRow
column = 2
row = 1
<canvas>
width="700"
height="700"
mousedown
Pests with a radius of 20 px
The canvas is transparent and lies over a background image. This means that the 49 empty green cells with black border are not drawn on the canvas but are contained in the background image. Thus all 49 empty cells do not have to be drawn every time the canvas is drawn.
has positionXY and positionColumnRow
bounces of canvas wall when canvas wall is hit
pest attacks and destroys seedlings / vegetables
when vegetable is fully grown, can be harvested
Seedling / vegetable dies when over-watered, over-fertilized; not watered, not fertilized; or when destroyed by pests
Planted seedling / vegetable can grow and shrink in size
Growth 1.0 / 5
Water 0.0
Fertilizer 0.0
positionXY
x = 700
y = 700
<select>
id = "action-options"
change
'Prices' column and 'Sales prices' column give the buying and selling prices for the seedlings, fertilizer and pesticide; the market prices are simulated and change in the ranges defined before the start of the game
Choose an action
Rest
Choose a seedling
None
<label>
<select>
id = "seedling-options"
change
Values are drawn on canvas and show growth, available water, and available fertilizer. E.g., growth of 5 would indicate that this vegetable can be harvested
Seedlings, Fertilizer & Pesticide
Vegetables
Buy
Prices
Your inventory
Sales prices
Red cabbage 3 0 8
Corn 4 0 23
Carrot 4 0 25
Tomato 6 0 50
Onion 5 0 0
Fertilizer 3 0 0
Pesticide 2 0 0
Cash: 100
Column 'Your Inventory' indicates how many seedlings, fertilizer and pesticide have been purchased
<table>

id = "wallet"
<button>
type = 'button'
id = 'buy-red-cabbage'
value = 'redCabbage'
click
<button>
type = 'button'
id = 'buy-corn '
value = 'corn'
click
<button>
type = 'button'
id = 'buy-carrot '
value = 'carrot'
click
<button>
type = 'button'
id = 'buy-tomato '
value = 'tomato'
click
<button>
type = 'button'
id = 'buy-onion'
value = 'onion'
click
<button>
type = 'button'
id = 'buy-fertilizer'
value = 'fertilizer'
click
<button>
type = 'button'
id = 'buy-pesticide'
value = 'pesticide'
click

Class Diagram


Vegetable
+ name: string
+ color: string
+ neededWater: number
+ neededFertilizer: number
+ neededGrowthTime: number
+ actualGrowthTime: number
+ hasPest: boolean
constructor(_name: string, _color: string, _neededWater: number, _neededFertilizer: number, _neededGrowthTime: number)
+ isReadyForHarvest(): boolean
+ grow(_cell: Cell): void
+ die(_cell: Cell): void
+ draw(_positionXY: Vector): void
Cell
+ positionCR: Position
+ positionXY: Vector
+ vegetable: Vegetable
+ availableWater: number
+ availableFertilizer: number
+ attackingPest: Pest
constructor(_positionCR: Position)
+ plantSeedling(): void
+ isEmpty(): boolean
+ water(): void
+ fertilize(): void
+ harvest(): void
+ clearCell(): void
+ draw(): void
Position
+ row: number
+ column: number
constructor(_column: number, _row: number)
RedCabbage
constructor()
Corn
constructor()
Carrot
constructor()
Tomato
constructor()
Onion
constructor()
Vector
+ x: number
+ y: number
constructor(_x: number, _y: number)
Pest
+ positionXY: Vector
+ velocityXY: Vector
+ radius: number
+ isOnVegetable: boolean
+ attackedCell: Cell
constructor(_positionXY: Vector, _velocityXY: Vector, _radius: number)
+ draw(): void
+ update(): void
Inventory
+ redCabbage: RedCabbage[]
+ corn: Corn[]
+ carrot: Carrot[]
+ tomato: Tomato[]
+ onion: Onion[]
+ fertilizer: number
+ pesticide: number
constructor()
+ addSeedling(selectedSeedling: string): void
+ getSeedling(selectedSeedling: string): Vegetable
+ getSeedlingsCount(selectedSeedling: string): number
+ addFertilizer(): void
+ getFertilizer(): number
+ getFertilizerCount(): number
+ addPesticide(): void
+ getPesticide(): number
+ getPesticideCount(): number
Market
+ buyPricesSeedlings: {redCabbage: {current: number; min: number; max: number}, corn: ... }
+ sellPricesVegetables: {redCabbage: {current: number; min: number; max: number}, corn: ... }
+ buyPriceFertilizer: {fertilizer: {current: number; min: number; max: number}}
+ buyPricePesticide: {pesticide: {current: number; min: number; max: number}}
constructor()
+ buySeedling(selectedSeedling: string): void
+ buyFertilizer(): void
+ buyPesticide(): void
+ sellVegetable(selectedVegetable: string): void
+ simulatePrices(): void
Wallet
+ cash: number
constructor()
+ addCash(_cash: number): void
+ removeCash(price: number): boolean

Activity Diagrams

Settings

register load listener

load window

handleLoad

mouse click

setMarketPrices

Globals

inputs: HTMLCollectionOf<HTMLInputElement>

handleLoad

inputs = document
.getElementById("prices")
.getElementsByTagName("input")

install button listener

setMarketPrices

_event

isValid = isValidateInput

[isValid == false]

i = 0

[i is greater than or equal to inputs.length]

window.location.href = "./simulated-farm.html"
(will open this page and thus start the game)

localStorage.setItem(inputs[i].id, inputs[i].value)

increase i by 1

isValidateInput

isValid = true
validationMessage = "invalid ids: "
i = 0

[i is greater than or equal to inputs.length]

isValid

min = parseInt(inputs[i].value);
max = parseInt(inputs[i + 1].value);

[min is greater than or equal to max
OR min is NaN OR max is NaN]

isValid = false
validationMessage += `${inputs[i].id} `

increase i by 1

Main

register load listener

load window

handleLoad

every 2 sec.

createPests

every 5 sec.

simulateMarket

every 20 ms

simulateAndAnimateFarm

mouse button down

getClickedCell

mouse button down

farm

change

getSelectedAction

change

getSelectedSeedling

mouse click

buySeedling

mouse click

buyFertilizer

mouse click

buyPesticide

mouse button down

sprayPesticide

handleLoad

get and configure canvas

get and configure RenderingContext

create cells (the farm land)

create player's inventory

create market

create pests

get table displaying market prices and inventory

get div displaying player's cash

install listeners

simulate market

simulate and animate farm

createPests

[exist more than 4 pests]

numberPests = getRandomInt
i = 0

[i is greater than or equal to numberPests]

positionXY = new Vector(20, 20)
velocityXY = new Vector(getRandomInt, getRandomInt)
pest = new Pest(positionXY, velocityXY)

add (push) pest to pests

increase i by 1

simulateMarket

market.simulatePrices()

simulateAndAnimateFarm

clear canvas

grow all vegetables

draw all vegetables

move all pests

draw all pests

detect if pest is flying over an uninfested vegetable & handle attack

refresh HTML prices and inventory table

refresh HTML cash display

getClickedCell

_event

column = Math.foor((xMousePositionOnCanvas / canvasWidth) * numberColumns)
row = Math.foor((yMousePositionOnCanvas / canvasHeight) * numberRows)
selectedCell = cells[column][row]

farm

_event

what is the selectedAction?

[plant]

plant seedling in cell

[water]

water cell

[fertilize]

fertilize cell

[harvest]

harvest vegetable in cell

getSelectedAction

_event

selectedAction = (<HTMLInputElement>_event.target).value

getSelectedSeedling

_event

selectedSeedling = (<HTMLInputElement>_event.target).value

sprayPesticide

_event

[selected action is not pest control OR has no pesticide in inventory]

get pesticide from inventory

get clicked positionXY on canvas

get positionXY of each pest

calculate for each pest how far away pest is from clicked position, use getDistanceBetween2Vectors

determine if pest was hit, use isHit

[no hit]

remove pest from pests array

buySeedling

_event

market.buySeedling(selectedSeedling)

buyFertilizer

_event

market.buyFertilizer()

buyPesticide

_event

market.buyPesticide()

Globals

crc2: CanvasRenderingContext2D
canvas: HTMLCanvasElement
canvasWidth: number
canvasHeight: number
numberRows: number
numberColumns: number

cells: Cell[][]
selectedCell: Cell
selectedAction: string
selectedSeedling: string
pests: Pest[]
inventory: Inventory
market: Market
wallet: Wallet
cashDisplay: HTMLElement
table: HTMLTableElement

getDistanceBetween2Vectors

_sprayXY: Vector, _pestXY: Vector

distance = sqrt(_sprayXY.x - _pestXY.x)^2 + (_sprayXY.y - _pestXY.y)^2)

distance

isHit

_radiusPest: number, _distance: number

[_radiusPest is greater than or equal to _distance]

wasHit = true

wasHit = false

wasHit

Vegetable

constructor

_name: string
_color: string
_neededWater: number
_neededFertilizer: number
_neededGrowthTime: number

set this.name to _name, this.color to _color,
this.neededWater to _neededWater,
this.neededFertilizer to _neededFertilizer, and
this.neededGrowthTime to _neededGrowthTime

isReadyForHarvest

[actualGrowthTime is greater than or equal to neededGrowthTime]

result = true

result = false

result

draw

_positionXY: Vector

growth = actualGrowthTime / neededGrowthTime
crc2.fillStyle = color

crc2.fillRect(_positionXY.x, _positionXY.y,
CELL_SIZE * growth, CELL_SIZE * growth)

use fillText to plot actualGrowthTime

grow

_cell: Cell

consume = 0.001

[_cell.availableWater > consume AND
_cell.availableFertilizer > consume]

_cell.availableWater -= consume
_cell.availableFertilizer -= consume
actualGrowthTime += 0.005

actualGrowthTime -= 0.0001

[hasPest]

actualGrowthTime -= 0.001

[_cell.availableWater > 10 OR
_cell.availableFertilizer > 10 OR
actualGrowthTime < 0 OR
this.actualGrowthTime >= this.neededGrowthTime * 2]

die(_cell)

die

_cell: Cell

_cell.clearCell()

RedCabbage

constructor

super("redCabbage", "#BF45AB", 5, 5, 5)

Carrot

constructor

super("carrot", "#F2921D", 2, 2, 2)

Tomato

constructor

super("tomato", "#F24130", 6, 6, 6)

Onion

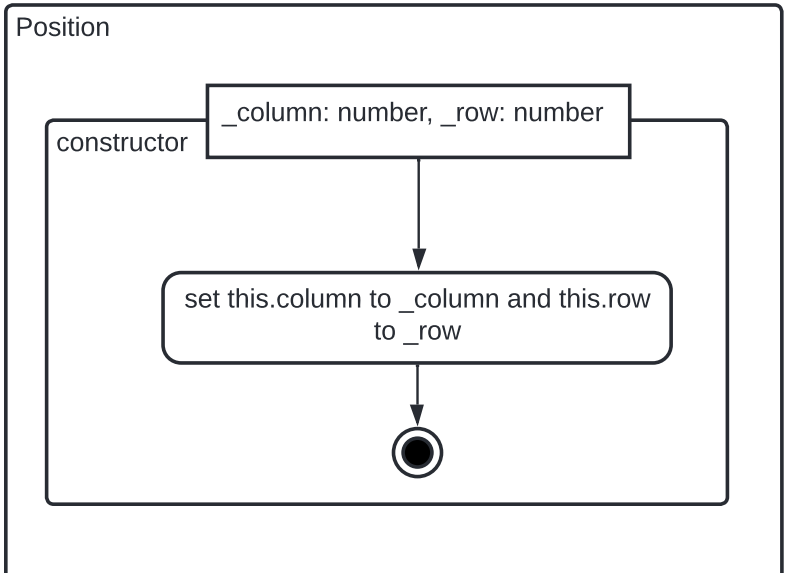
constructor

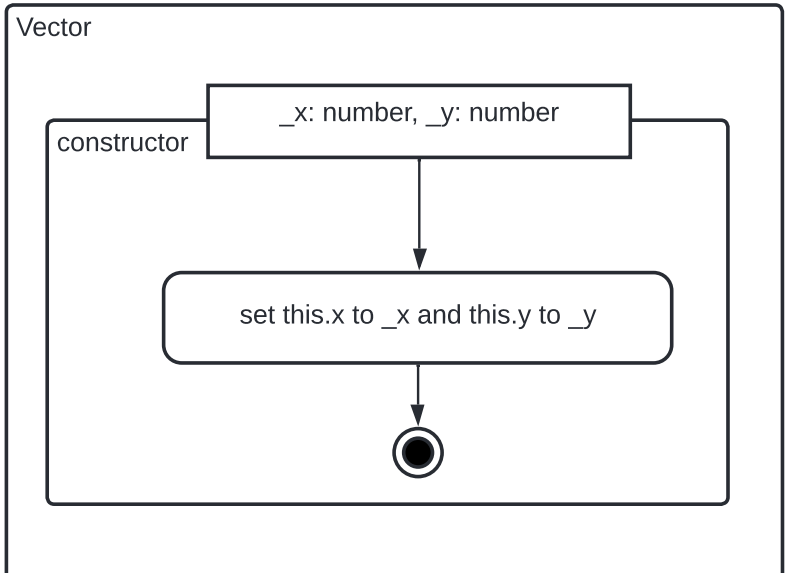
super("onion", "#F2F2F2", 4, 4, 4)

Corn

constructor

super("corn", "#F2C029", 3, 3, 3)


Position
constructor
_column: number, _row: number
set this.column to _column and this.row to _row


Vector
constructor
_x: number, _y: number
set this.x to _x and this.y to _y

Cell

constructor

_positionCR: Position

set this.positionCR to _positionCR

this.positionXY = new Vector(
_positionCR.column * CELL_SIZE + 5,
_positionCR.row * CELL_SIZE + 5)

harvest

isHarvestReady =
this.vegetable.isReadyForHarvest()

[isHarvestReady == true
AND selectedCell is not
empty]

sell vegetable on market for
market sell price

clear the cell with the use of
this.clearCell()

plantSeedling

[has seedling
in inventory
AND selectedCell
is empty]

this.vegetable =
inventory.getSeedling(selectedSeedling)

draw

[selectedCell
is empty]

this.vegetable.draw(this.positionXY)

use fillText to plot available water

use fillText to plot available fertilizer

clearCell

this.vegetable = null;
this.availableWater = 0;
this.availableFertilizer = 0;
this.attackingPest = null;

water

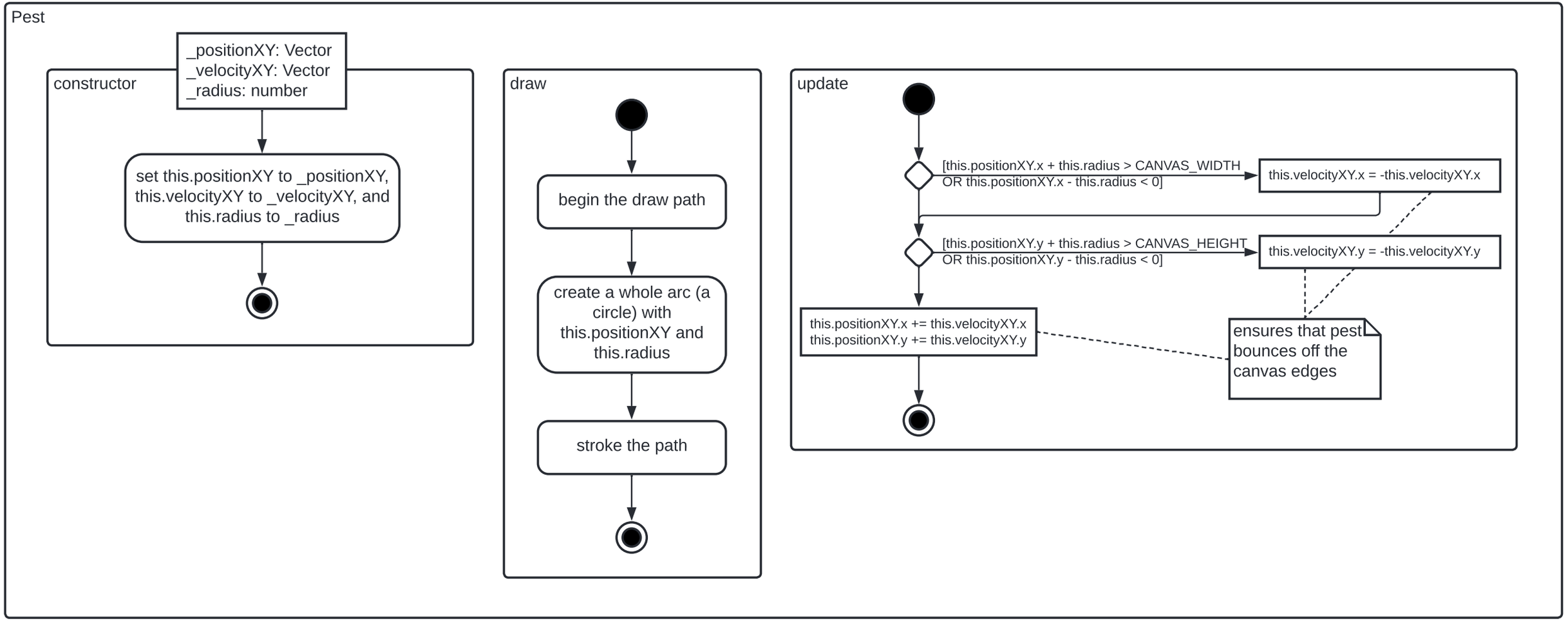
this.availableWater += 1

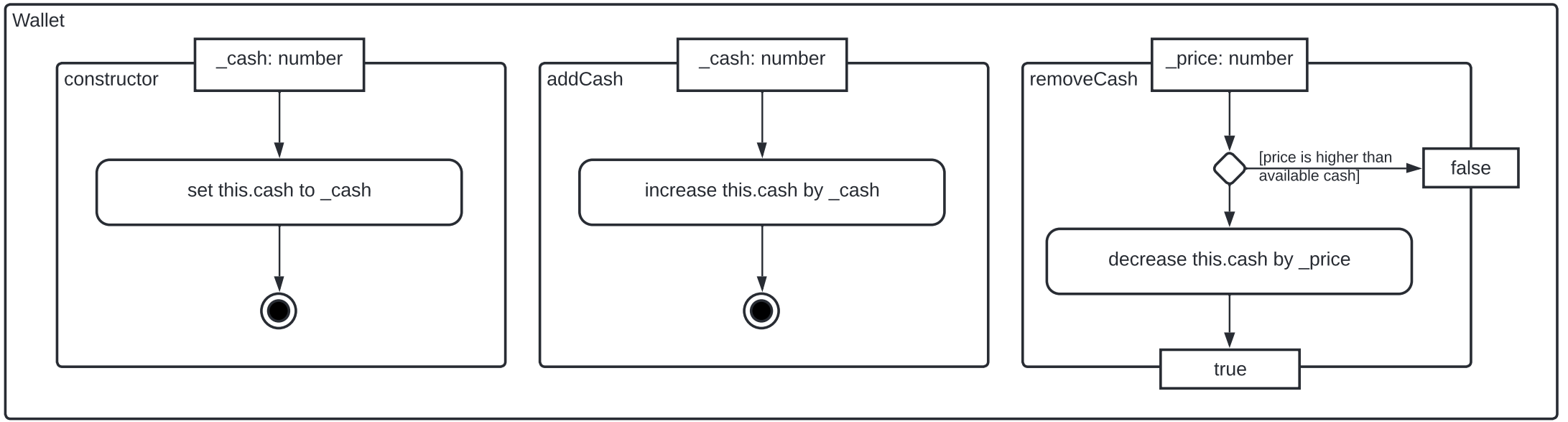
[selectedCell is empty]

fertilize

[has fertilizer in inventory
AND selectedCell is not
empty]

this.availableFertilizer +=
inventory.getFertilizer()


Pest
constructor
_positionXY: Vector
_velocityXY: Vector
_radius: number
set this.positionXY to _positionXY, this.velocityXY to _velocityXY, and this.radius to _radius
draw
begin the draw path
create a whole arc (a circle) with this.positionXY and this.radius
stroke the path
update
[this.positionXY.x + this.radius > CANVAS_WIDTH OR this.positionXY.x - this.radius < 0]
this.velocityXY.x = -this.velocityXY.x
[this.positionXY.y + this.radius > CANVAS_HEIGHT OR this.positionXY.y - this.radius < 0]
this.velocityXY.y = -this.velocityXY.y
this.positionXY.x += this.velocityXY.x
this.positionXY.y += this.velocityXY.y
ensures that pest bounces off the canvas edges


Wallet
constructor
_cash: number
set this.cash to _cash
addCash
_cash: number
increase this.cash by _cash
removeCash
_price: number
[price is higher than available cash]
false
decrease this.cash by _price
true

Inventory

addSeedling

_selectedSeedling: string

what is the _selectedSeedling?

[redCabbage] this.redCabbage.push(new RedCabbage())

[corn] this.corn.push(new Corn())

[carrot] this.carrot.push(new Carrot())

[tomato] this.tomato.push(new Tomato())

[onion] this.onion.push(new Onion())

addFertilizer

increase this.fertilizer by 1

addPesticide

increase this.pesticide by 1

getSeedling

_selectedSeedling: string

what is the _selectedSeedling?

[redCabbage] this.redCabbage.pop() → Vegetable

[corn] this.corn.pop() → Vegetable

[carrot] this.carrot.pop() → Vegetable

[tomato] this.tomato.pop() → Vegetable

[onion] this.onion.pop() → Vegetable

undefined

getFertilizer

[has fertilizer] decrease this.fertilizer by 1

0

1

getPesticide

[has pesticide] decrease this.pesticide by 1

0

1

getFertilizerCount

this.fertilizer

number

getPesticideCount

this.pesticide

number

getSeedlingsCount

_selectedSeedling: string

what is the _selectedSeedling?

[redCabbage] this.redCabbage.length → number

[corn] this.corn.length → number

[carrot] this.carrot.length → number

[tomato] this.tomato.length → number

[onion] this.onion.length → number

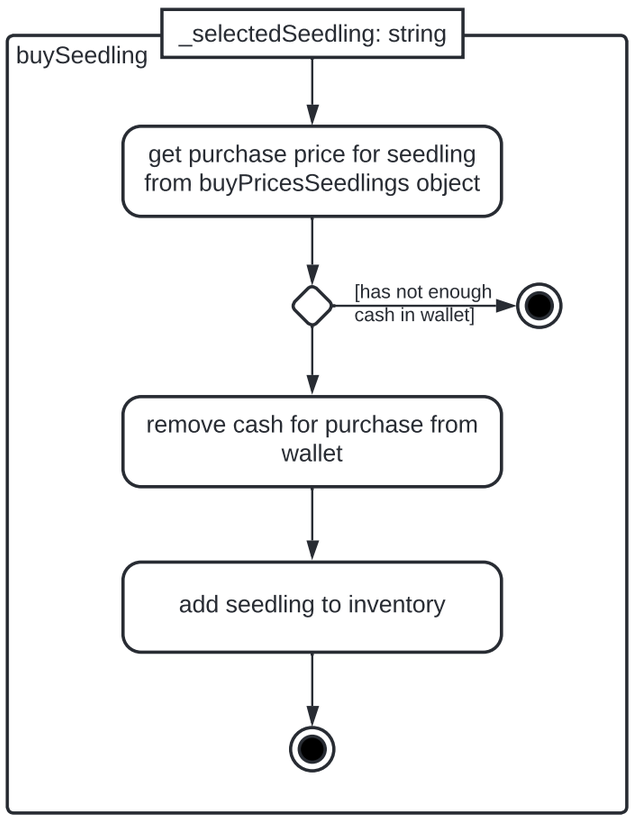
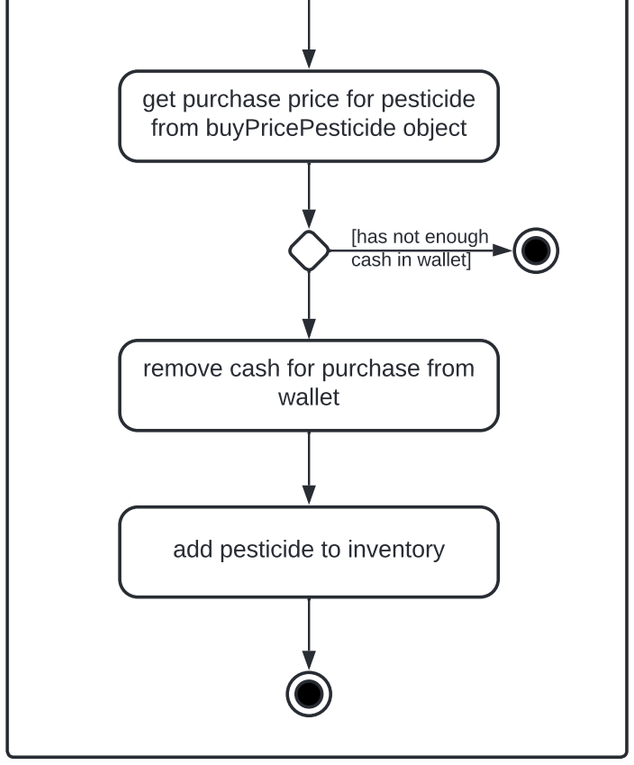
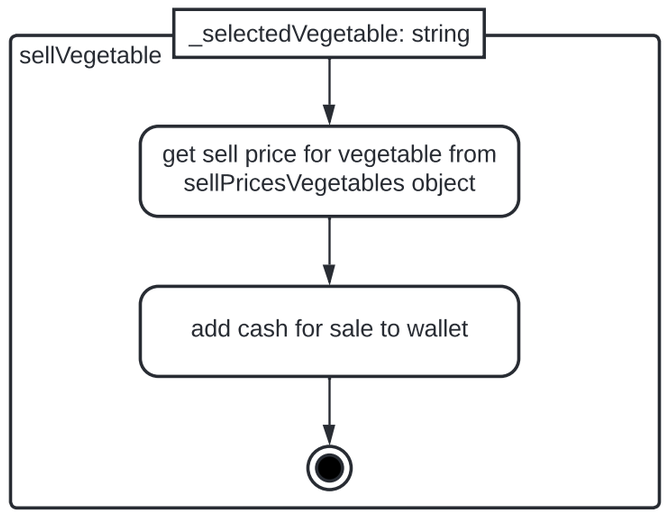
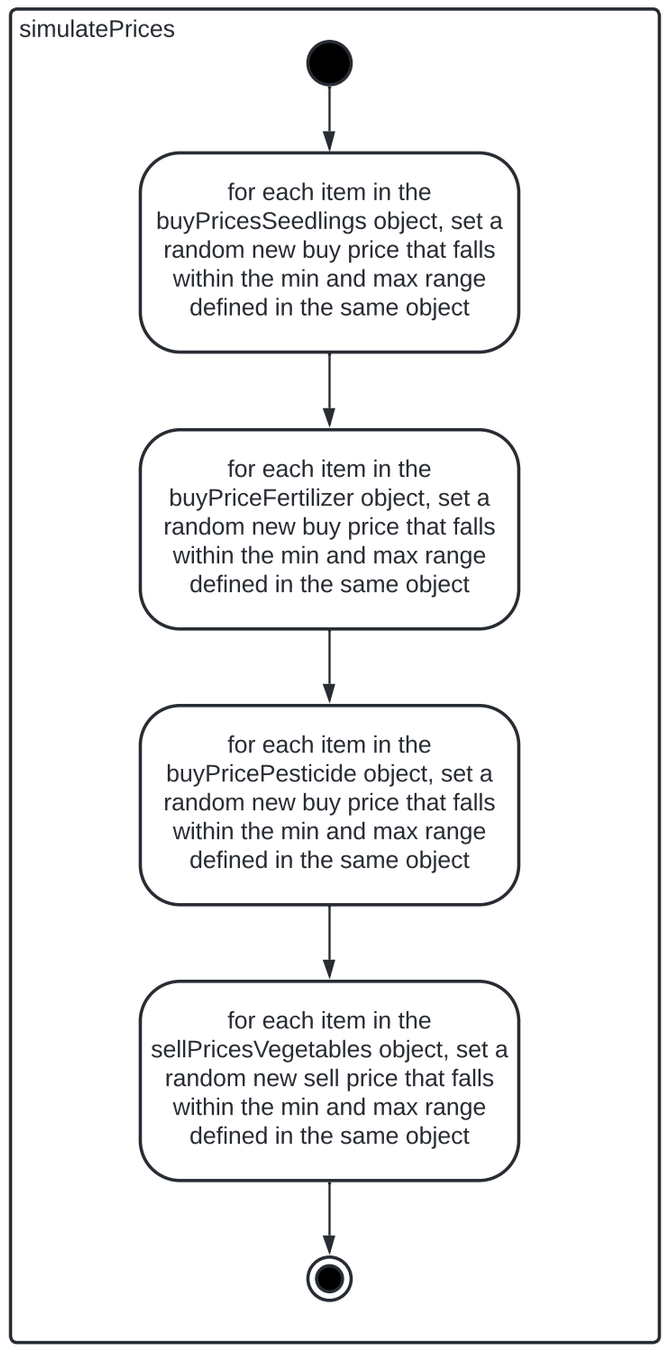
undefined

Market

constructor

```
this.buyPricesSeedlings = {
  redCabbage: {
    current: 1,
    min: parseInt(localStorage.getItem("purchase-redCabbage-min")),
    max: parseInt(localStorage.getItem("purchase-redCabbage-max")),
  },
  corn: {
    current: 2,
    min: parseInt(localStorage.getItem("purchase-corn-min")),
    max: parseInt(localStorage.getItem("purchase-corn-max")),
  },
  carrot: {
    current: 3,
    min: parseInt(localStorage.getItem("purchase-carrot-min")),
    max: parseInt(localStorage.getItem("purchase-carrot-max")),
  },
  tomato: {
    current: 4,
    min: parseInt(localStorage.getItem("purchase-tomato-min")),
    max: parseInt(localStorage.getItem("purchase-tomato-max")),
  },
  onion: {
    current: 5,
    min: parseInt(localStorage.getItem("purchase-onion-min")),
    max: parseInt(localStorage.getItem("purchase-onion-max")),
  },
}
```

```
this.sellPricesVegetables = {
  redCabbage: {
    current: 10,
    min: parseInt(localStorage.getItem("sell-redCabbage-min")),
    max: parseInt(localStorage.getItem("sell-redCabbage-max")),
  },
  corn: {
    current: 20,
    min: parseInt(localStorage.getItem("sell-corn-min")),
    max: parseInt(localStorage.getItem("sell-corn-max")),
  },
  carrot: {
    current: 30,
    min: parseInt(localStorage.getItem("sell-carrot-min")),
    max: parseInt(localStorage.getItem("sell-carrot-max")),
  },
  tomato: {
    current: 40,
    min: parseInt(localStorage.getItem("sell-tomato-min")),
    max: parseInt(localStorage.getItem("sell-tomato-max")),
  },
  onion: {
    current: 50,
    min: parseInt(localStorage.getItem("sell-onion-min")),
    max: parseInt(localStorage.getItem("sell-onion-max")),
  },
}
```

```
this.buyPriceFertilizer = {
  fertilizer: {
    current: 2,
    min: parseInt(localStorage.getItem("purchase-fertilizer-min")),
    max: parseInt(localStorage.getItem("purchase-fertilizer-max")),
  },
}
```

```
this.buyPricePesticide = {
  pesticide: {
    current: 2,
    min: parseInt(localStorage.getItem("purchase-pesticide-min")),
    max: parseInt(localStorage.getItem("purchase-pesticide-max")),
  },
}
```