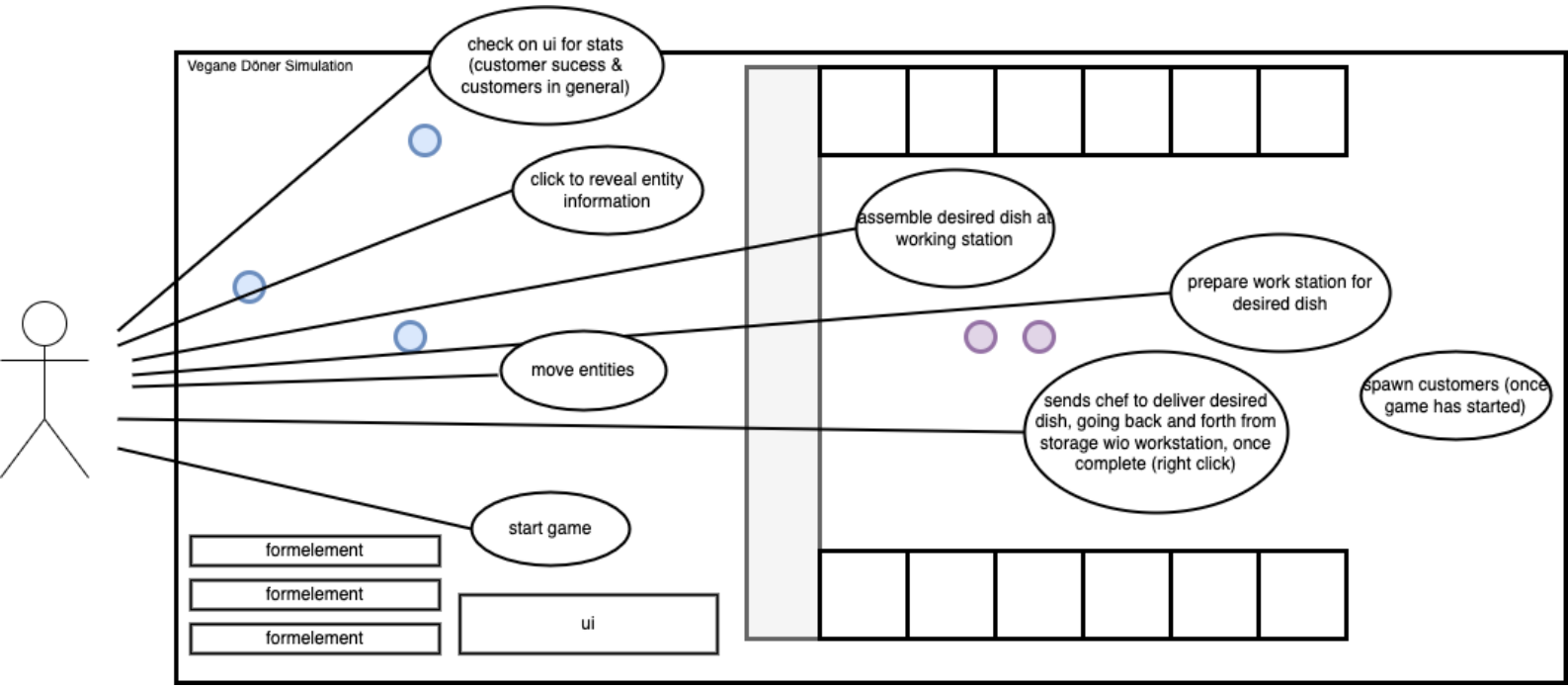
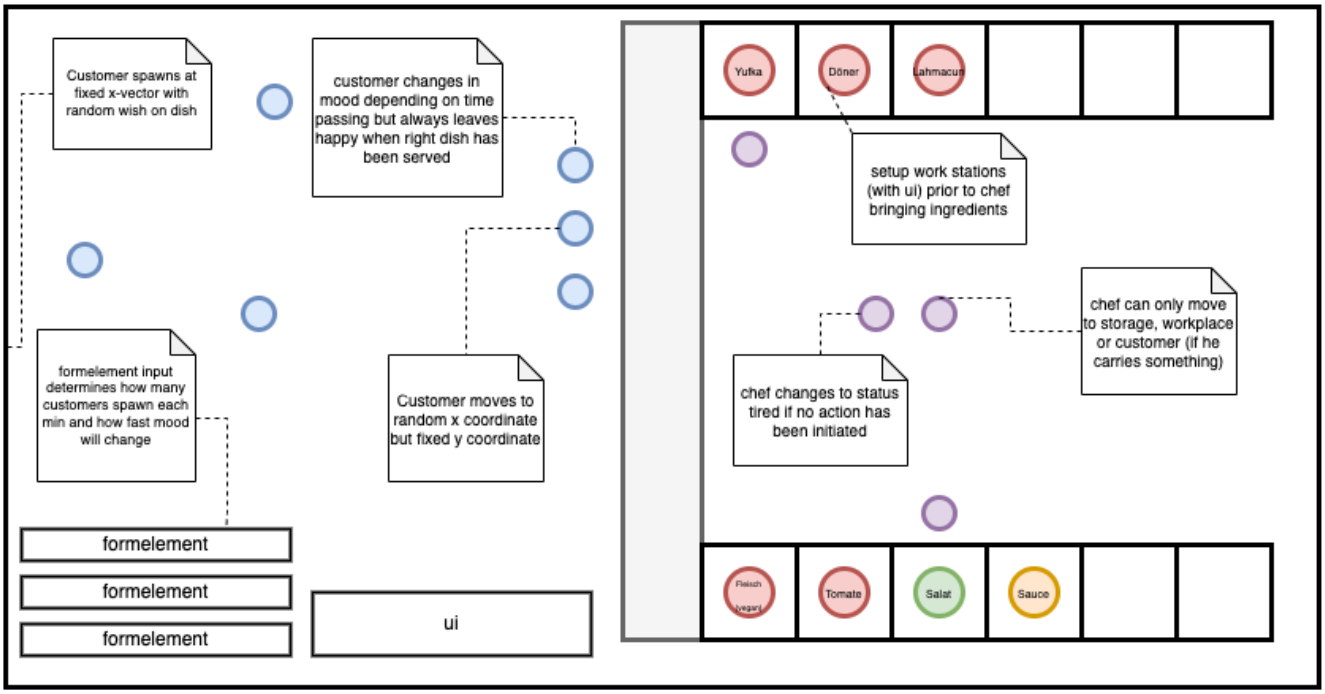
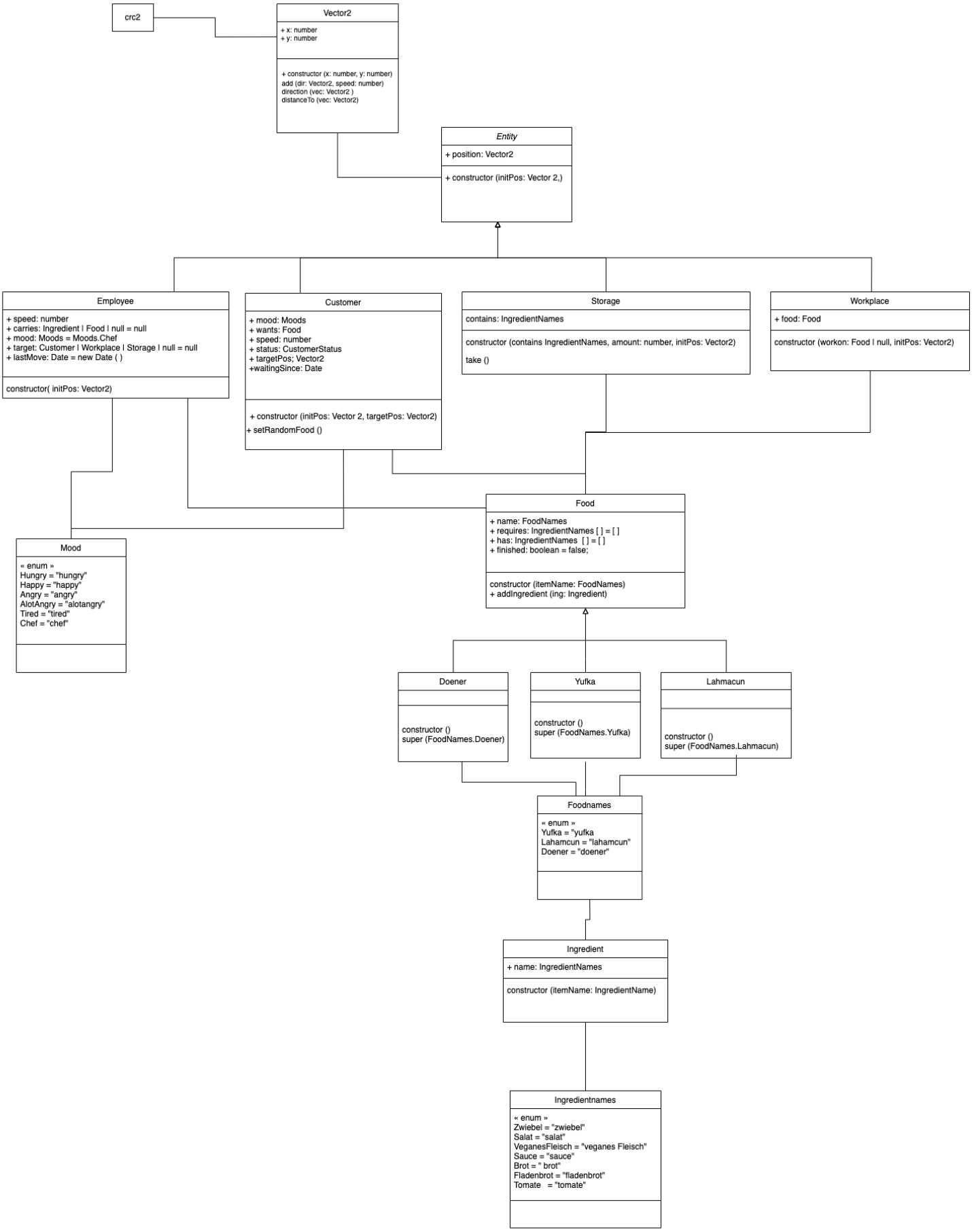
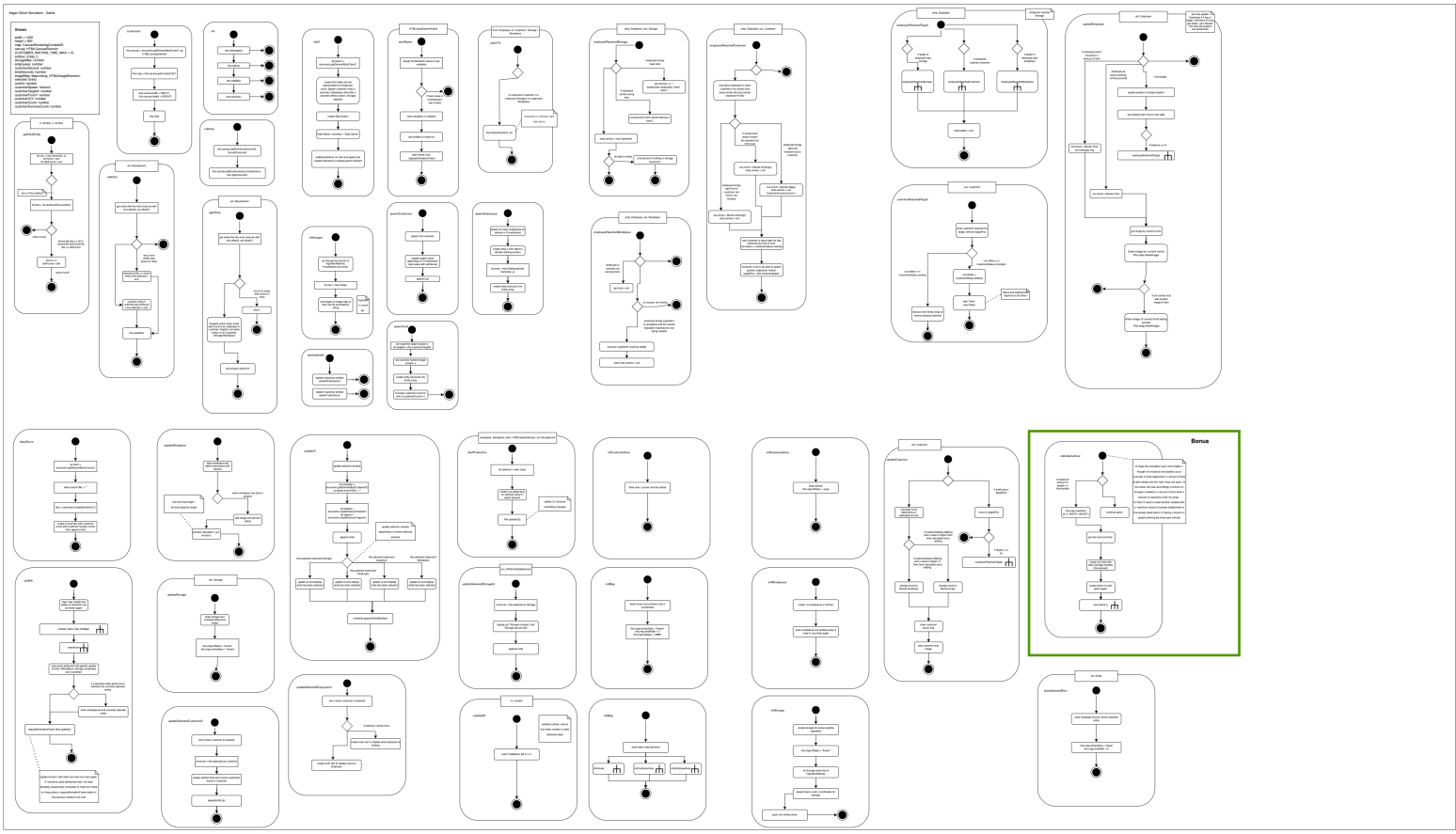
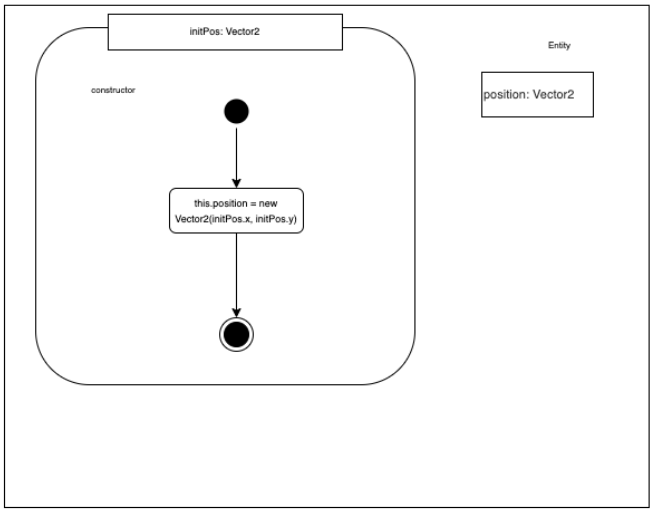
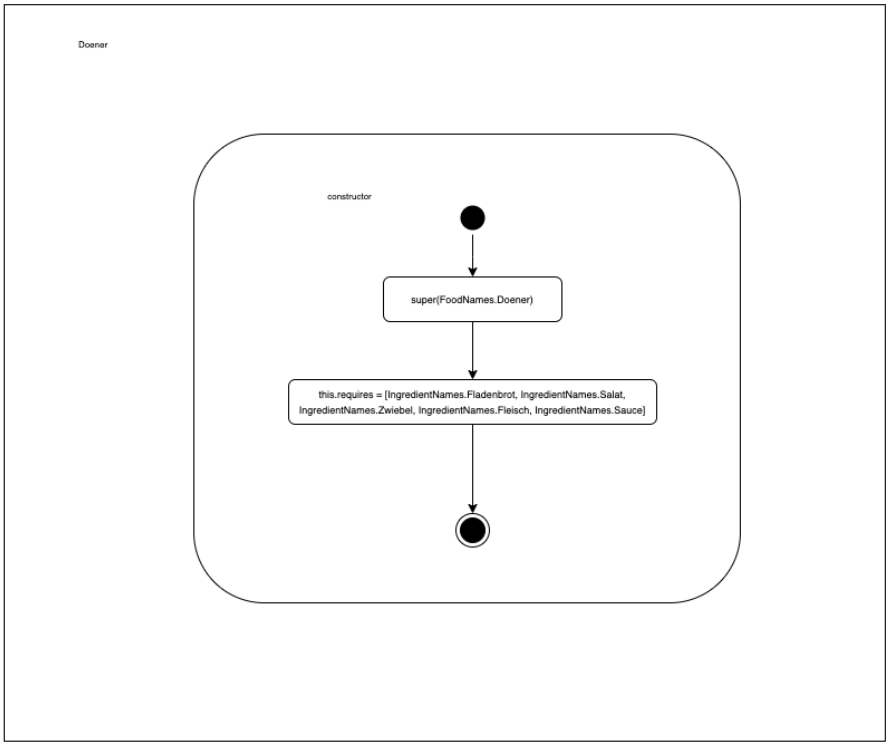

Vegane Döner Simulation
check on ui for stats (customer sucess & customers in general)
click to reveal entity information
move entities
start game
formelement
formelement
formelement
ui
assemble desired dish at working station
prepare work station for desired dish
sends chef to deliver desired dish, going back and forth from storage wio workstation, once complete (right click)
spawn customers (once game has started)

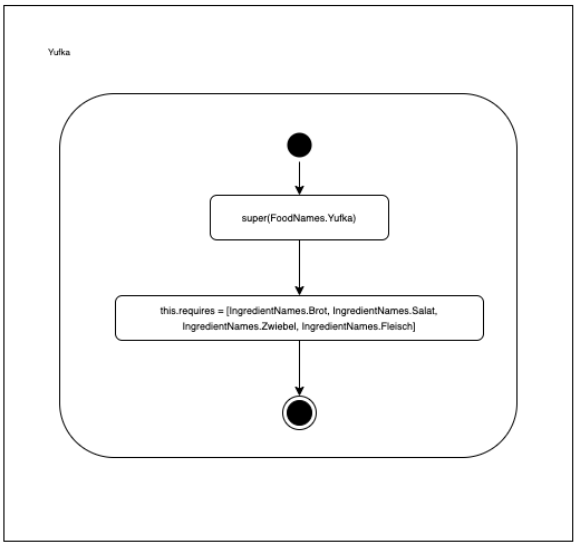

Customer spawns at fixed x-vector with random wish on dish
customer changes in mood depending on time passing but always leaves happy when right dish has been served
formelement input determines how many customers spawn each min and how fast mood will change
Customer moves to random x coordinate but fixed y coordinate
formelement
formelement
formelement
ui
Yufka
Döner
Lahmacun
setup work stations (with ui) prior to chef bringing ingredients
chef can only move to storage, workplace or customer (if he carries something)
chef changes to status tired if no action has been initiated
Fleisch (vegan)
Tomate
Salat
Sauce

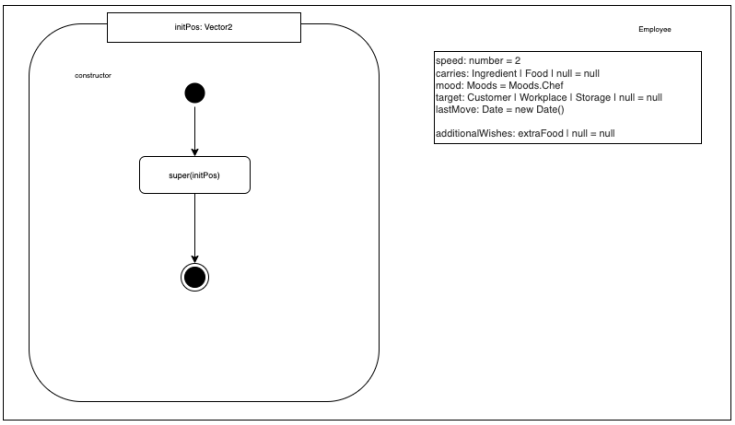

crc2
Vector2
+ x: number
+ y: number
+ constructor (x: number, y: number)
add (dir: Vector2, speed: number)
direction (vec: Vector2)
distanceTo (vec: Vector2)
Entity
+ position: Vector2
+ constructor (initPos: Vector 2,)
Employee
+ speed: number
+ carries: Ingredient I Food I null = null
+ mood: Moods = Moods.Chef
+ target: Customer I Workplace I Storage I null = null
+ lastMove: Date = new Date ( )
constructor(initPos: Vector2)
Customer
+ mood: Moods
+ wants: Food
+ speed: number
+ status: CustomerStatus
+ targetPos; Vector2
+waitingSince: Date
+ constructor (initPos: Vector 2, targetPos: Vector2)
+ setRandomFood ()
Storage
contains: IngredientNames
constructor (contains IngredientNames, amount: number, initPos: Vector2)
take ()
Workplace
+ food: Food
constructor (workon: Food I null, initPos: Vector2)
Mood
« enum »
Hungry = "hungry"
Happy = "happy"
Angry = "angry"
AlotAngry = "alotangry"
Tired = "tired"
Chef = "chef"
Food
+ name: FoodNames
+ requires: IngredientNames [ ] = [ ]
+ has: IngredientNames [ ] = [ ]
+ finished: boolean = false;
constructor (itemName: FoodNames)
+ addIngredient (ing: Ingredient)
Doener
constructor ()
super (FoodNames.Doener)
Yufka
constructor ()
super (FoodNames.Yufka)
Lahmacun
constructor ()
super (FoodNames.Lahmacun)
Foodnames
« enum »
Yufka = "yufka
Lahamcun = "lahamcun"
Doener = "doener"
Ingredient
+ name: IngredientNames
constructor (itemName: IngredientName)
Ingredientnames
« enum »
Zwiebel = "zwiebel"
Salat = "salat"
VeganesFleisch = "veganes Fleisch"
Sauce = "sauce"
Brot = " brot"
Fladenbrot = "fladenbrot"
Tomate = "tomate"

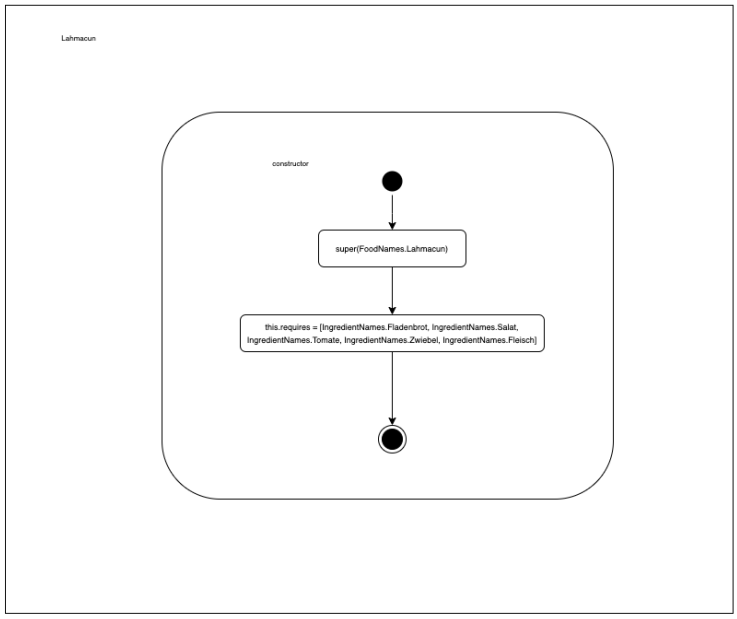

Vegan Dinner Simulation - Game
Globals
width = 1200
height = 800
canvas: HTMLRenderingContext2D
tables: HTMLCanvasElement
SCATTERER_MIN TIME, TIME_MAX = 10
dragTable: number
dragOffset: number
customerCount: number
customerSecond: number
customerTimer: number
imageMap: Map<string, HTMLImageElement>
selected: Entity
hovered: Entity
customerSpawn: Tween
customerSpawnTimer: number
customerTotal?: number
customerCount: number
customerSuccessCount: number
addCustomer
updateCustomer
updateDragTable
updateDragOver
updateDragOverCustomer
updateDragOverEmployee
updateDragOverTable
updateDragOverTrash
updateDragOverWorkplace
updateDragOverEmployee2
updateDragOverTable2
updateDragOverTrash2
updateDragOverWorkplace2
updateDragOverCustomer2
updateDragOverEmployee3
updateDragOverTable3
updateDragOverTrash3
updateDragOverWorkplace3
updateDragOverCustomer3
updateDragOverEmployee4
updateDragOverTable4
updateDragOverTrash4
updateDragOverWorkplace4
updateDragOverCustomer4
updateDragOverEmployee5
updateDragOverTable5
updateDragOverTrash5
updateDragOverWorkplace5
updateDragOverCustomer5
updateDragOverEmployee6
updateDragOverTable6
updateDragOverTrash6
updateDragOverWorkplace6
updateDragOverCustomer6
updateDragOverEmployee7
updateDragOverTable7
updateDragOverTrash7
updateDragOverWorkplace7
updateDragOverCustomer7
updateDragOverEmployee8
updateDragOverTable8
updateDragOverTrash8
updateDragOverWorkplace8
updateDragOverCustomer8
updateDragOverEmployee9
updateDragOverTable9
updateDragOverTrash9
updateDragOverWorkplace9
updateDragOverCustomer9

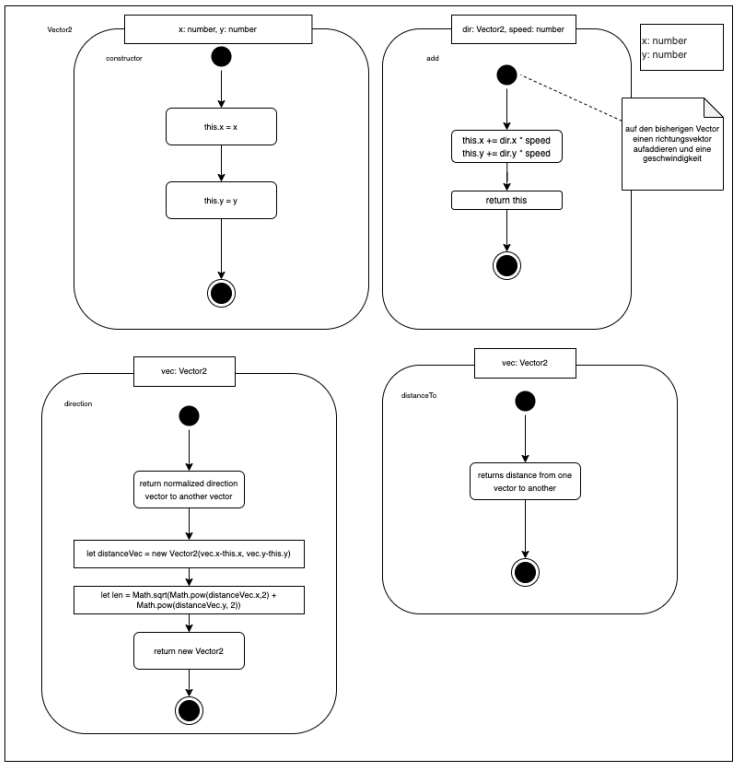

initPos: Vector2
constructor
this.position = new Vector2(initPos.x, initPos.y)
Entity
position: Vector2

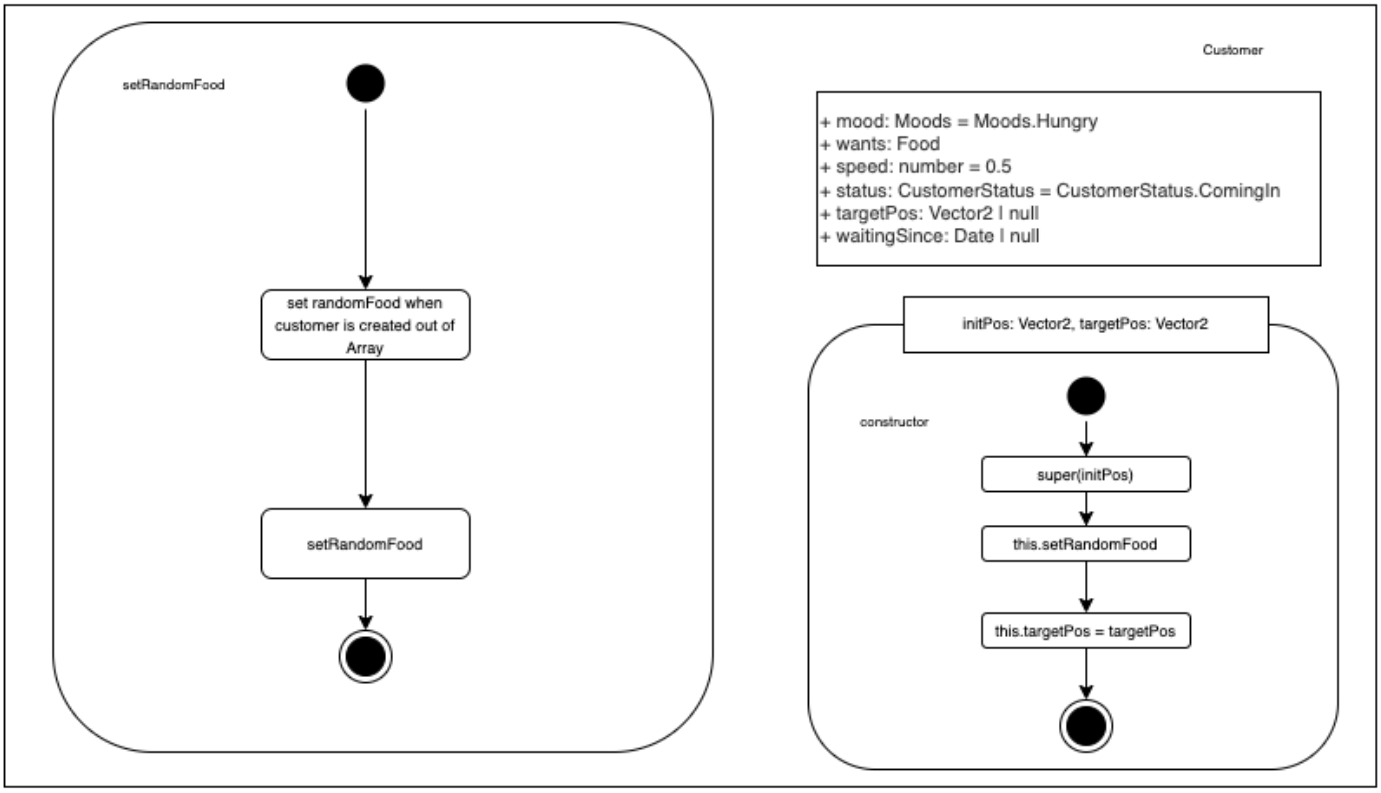

Doener
constructor
super(FoodNames.Doener)
this.requires = [IngredientNames.Fladenbrot, IngredientNames.Salat, IngredientNames.Zwiebel, IngredientNames.Fleisch, IngredientNames.Sauce]

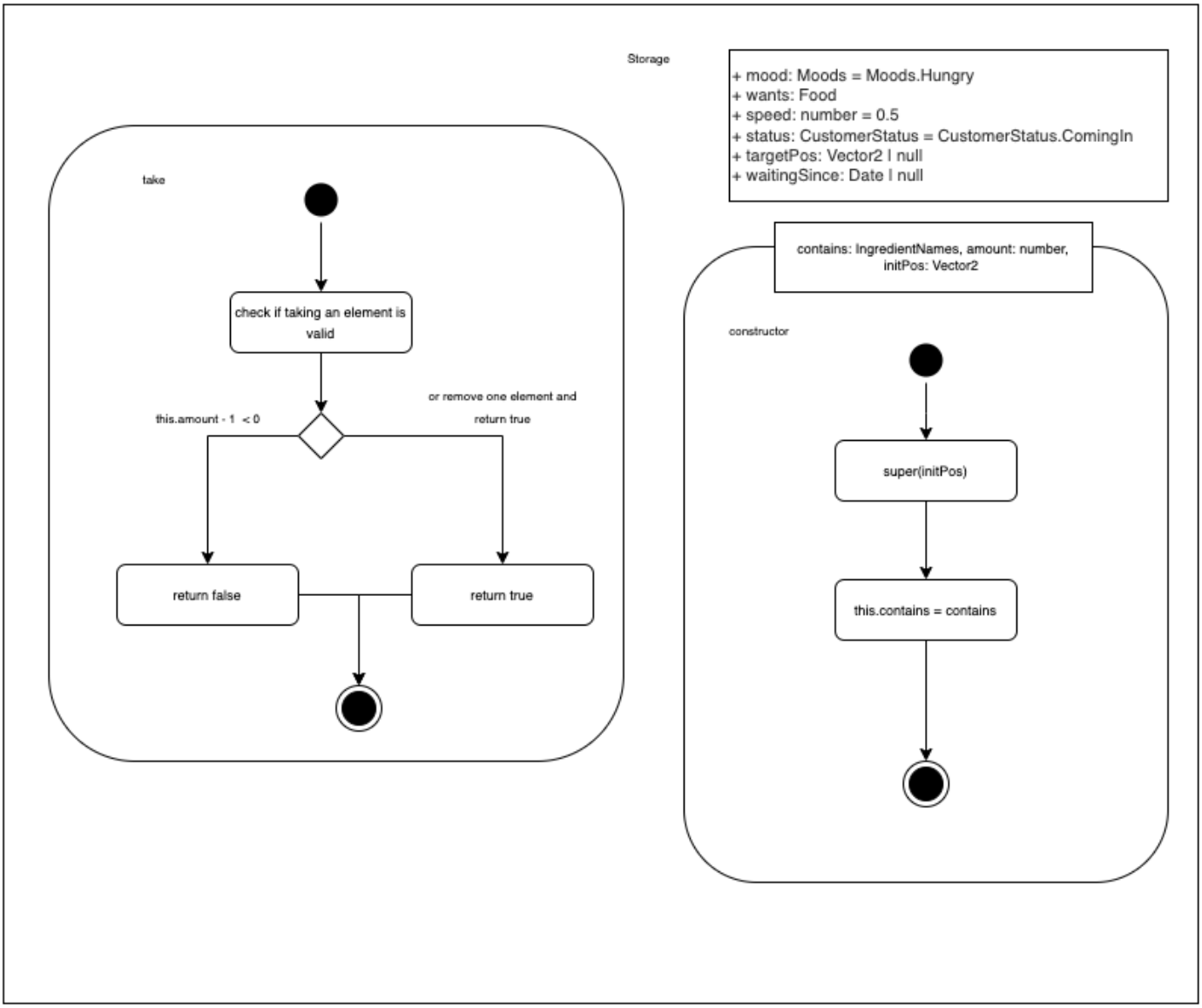

Yufka
super(FoodNames.Yufka)
this.requires = [IngredientNames.Brot, IngredientNames.Salat, IngredientNames.Zwiebel, IngredientNames.Fleisch]


initPos: Vector2
constructor
super(initPos)
Employee
speed: number = 2
carries: Ingredient | Food | null = null
mood: Moods = Moods.Chef
target: Customer | Workplace | Storage | null = null
lastMove: Date = new Date()
additionalWishes: extraFood | null = null


Lahmacun
constructor
super(FoodNames.Lahmacun)
this.requires = [IngredientNames.Fladenbrot, IngredientNames.Salat, IngredientNames.Tomate, IngredientNames.Zwiebel, IngredientNames.Fleisch]


Vector2
x: number, y: number
constructor
this.x = x
this.y = y
dir: Vector2, speed: number
add
this.x += dir.x * speed
this.y += dir.y * speed
return this
x: number
y: number
auf den bisherigen Vector einen richtungsvektor aufaddieren und eine geschwindigkeit
vec: Vector2
direction
return normalized direction vector to another vector
let distanceVec = new Vector2(vec.x-this.x, vec.y-this.y)
let len = Math.sqrt(Math.pow(distanceVec.x,2) + Math.pow(distanceVec.y, 2))
return new Vector2
vec: Vector2
distanceTo
returns distance from one vector to another


setRandomFood
set randomFood when customer is created out of Array
setRandomFood
Customer
+ mood: Moods = Moods.Hungry
+ wants: Food
+ speed: number = 0.5
+ status: CustomerStatus = CustomerStatus.ComingIn
+ targetPos: Vector2 | null
+ waitingSince: Date | null
initPos: Vector2, targetPos: Vector2
constructor
super(initPos)
this.setRandomFood
this.targetPos = targetPos


take
check if taking an element is valid
this.amount - 1 < 0
or remove one element and return true
return false
return true
Storage
+ mood: Moods = Moods.Hungry
+ wants: Food
+ speed: number = 0.5
+ status: CustomerStatus = CustomerStatus.ComingIn
+ targetPos: Vector2 | null
+ waitingSince: Date | null
contains: IngredientNames, amount: number, initPos: Vector2
constructor
super(initPos)
this.contains = contains