

Endabgabe

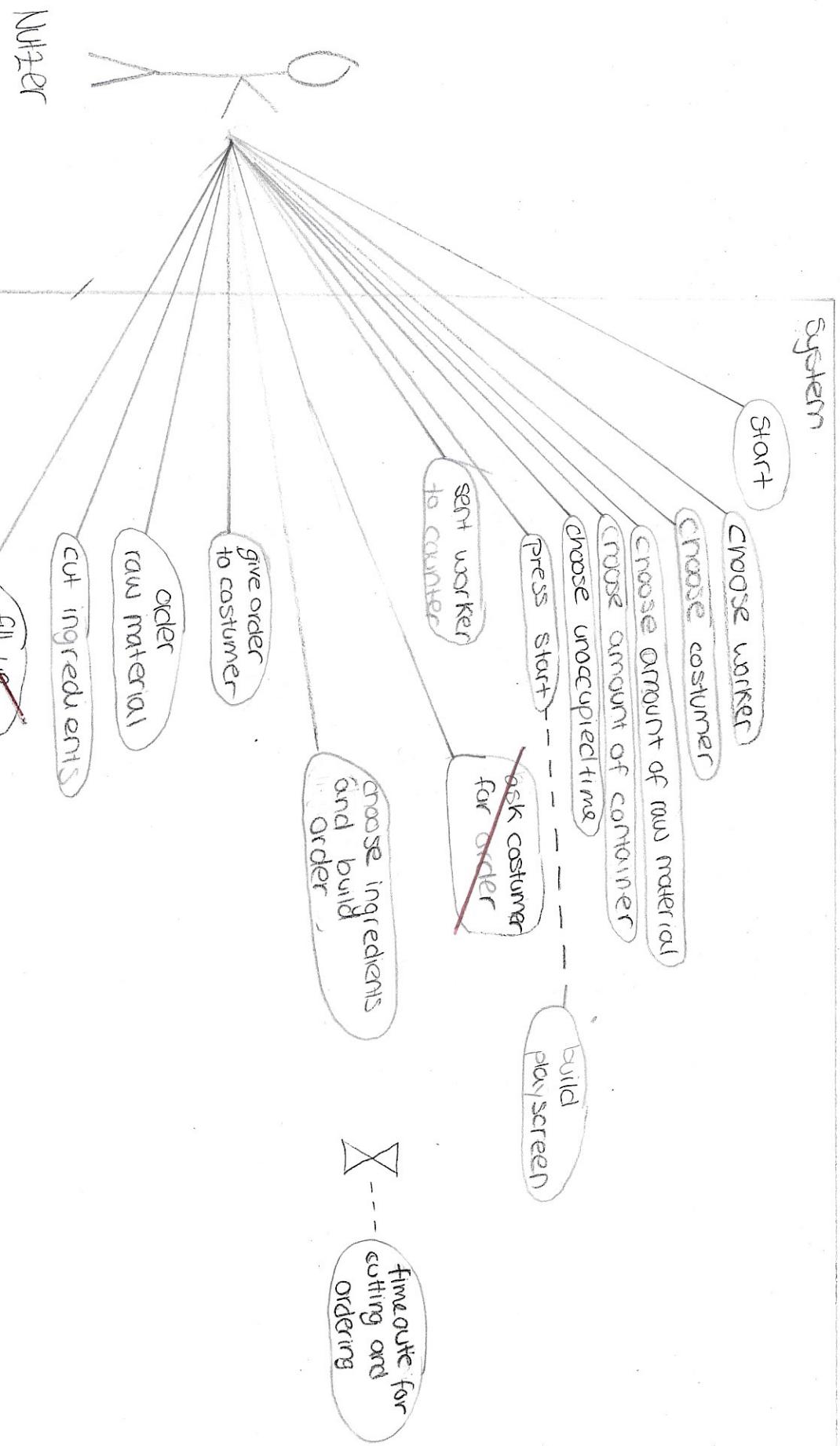
Gruppenarbeit von Asya Tetik, Christina Gabler, Lisa Fingerle, Deborah Reinbold

Aus Gründen der Übersichtlichkeit haben wir gemeinsam nur in einem Repository gearbeitet und dort rein gepusht und gepullt und die fertige Endabgabe dann in einzelne Repositories hochgeladen.

Döner-Trainer Anleitung

1. Zu Beginn kann der Nutzer auf der Startseite die Anzahl der Kunden pro Minute, die Anzahl der Arbeiter, die Kapazität vom Rohmaterial und geschnittenem Material und der Leerlaufzeit der Mitarbeiter bestimmen.
2. Der Kunde gibt seine Bestellung auf und wird vom Mitarbeiter (durch den Nutzer gesteuert) bedient. Um die Mitarbeiter zu bewegen, muss er zuvor durch einen Klick ausgewählt werden.
3. Durch Klicken auf die einzelnen Zutaten wird die Bestellung zusammengestellt und die zur Verfügung stehenden Zutaten werden verringert. Danach kann das Gericht mit Klick auf den Kunden diesem übergeben werden.
4. Die Zufriedenheit der Kunden kann sich durch die Geschwindigkeit und die Genauigkeit des Spielers verändern und das Spiel beeinflussen. Bspw.: Wenn der Spieler zu lange braucht, um ein Gericht zu servieren, wird der Kunde wütend und verlässt die Dönerbude.
5. Auch die Mitarbeiter verändern ihre Stimmung je nach dem, wie hoch ihr Stresslevel ist.
6. Falls die Zutaten an der Theke leer sind, muss der Mitarbeiter in die Küche geschickt werden, um die Rohstoffe zu schneiden. Sind die Rohstoffe leer, können sie nachbestellt werden.
7. Die Zufriedenheit der Kunden und Mitarbeiter sowie die Anzahl der verkauften Gerichte werden fortlaufend angezeigt.

Döner-Trainer : Use-Case-Diagram



UI-Scribble: Startseite

Donertrainer

1. `<h1>`

Mitarbeiter:

A ▲ ▼

Kunden pro Minute

1 | ↕

Rohmaterial

1 Stock

0 Stock

Container

1 Stück → 0 → 15 Stück

Leeraufzeit

20 Sekunden ↕

START

<select>

id: "worker"
name: "Worker"

<div>
id="form"

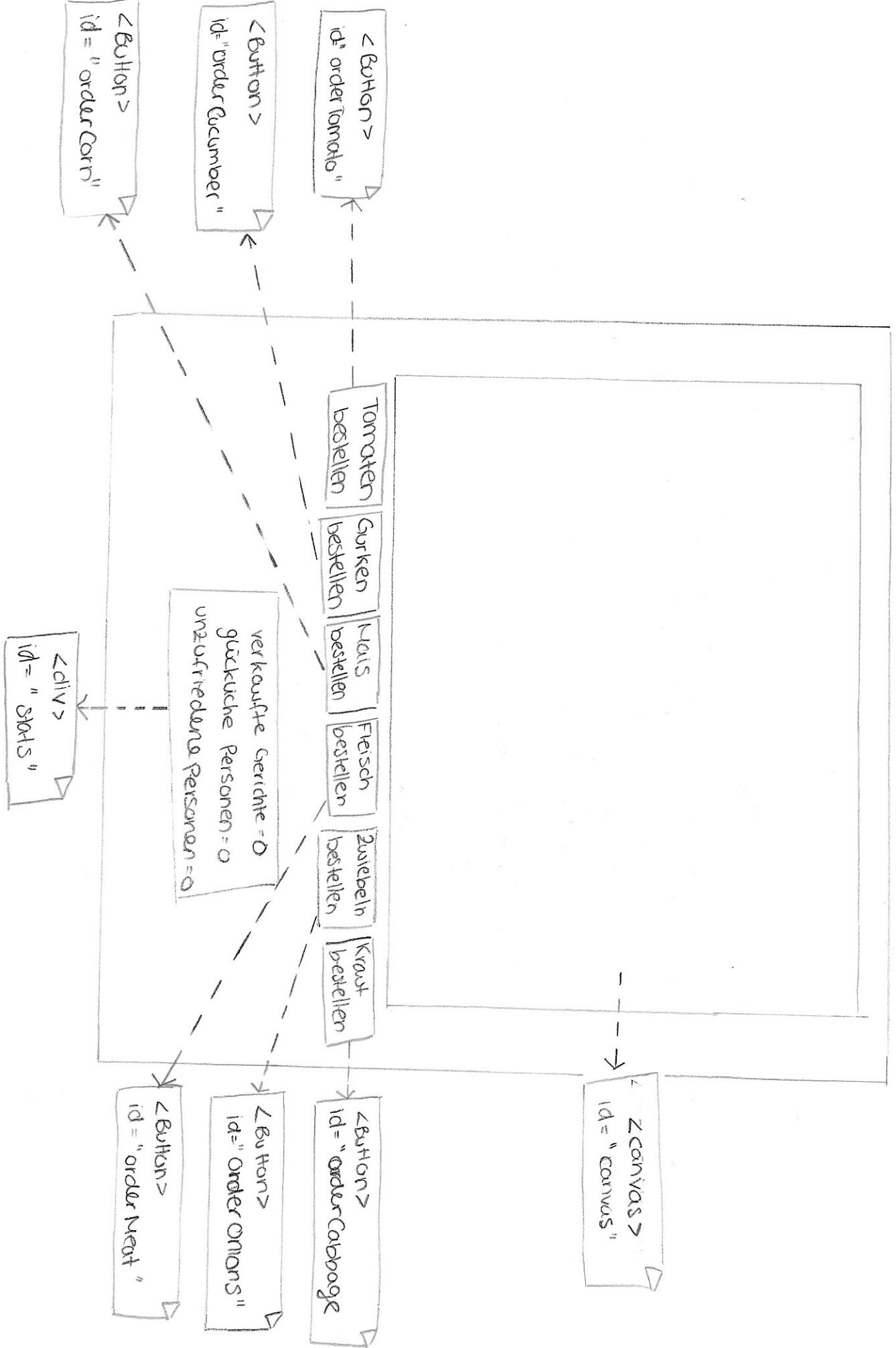
<input>
type="range"
name="Container"
min="1"
max="15"
step="1"

<input>
type="range"
name="Container"
min="1"
max="15"
step="1"

<select>
id: "unoccupied"
name: "unoccupied"

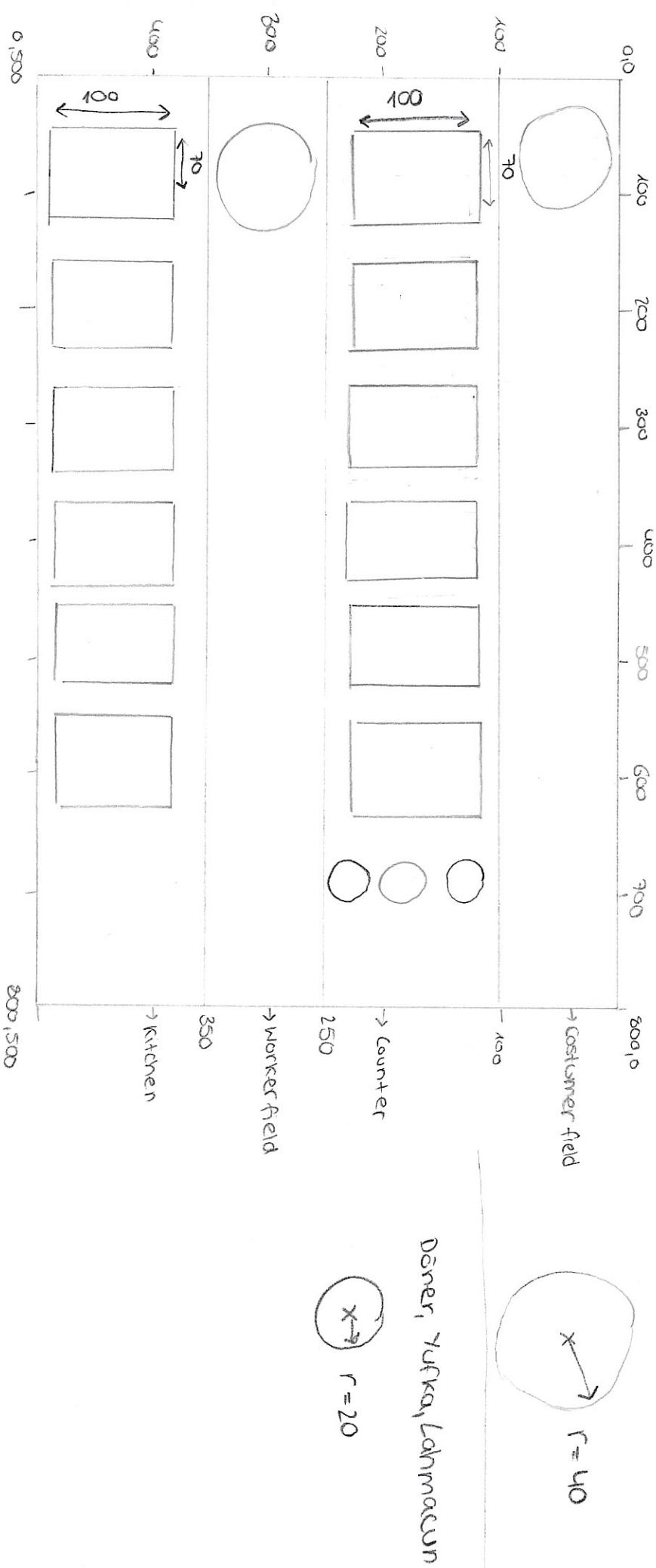
<button>
id="start"

UI - Skizze : Dinerbude

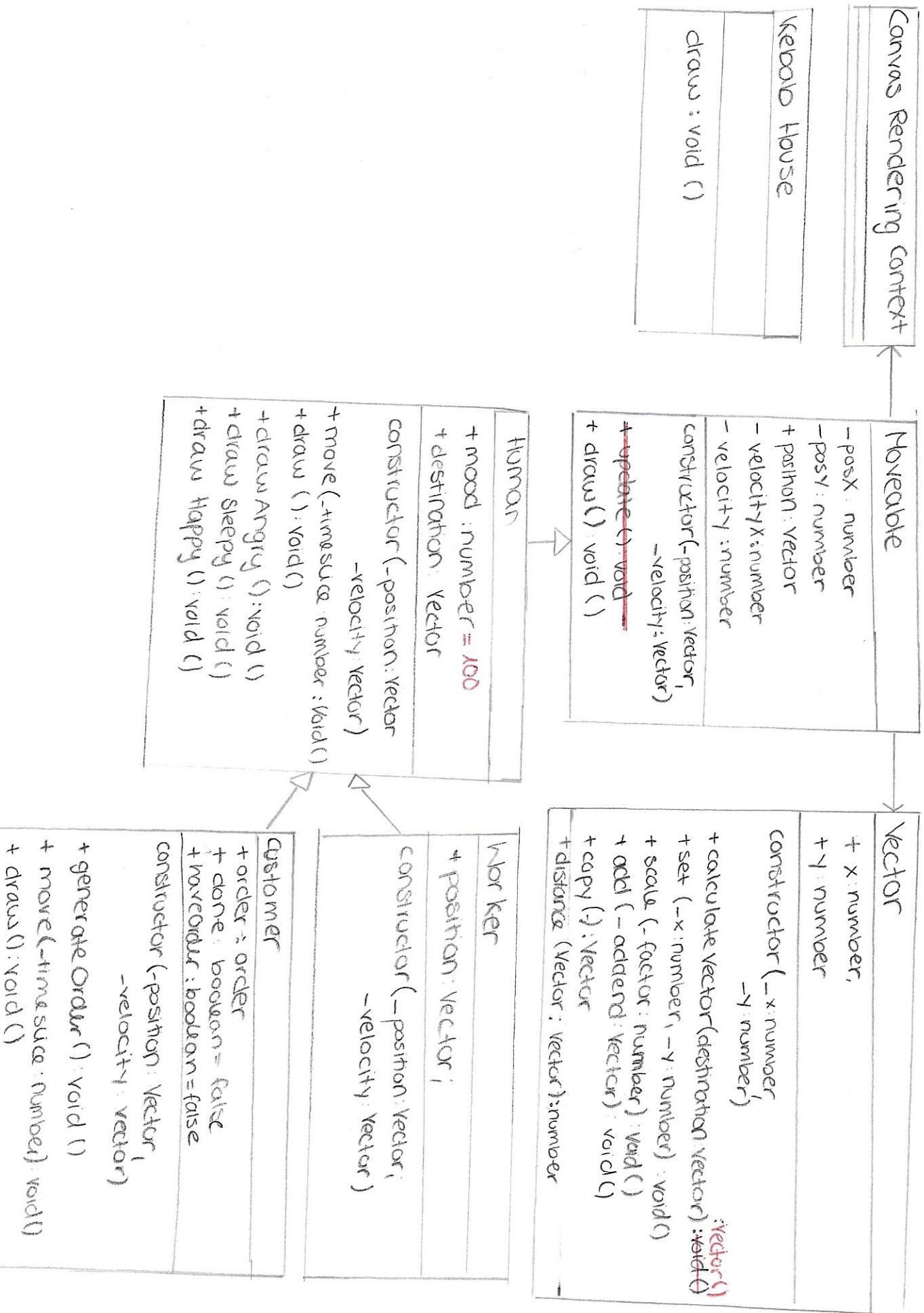


Canvas: Keyboard - House

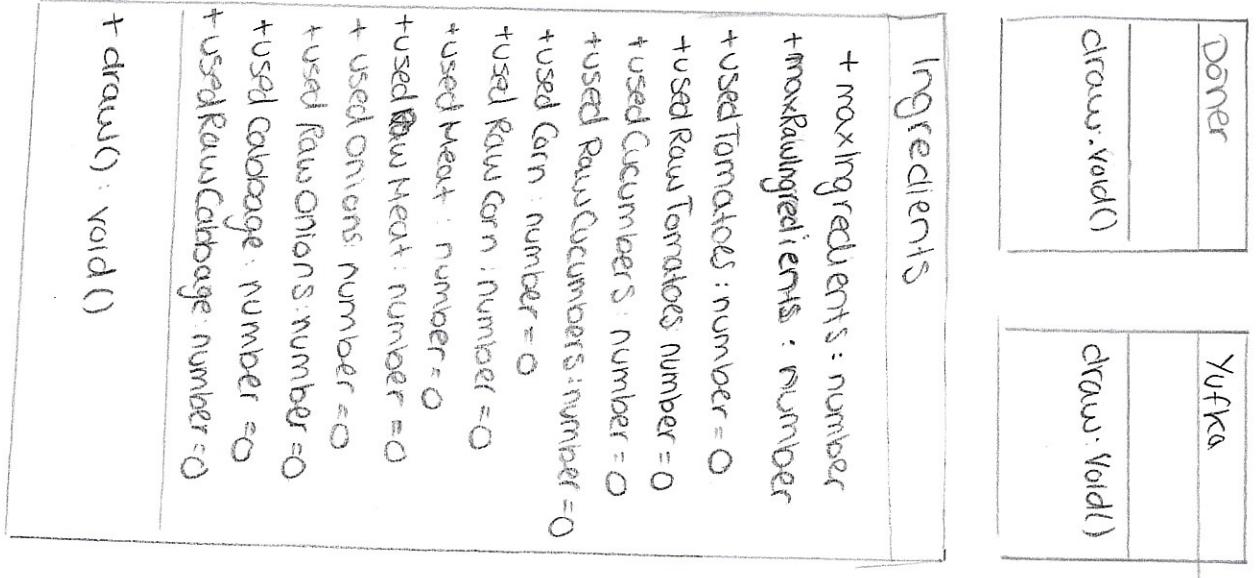
worker & customer



Classdiagram @



Classdiagram ②



Lahmacun
draw(): void()

<< interface >>
order
name: string ingredients: IngredientsList[]

<< enum >>
IngredientsList

DÖNER,
 YUFKA,
 LAHMACUN,
 TOMATOES,
 CUCUMBER,
 CORN,
 MEAT,
 ONION,
 CABBAGE

Activity Diagram : Nachi

```

let crc2: CanvasRenderingContext2D
let startButton: HTMLButtonElement
let kebabhouse: KebabHouse
let ingredients: Ingredients
let doener: Doener
let yufka: Yufka
let Lahmacun: Lahmacun
let workerNumber = 2
let customerNumber = 3
let rawNumber = 10
let containerNumber = 15
let unoccupiedNumber = 20
let movable: Movable[] = []
let orientation: Position = false
let allCustomerNumber = 0
let imgData = imageData
let usedIngredients: IngredientsList[] = []
let activeWorker: Worker
let doneOrderNumber = 0
let happyCustomerNumber = 0
let angryCustomerNumber = 0
let happyWorkerNumber = 0
let unhappyWorkerNumber = 0
let statsDiv: HTMLDivElement

```



Load

Click Start Button

MouseOver

handle Load

handle Click

handle MouseOver



Activity Diagram : Main ②

startGame

delete formelements

kebabHouse = new KebabHouse()

kebabHouse.draw()

imgData = arr[2].getImageData

ingredients = newIngredients()

ingredients.draw()

same for Lahmacun, dosener, yufka

get settings in

create worker in

create customer in

set container & raw
to chosen max

set interval for
createCustomer & update

every 20sek

```

let order TomatoButton : HTMLButtonElement
let order CucumberButton : HTMLButtonElement
let order CornButton : HTMLButtonElement
let order MeatButton : HTMLButtonElement
let order OnionButton : HTMLButtonElement
let order CabbageButton : HTMLButtonElement
    
```

Activity Diagram : Main ③

get settings ↓

[let formData : FormData = new FormData]

worker = Number(formData.get("worker"))

customer = Number(formData.get("customer"))

raw = Number(formData.get("Raw"))

container = Number(formData.get("Container"))

unoccupied = Number(formData.get("Unoccupied"))



create worker

[nWorker = 0]

② ← [nWorker < worker]

draw new worker

push to Moreable[]

increase
nWorker
++

create customer

③ ← [allCustomer < 5]

draw customer

push to Moreable[]



Activity Diagram : Main

count worker mood

set happy worker &
unhappy worker to 0

let moveable of moveables

[moveable is from worker]

happy worker
+1

[sleepy || angry]
unhappy worker
+1

update

let moveable of moveables

draw moveable

○

[moveable is from worker]
↓
[moveable is from customer]

mood decreases

generate order

+1

more customer

+1

mood decreases

move worker

+1

customer gone
[customer angry &
no order]

move to door

angry customer
+1

decrease
customer

+1

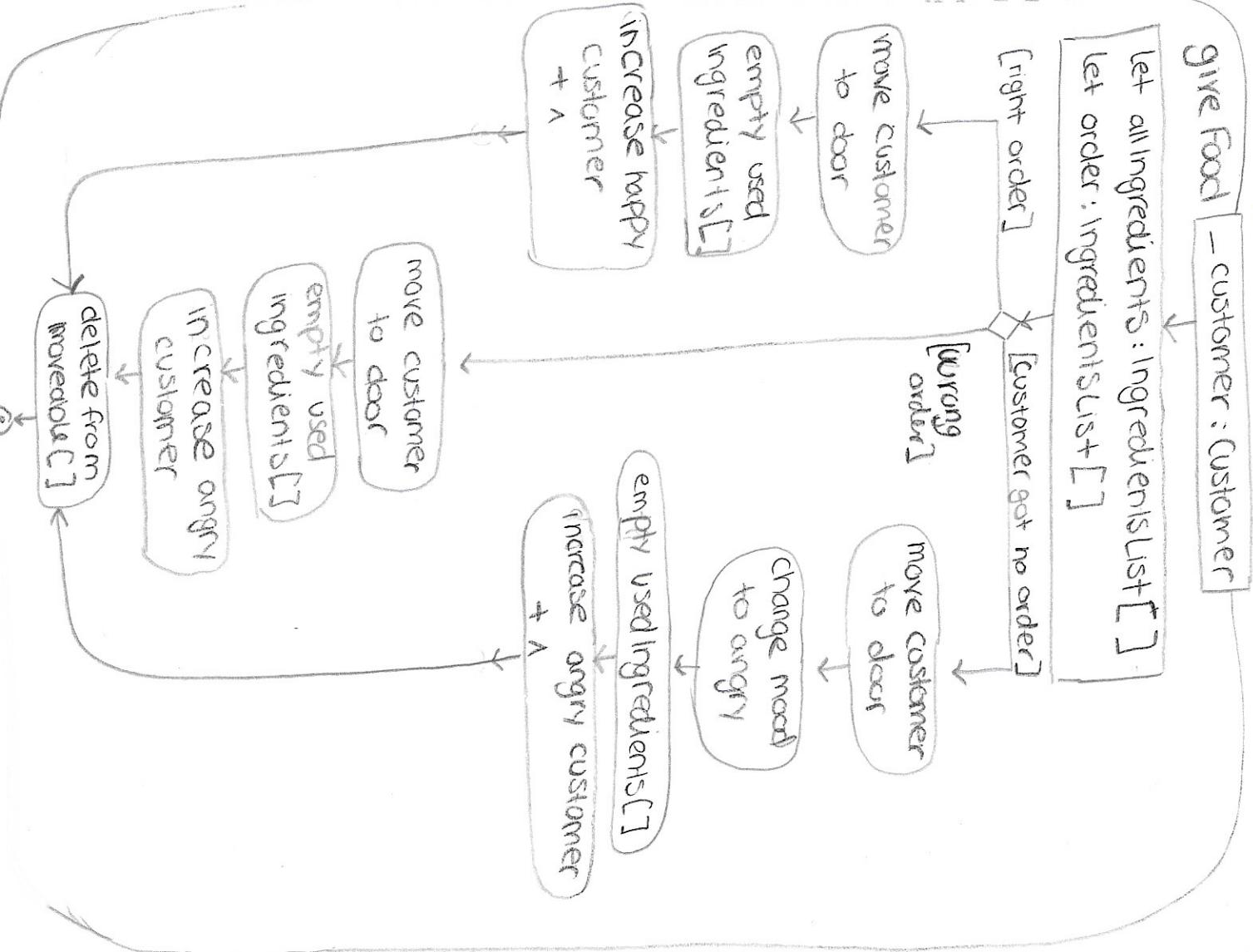
count worker mood

+1

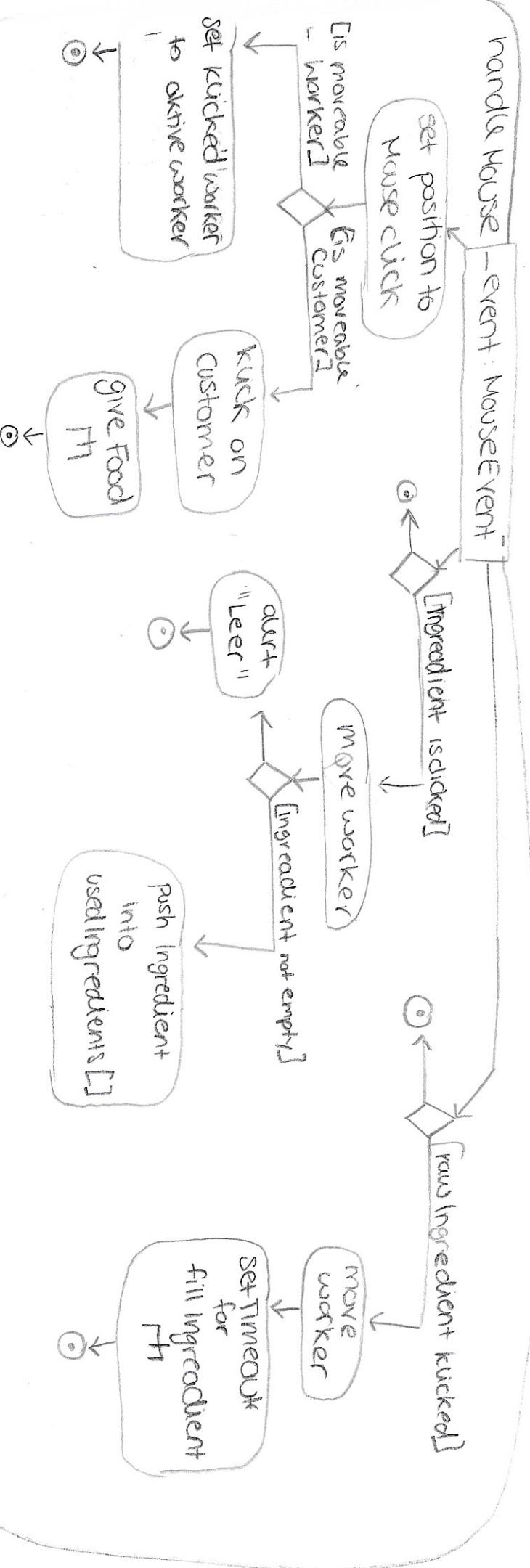
fill statsDir

draw ingredients, dinner,
yufka, tomato sauce

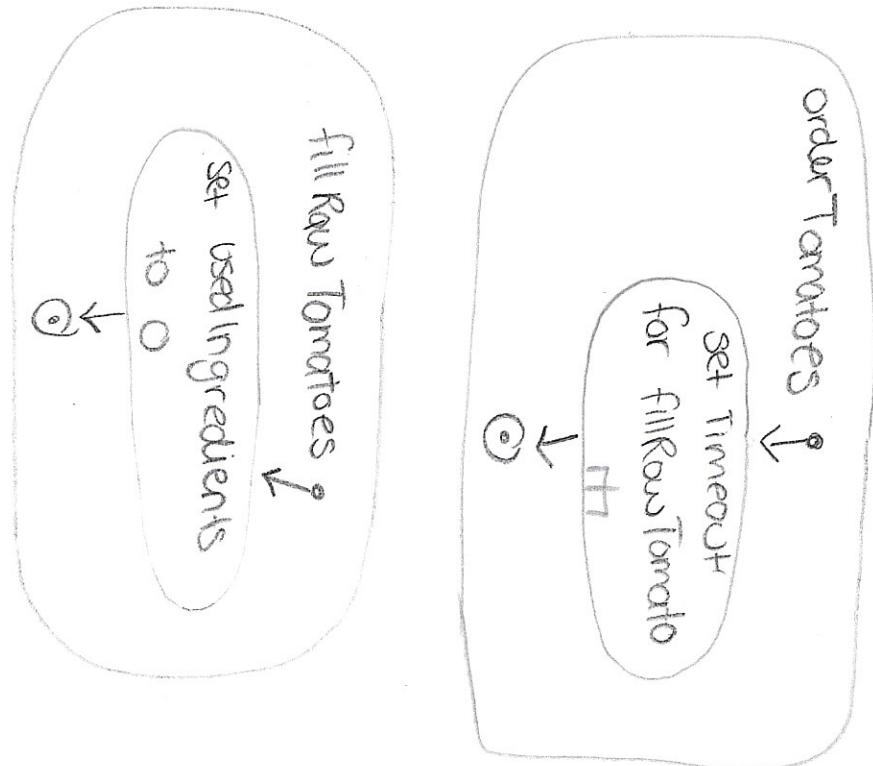
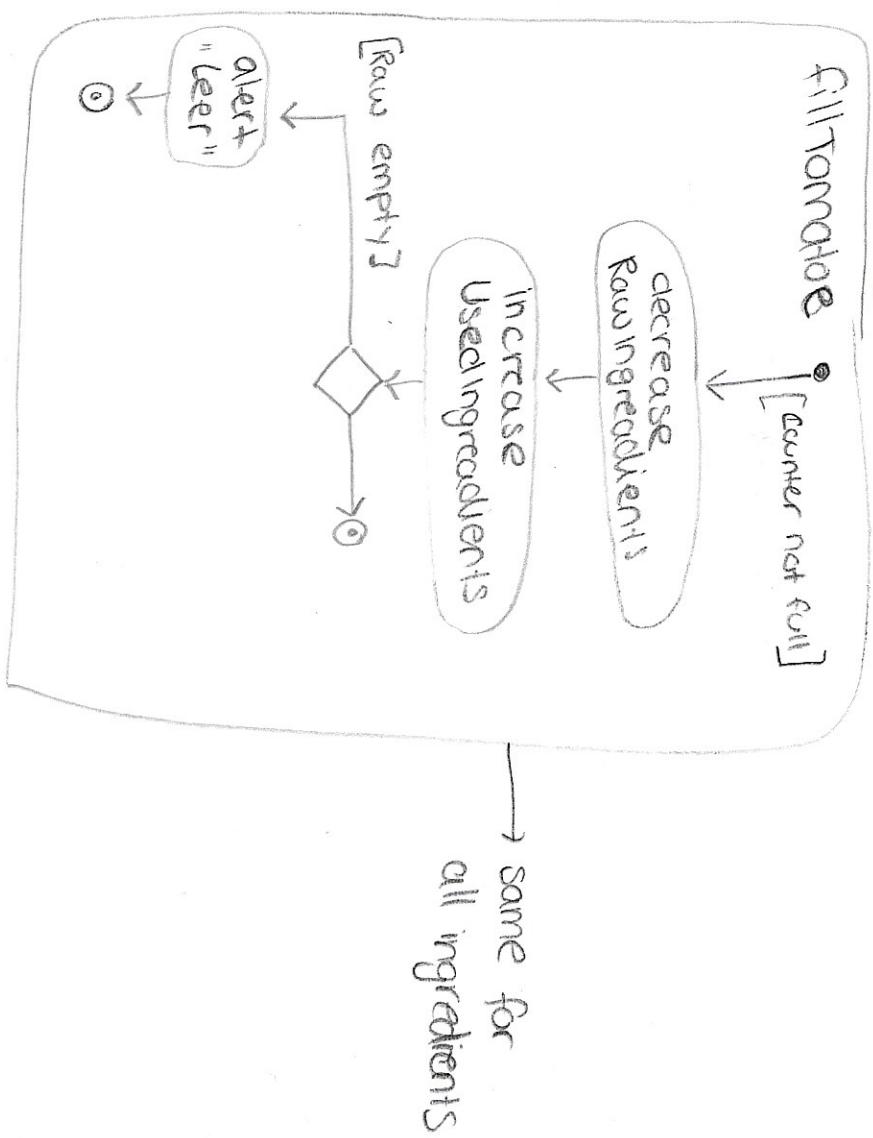
Activity Diagram: Main ⑤



Activity Diagram: Main



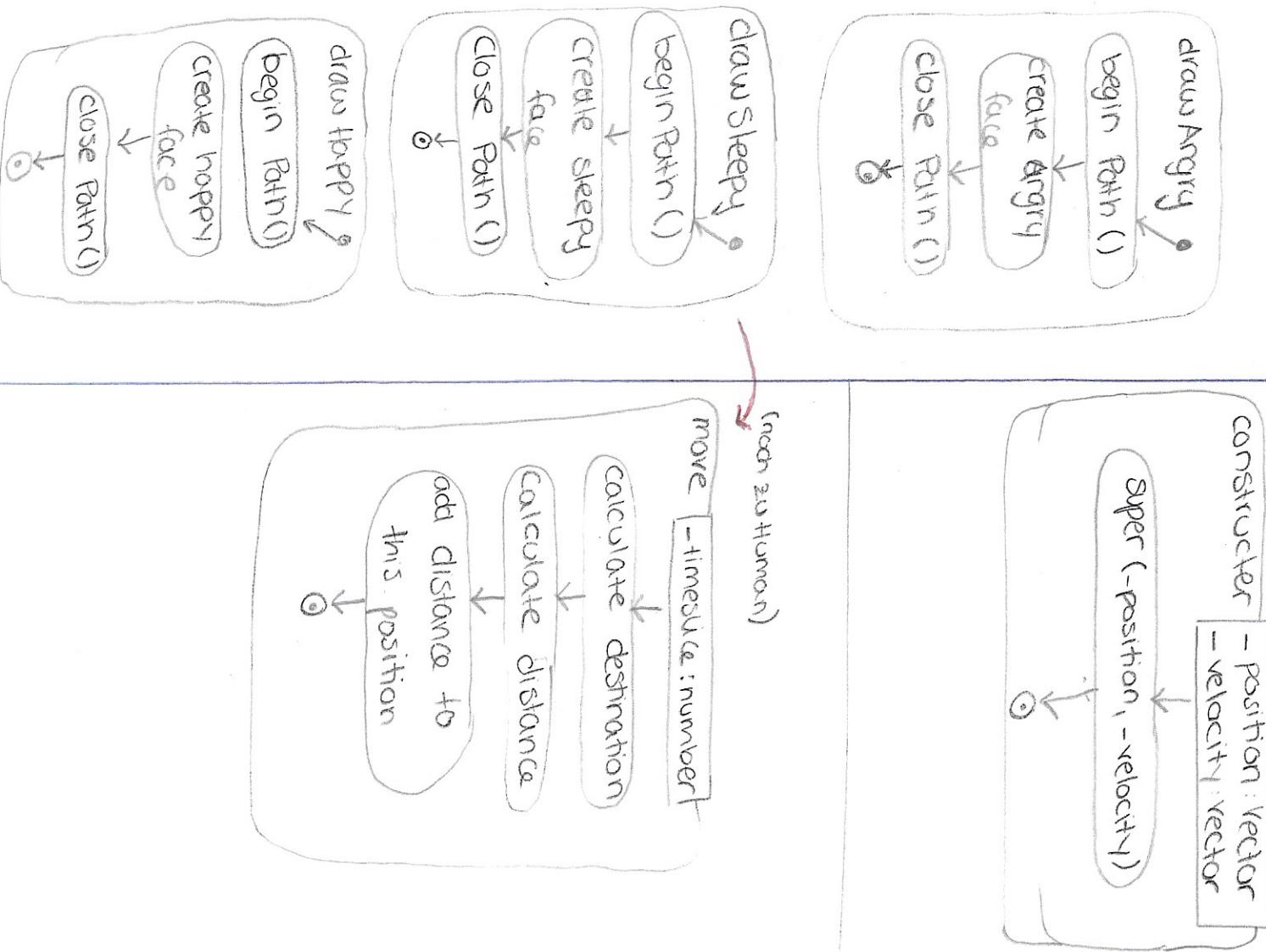
Activity Diagram : Main



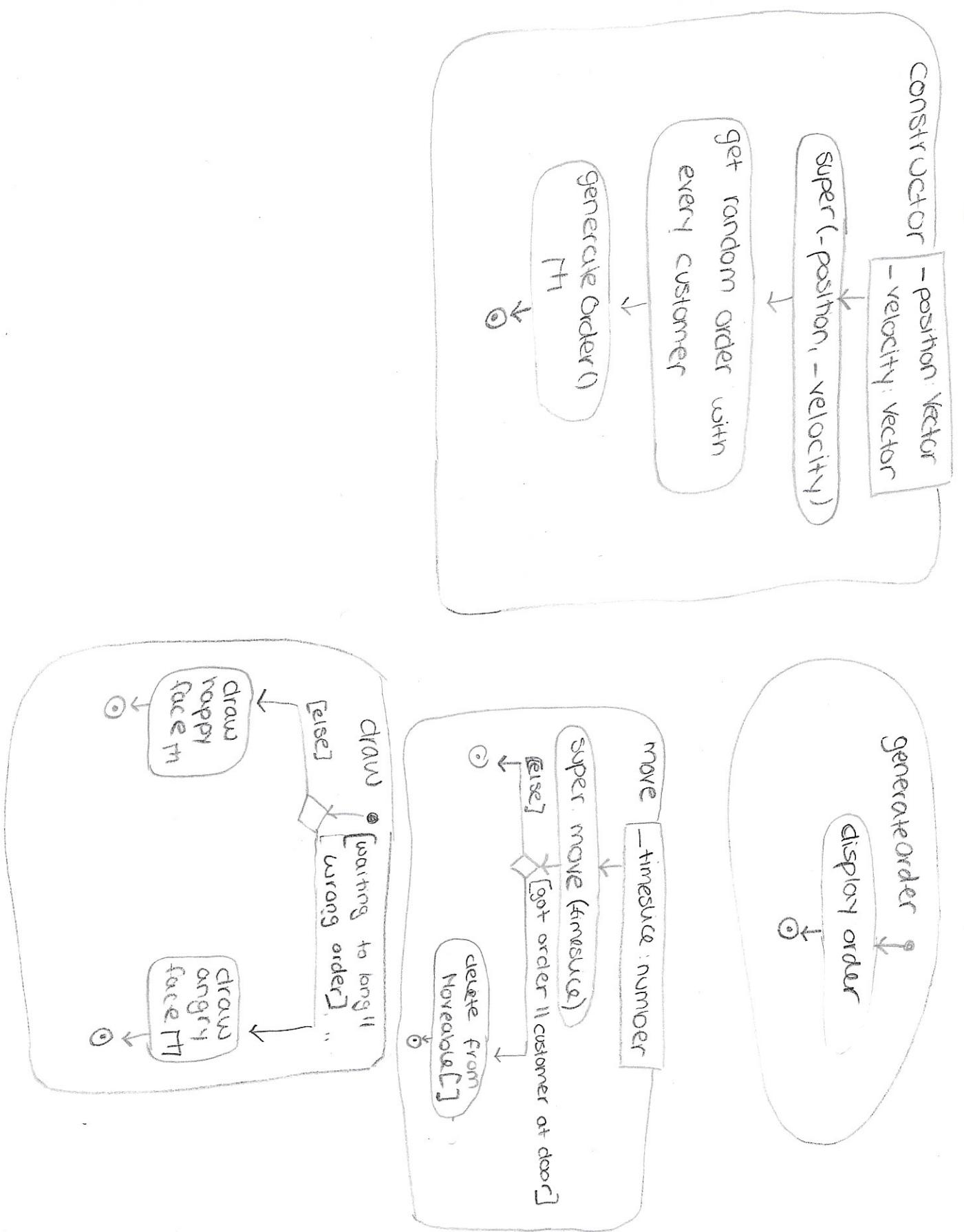
Activity Diagram: Human



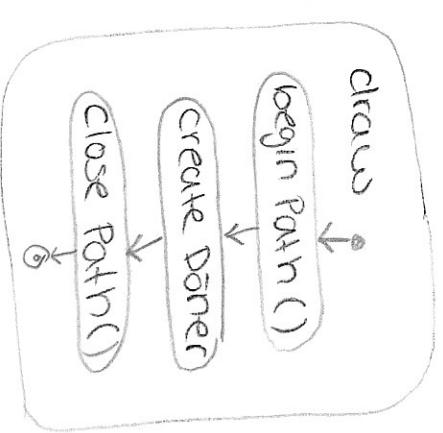
Activity Diagram: Worker



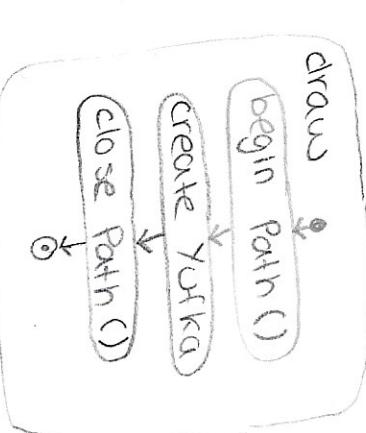
Activity Diagram: Customer



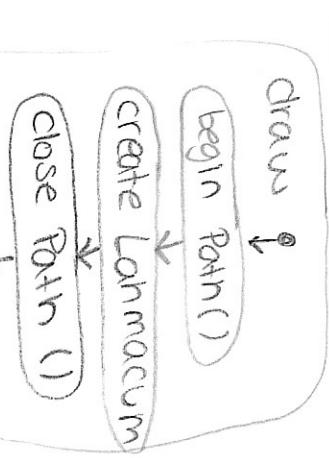
Activity Diagram : Döner



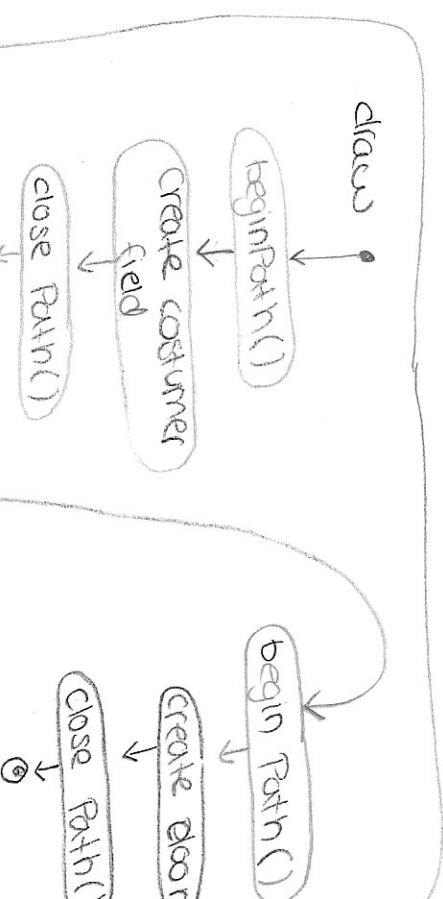
Activity Diagram : Yufka



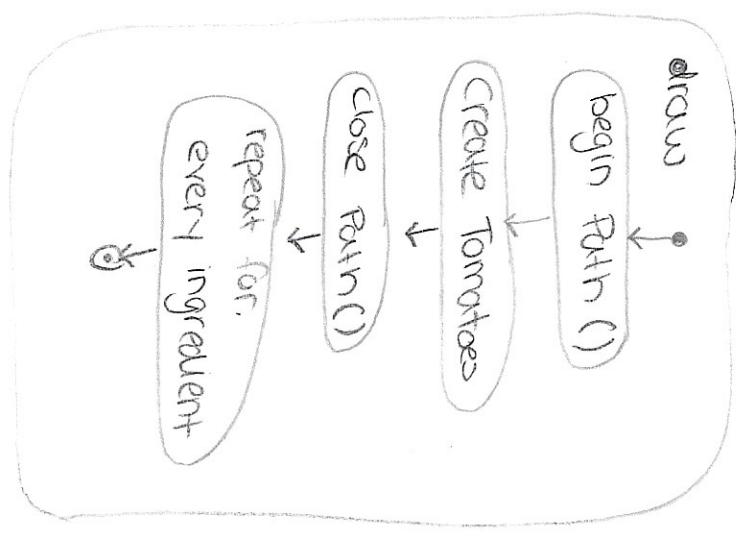
Activity Diagram : Lahmacum



Activity diagram : kebab - house



Activity Diagram: Ingredients



Activity Diagram: Moveable

