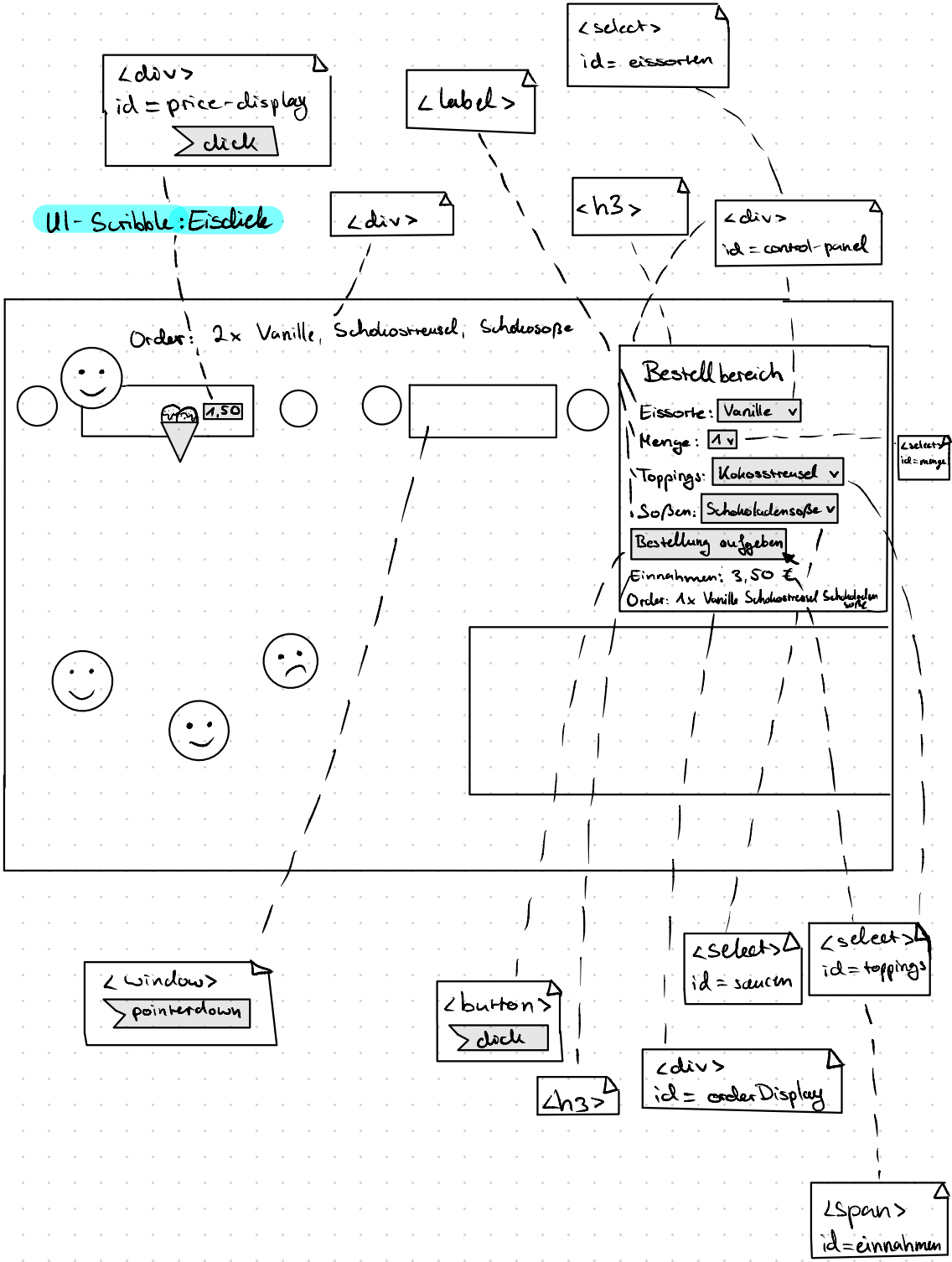

<select>
id= eissorten

id = price-display
click
<label>
UI- Scribble: Eisdiele

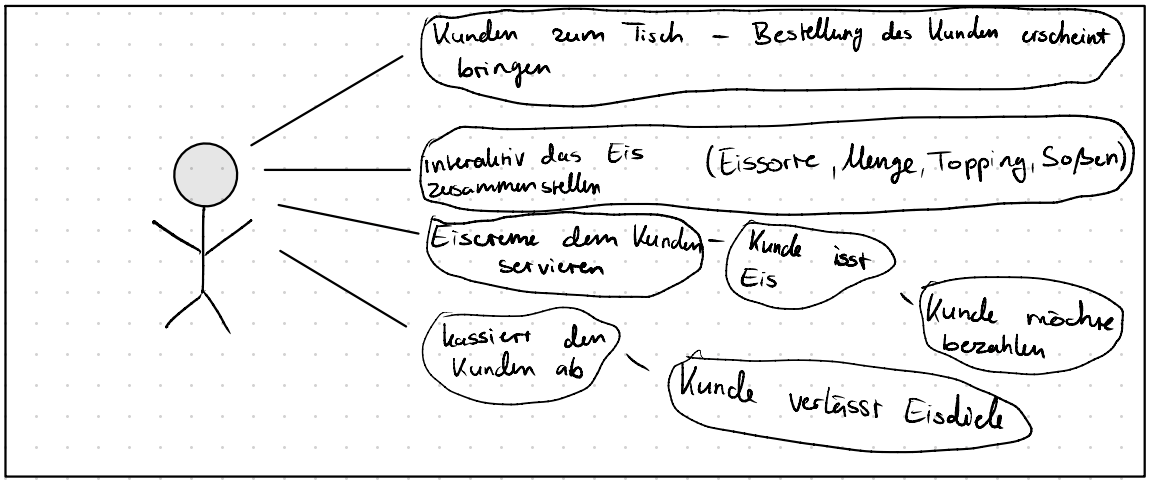
<h3>

id = control-panel
Order: 2x Vanille, Schokostreusel, Schokosoße
1,50
Bestellbereich
Eissorte: Vanille
Menge: 1
Toppings: Kokosstreusel
Soßen: Schokoladensoße
Bestellung aufgeben
Einnahmen: 3,50 €
Order: 1x Vanille Schokostreusel Schokoladensoße
<select>
id = menge
<window>
pointerdown
<button>
click
<h3>
<select>
id = sauces
<select>
id = toppings

id = orderDisplay

id=einnahmen

Use-Case-Diagram



Datenstruktur: FoodData

Category	name: string	price: number	color: string
Eissorte	Vanille Schokolade Erdbeere	1,0 1,0 1,0	'yellow' 'darkbrown' 'red'
Sauce	Schokoladensoße Erbeersoße	0,3 0,3	'brown' 'red'
Toppings	Schokoladenstreusel Kokosstreusel	0,2 0,2	'black' 'white'

Klassendiagramm: Table

Table

+position: (x,y) number
+color: string
+state: string

constructor (_x: number, _y: number, _color: string)
+ draw (): void

Klassendiagramm: Stool

Stool

+ position: (x,y) number
+ color: string

constructor (_x: number, _y: number, _color: string)
+ draw (): void

Klassendiagramm: Customer

Customer

+ position: (x, y) number
+ color: string
+ mood: string
+ state: string
+ targetPositionX: number
+ targetPositionY: number
- lastStateChangeTime: number
- moodchangeDelay: number
+ order: string
+ table: string
+ actualOrder: string

constructor (_x: number, _y: number, _color: string, _mood: string, _state: string)

+ draw (): void
+ move (): void
+ updatemood (): void
+ setState (): void
- generateRandomOrder (): void

Klassendiagramm: Counter

Counter

+ position: (x,y) number
+ color: string

constructor (_x: number, _y: number, _color: string)
+ draw (): void

CanvasRenderingContext

Klassendiagramm: Food

Food

+ positionX: number
+ positionY: number
+ color: string

constructor (_x: number, _y: number, color: string)

draw (): void

Toppings

+count: number
+streuselPosition: number

constructor (_x: number, _y: number, _count: number, _color: string)
-generateStreuselPosition ()
+draw (): void

Sauce

constructor (_x: number, _y: number, _color: string)

+draw (): void

IceCream

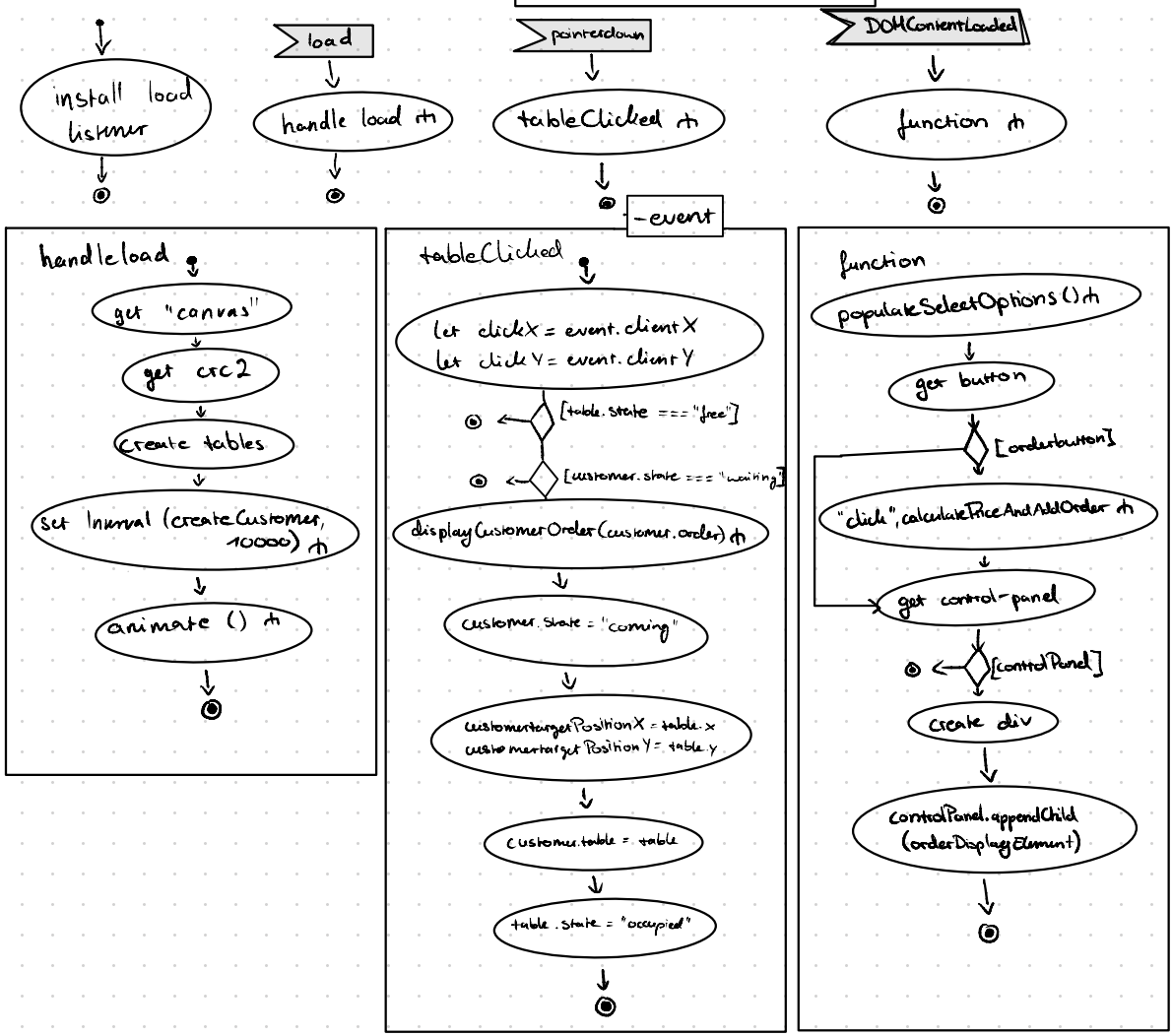
+waffleColor: string
+scoops: number

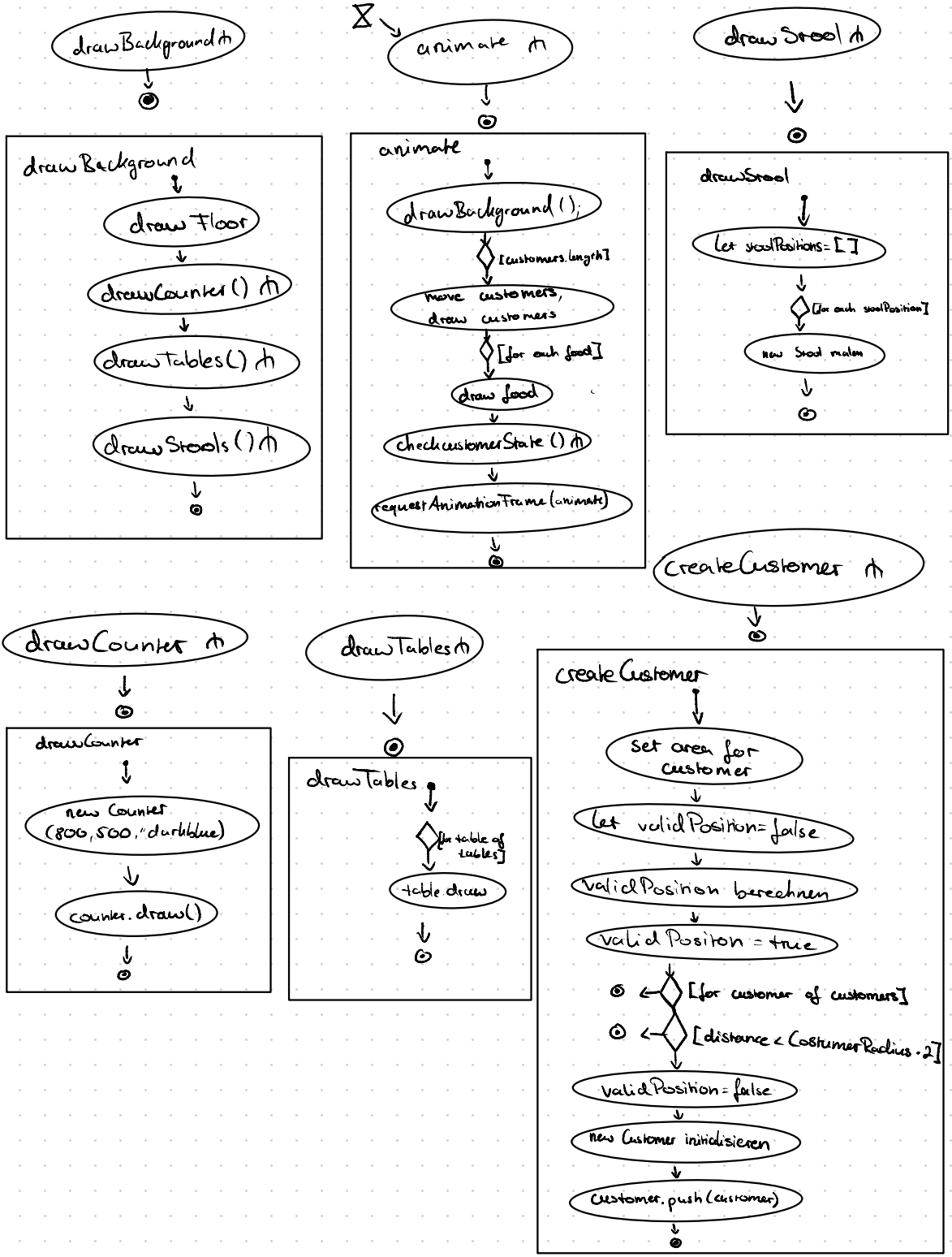
constructor (_x: number, _y: number, _color: string, _waffleColor: string, _scoops: number)

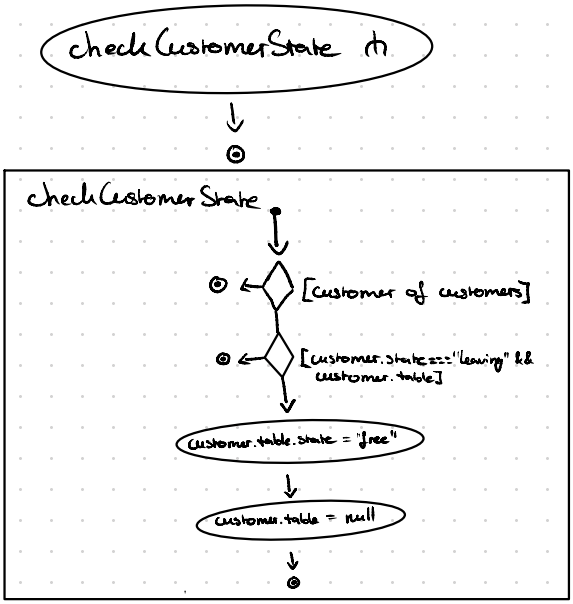
+draw (): void

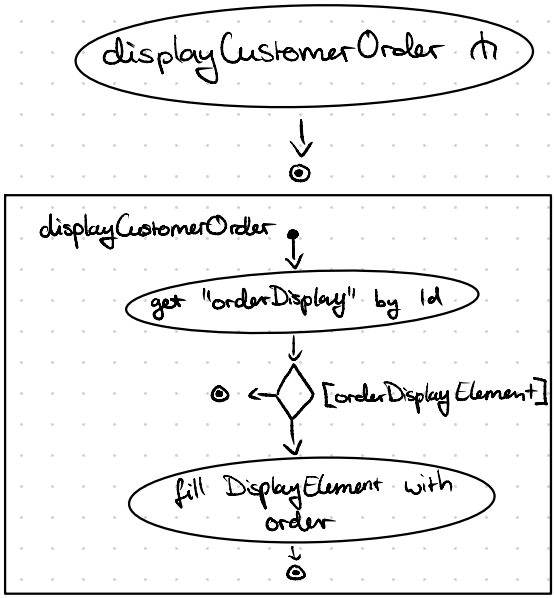
Aktivitätsdiagramm: Eisdiele – main

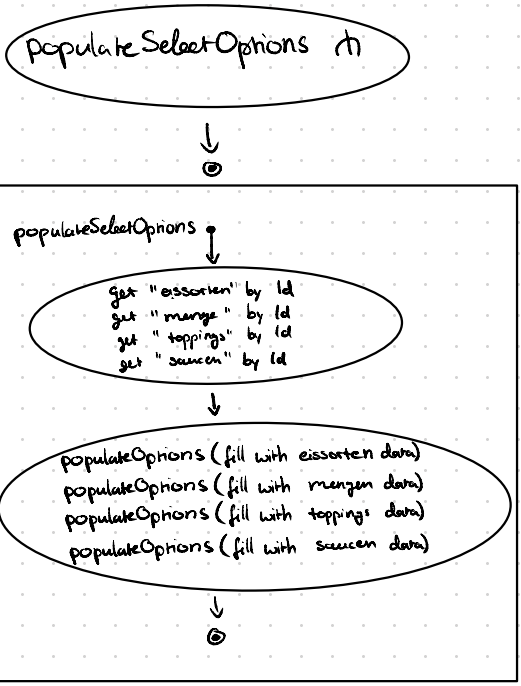
crc2
customers: Customer [] = []
tables: Table [] = []
foods: Food [] = []
customerRadius = 30 * 0.5

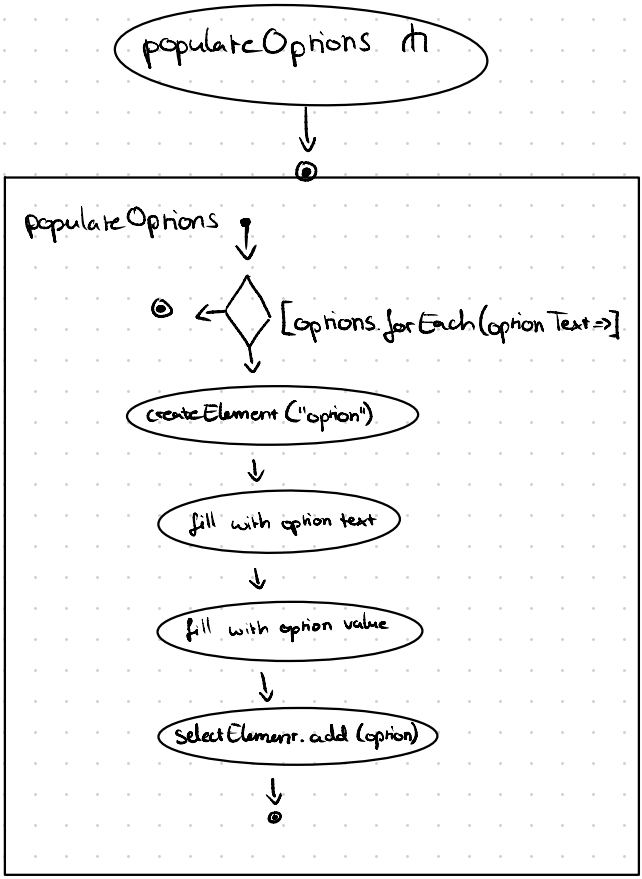


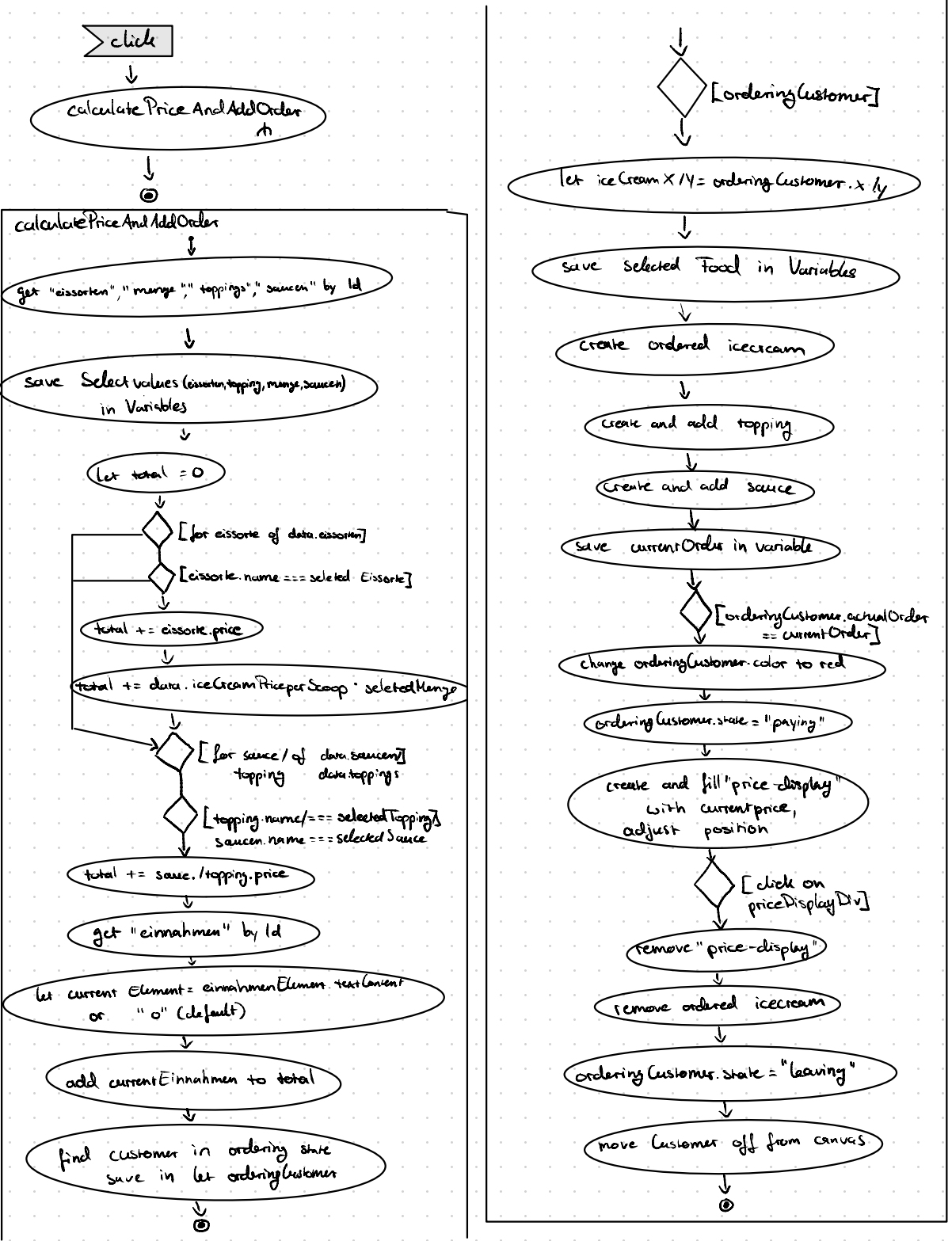

drawBackground
drawBackground
draw Floor
drawCounter()
drawTables()
drawStools()
animate
animate
drawBackground();
[customers.length]
move customers, draw customers
[for each food]
draw food
checkcustomerState()
requestAnimationFrame(animate)
drawStool
drawStool
let stoolPositions = [ ]
[for each stoolPosition]
new Stool maken
drawCounter
drawCounter
new Counter (800, 500, "darkblue)
counter.draw()
drawTables
drawTables
[for table of tables]
table.draw
createCustomer
createCustomer
set area for customer
let validPosition = false
validPosition berechnen
validPositon = true
[for customer of customers]
[distance < CostumerRadius * 2]
validPosition = false
new Customer initialisieren
customer.push(customer)


checkCustomerState
checkCustomerState
[customer of customers]
[customer.state===''leaving'' && customer.table]
customer.table.state = ''free''
customer.table = null

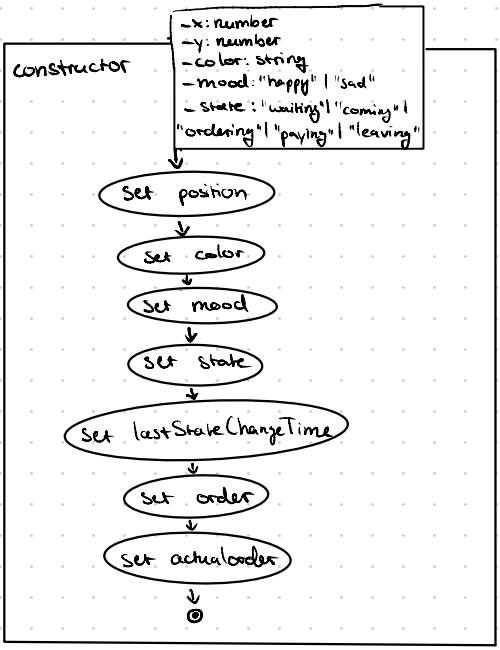

displayCustomerOrder
displayCustomerOrder
get ''orderDisplay'' by Id
[orderDisplayElement]
fill DisplayElement with order

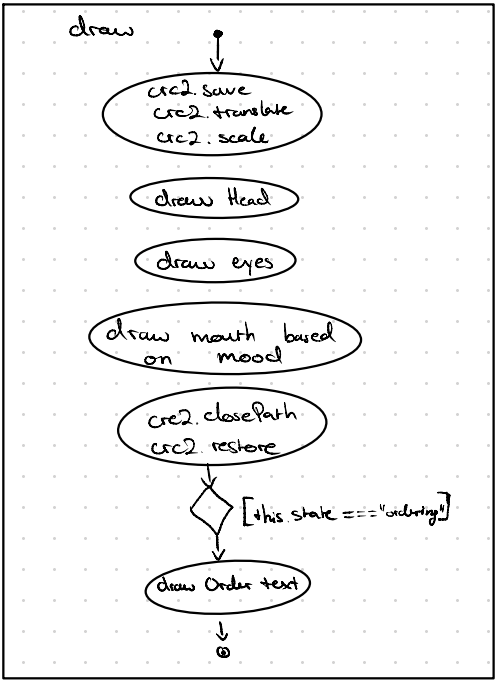

populateSelectOptions
populateSelectOptions
get ''eissorten'' by Id
get ''mengen'' by Id
get ''toppings'' by Id
get ''saucen'' by Id
populateOptions (fill with eissorten data)
populateOptions (fill with mengen data)
populateOptions (fill with toppings data)
populateOptions (fill with saucen data)

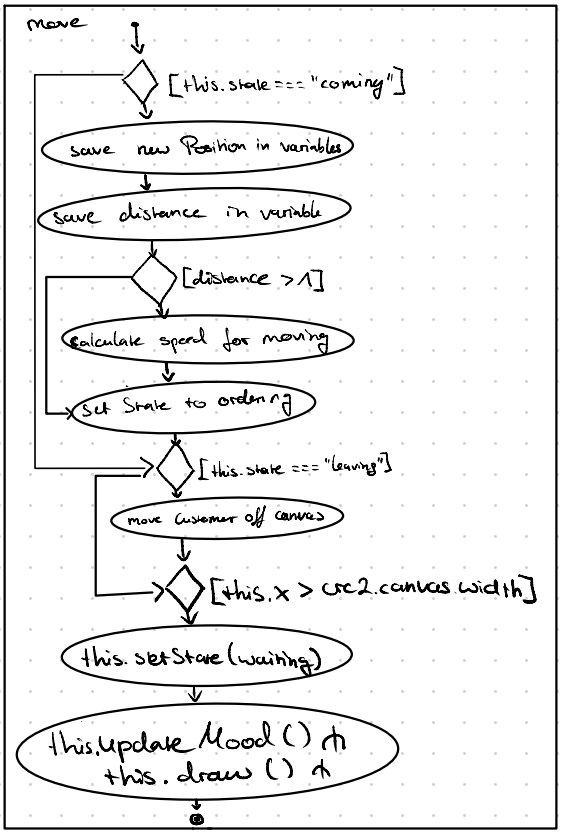

populateOptions
populateOptions
[options.forEach(option Text=>]
createElement ("option")
fill with option text
fill with option value
selectElement.add (option)

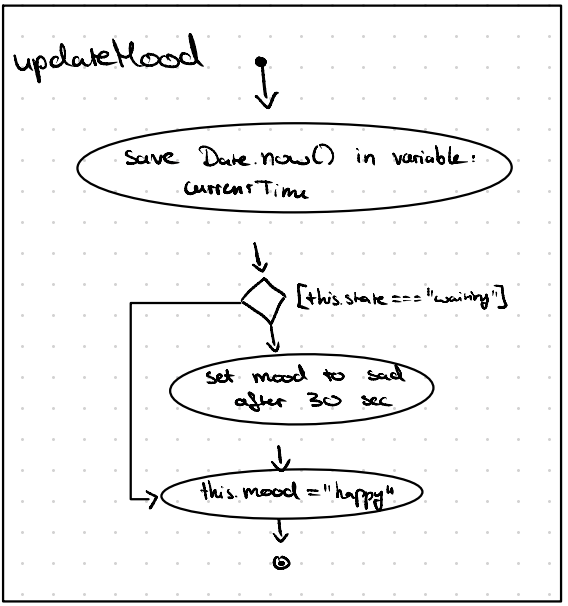

click
calculatePriceAndAddOrder
calculatePriceAndAddOrder
get "eissorten", "menge", "toppings", "saucen" by Id
save Select values (eissorten, topping, menge, saucen) in Variables
let total = 0
[for eissorte of data.eissorten]
[eissorte.name === seleted Eissorte]
total += eissorte.price
total += data.iceCreamPricepersCoop * seletedMenge
[for sauce / topping of data.saucen / data.toppings]
[topping.name / sauce.name === selected Topping / selected Sauce]
total += sauce./topping.price
get "einnahmen" by Id
let currentElement = einnahmenElement.textContent or "0" (default)
add currentEinnahmen to total
find customer in ordering state save in let orderingCustomer
[orderingCustomer]
let iceCream X / Y = orderingCustomer.x / y
save selected Food in Variables
create ordered icecream
create and add topping
create and add sauce
save currentOrder in variable
[orderingCustomer.actualOrder == currentOrder]
change orderingCustomer.color to red
orderingCustomer.state = "paying"
create and fill "price-display" with currentprice, adjust position
[click on priceDisplayDiv]
remove "price-display"
remove ordered icecream
orderingCustomer.state = "leaving"
move Customer off from canvas

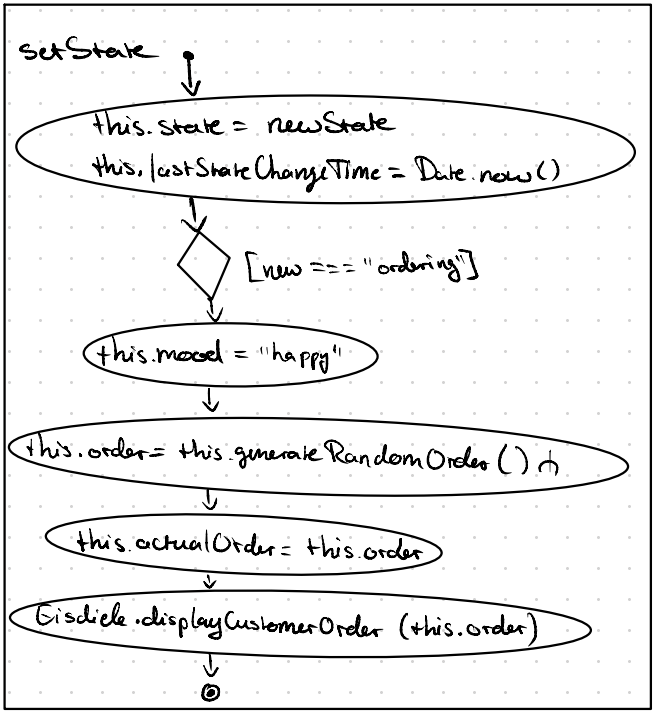
Aktivitätsdiagramm: Customer

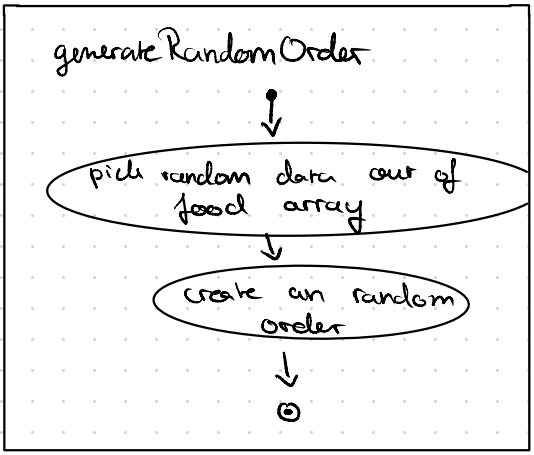




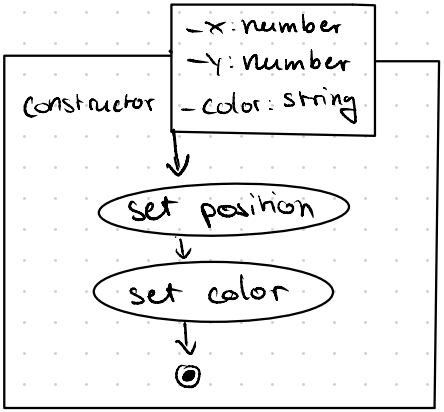


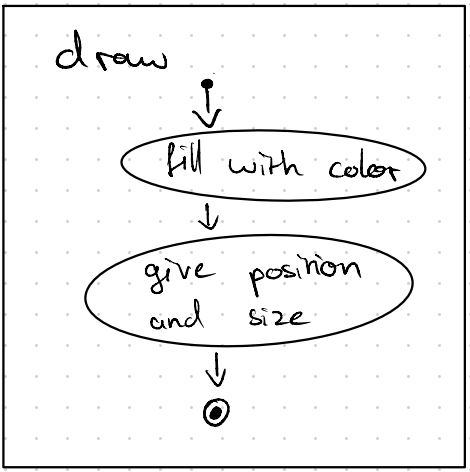




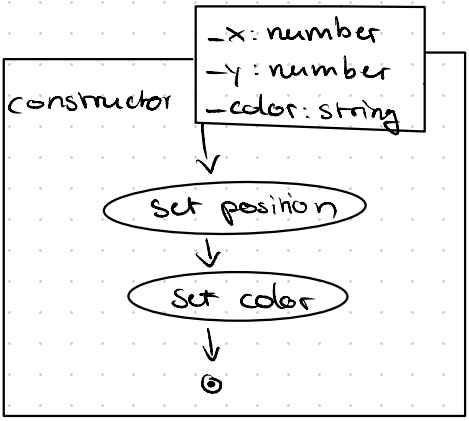


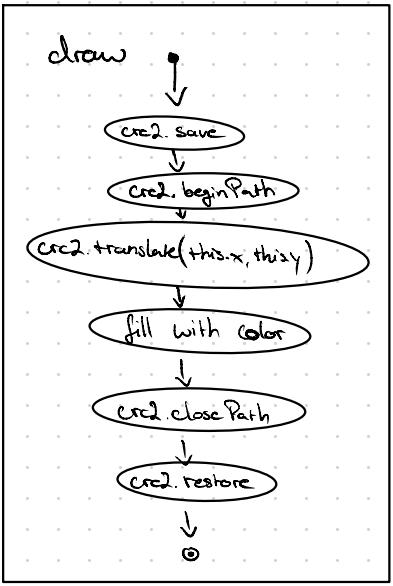
Aktivitätsdiagramm: Counter



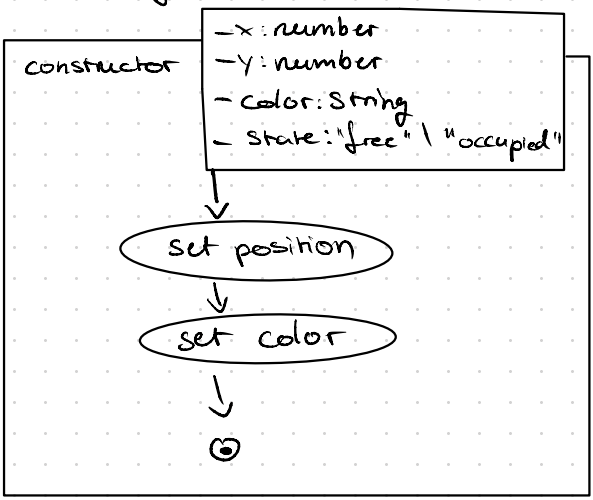


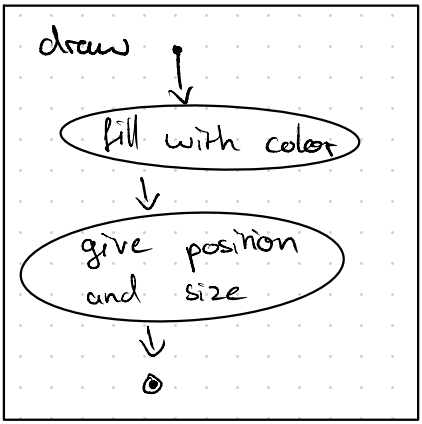
Aktivitätsdiagramm: Stool



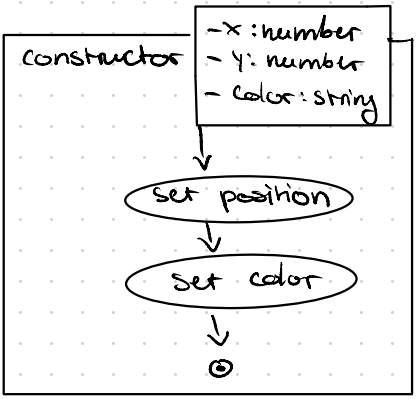


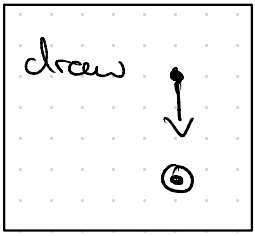
Aktivitätsdiagramm: Table



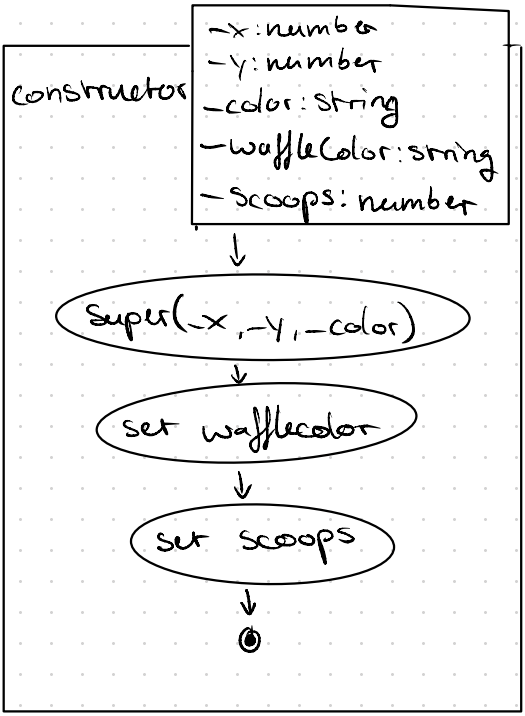


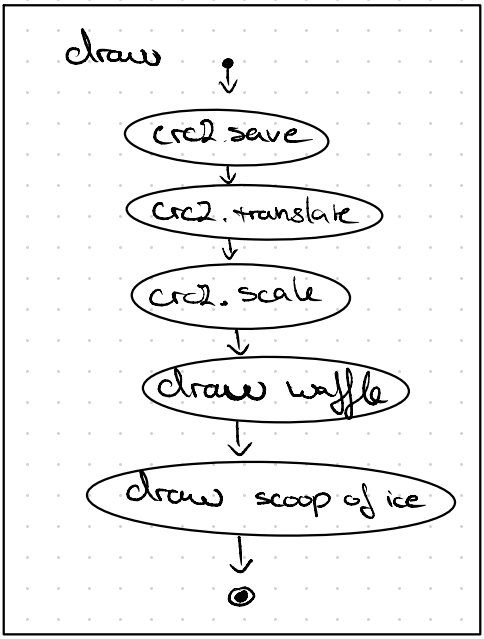
Aktivitätsdiagramm : Food



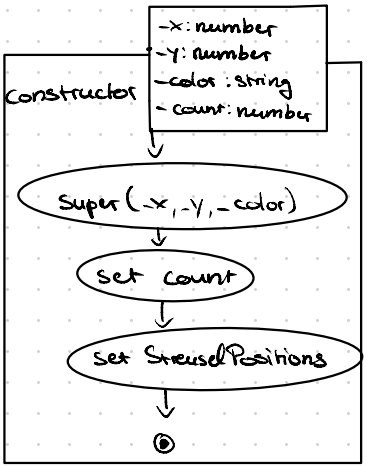


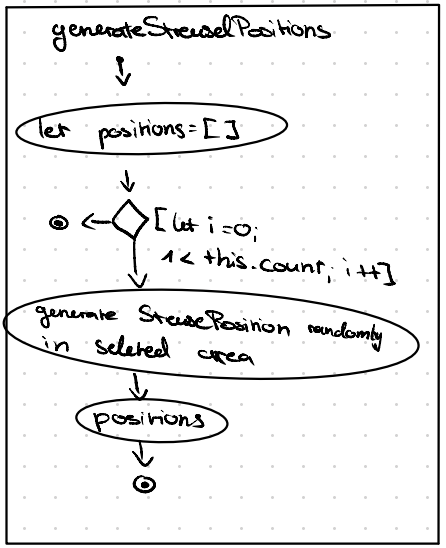
Aktivitätsdiagramm : Icecream

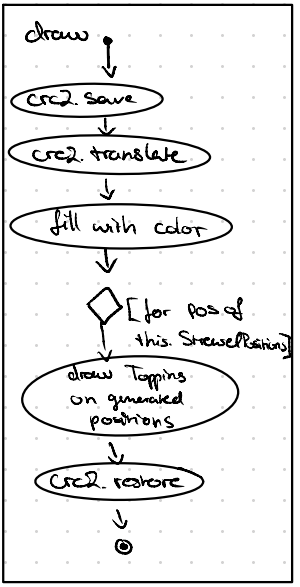
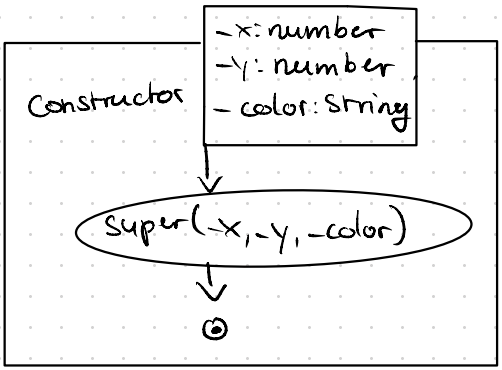
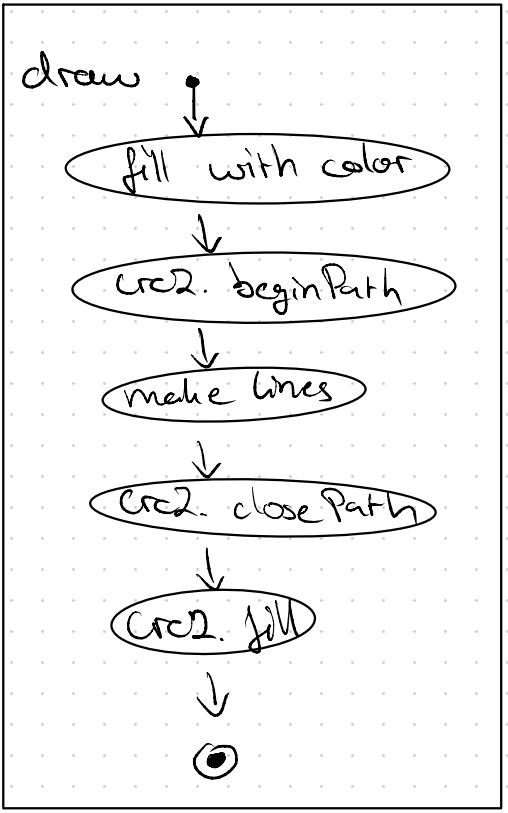




Aktivitätsdiagramm: Topping







Aktivitätsdiagramm: Sauce