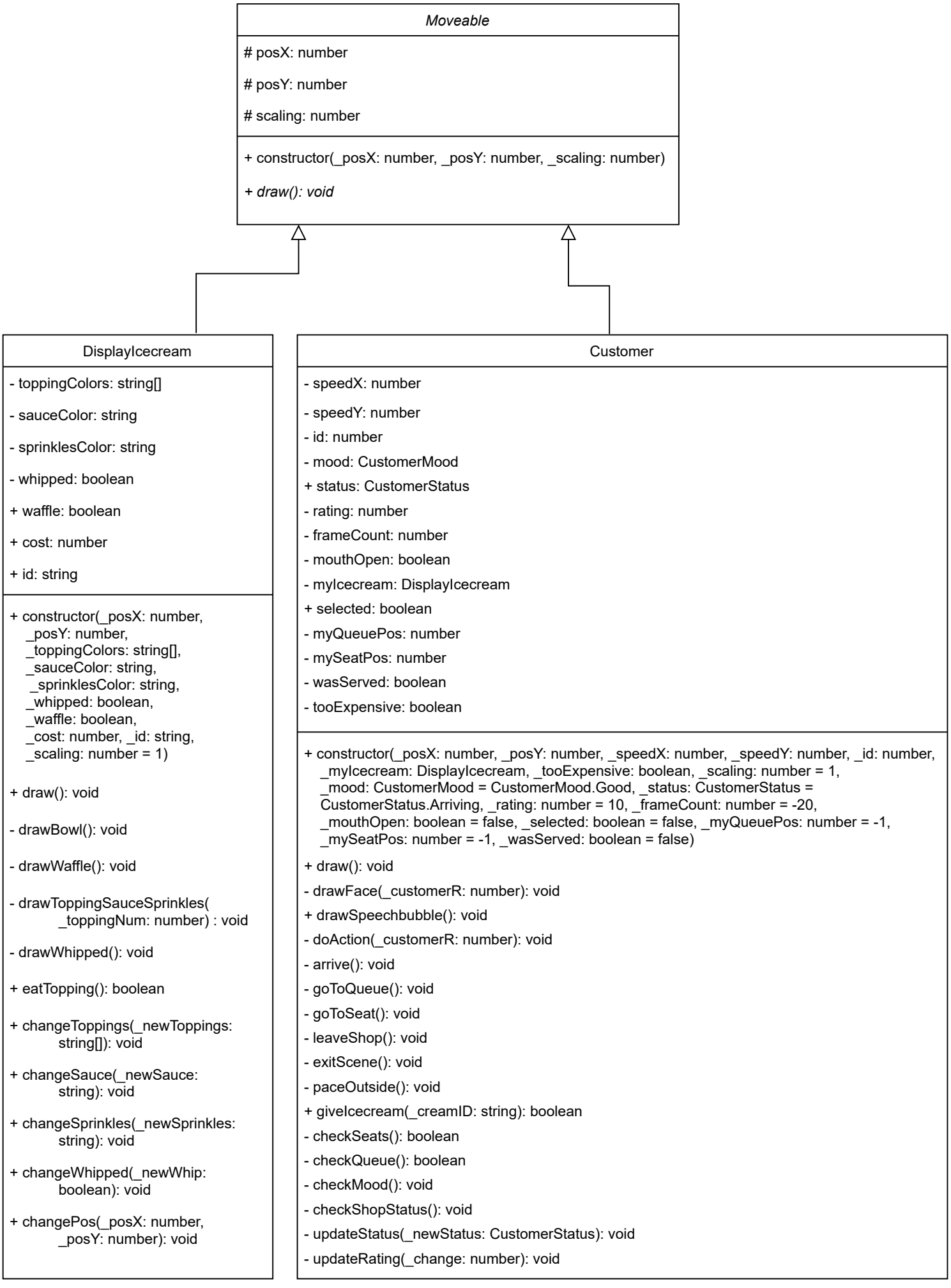
ClassDiagram

Jona Ruder



« interface » FullIcecream

title: string

price: number

toppings: string

toppingsAmount: number

sauce: number

hasSauce: boolean

sprinklesType: number

whippedCream: boolean

id: string

« interface » FormDataJSON

[key: string]: FormDataEntryValue | FormDataEntryValue[]

« enum » CreamTypes

Chocolate

Vanilla

Strawberry

Blueberry

Banana

Smurf

« enum » CustomerStatus

Arriving

WaitingOutside

GoingToQueue

AskingForIcecream

WaitingInside

GoingToSeat

Eating

Leaving

Reviewing

« enum » CustomerMood

Bad

Okay

Good

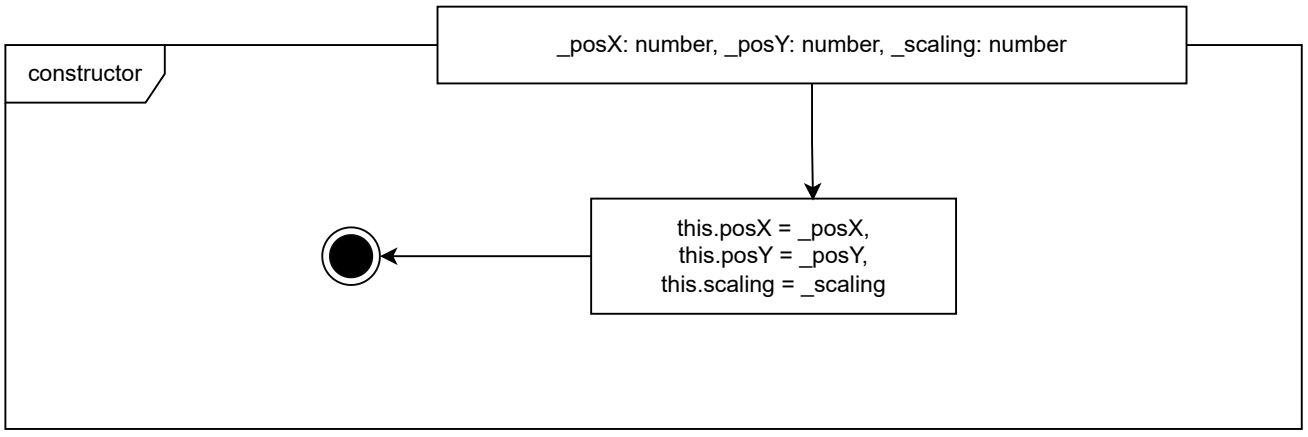
« enum » SprinklesType

None

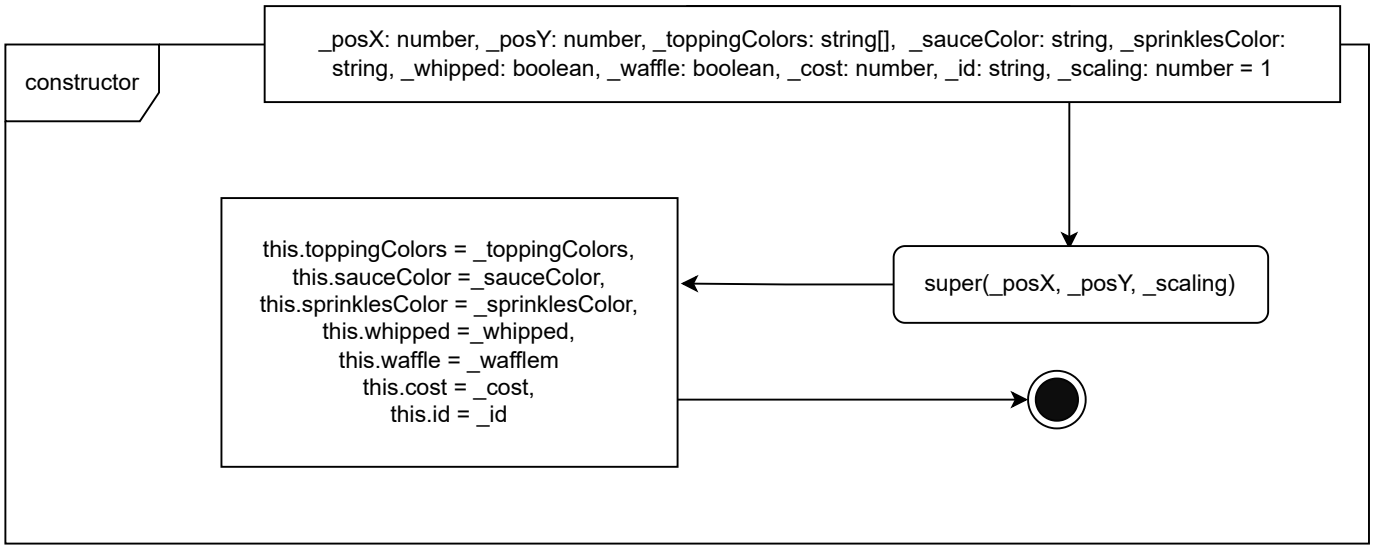
Chocolate

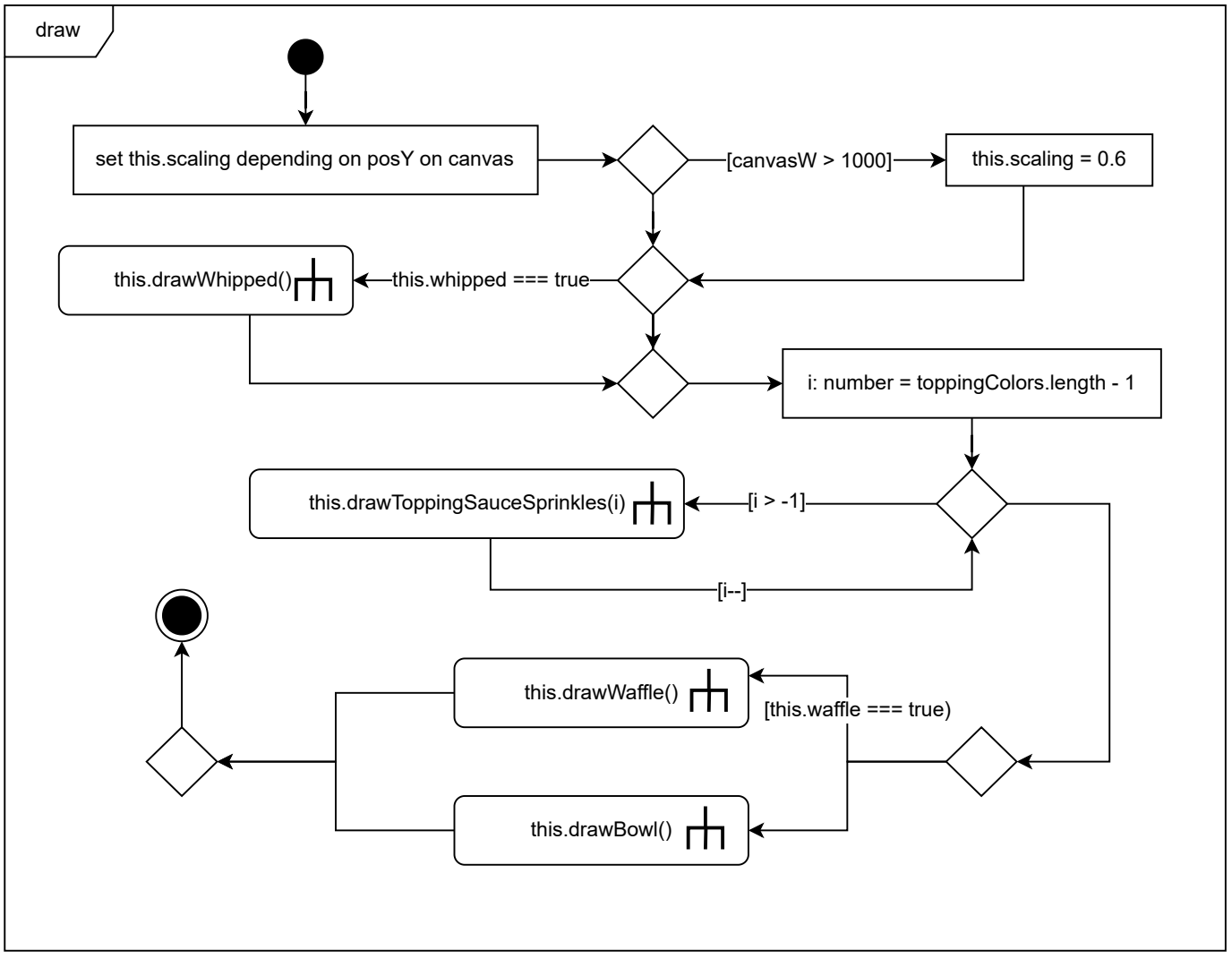
Mint

Moveable:

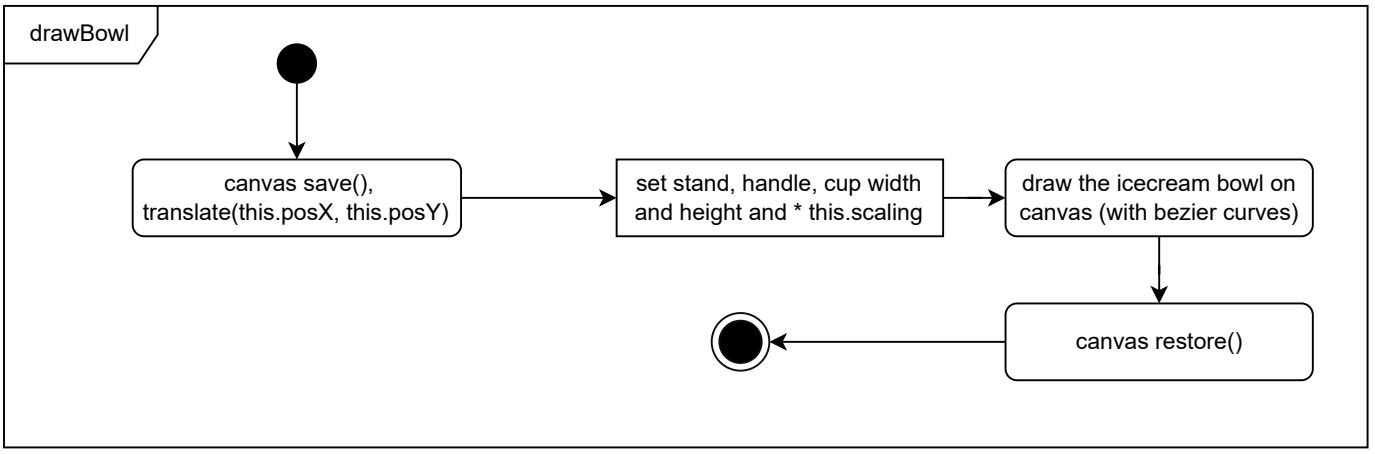


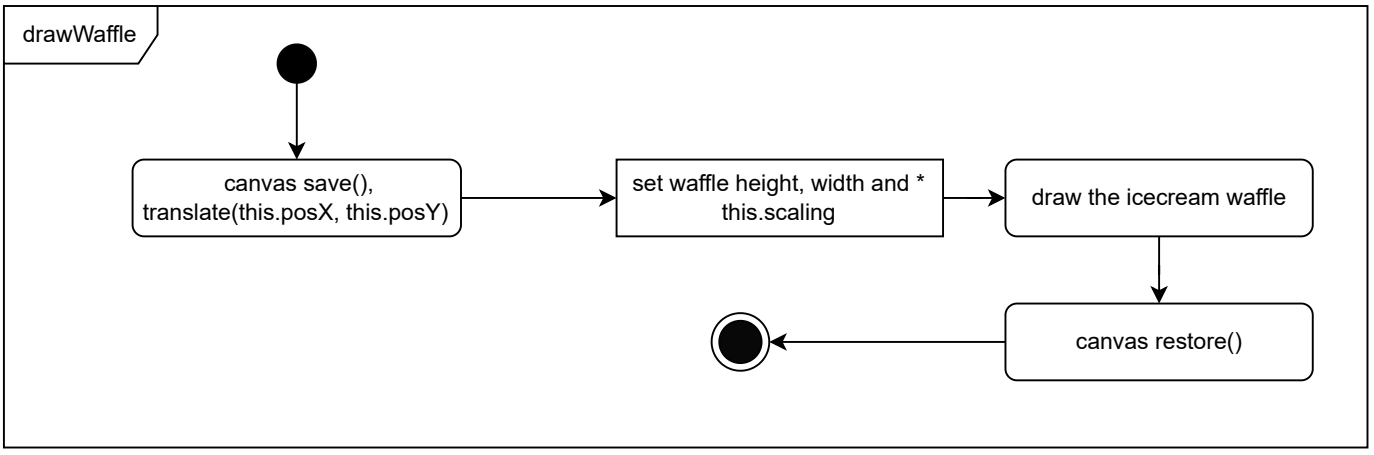
DisplayIcecream: (1)

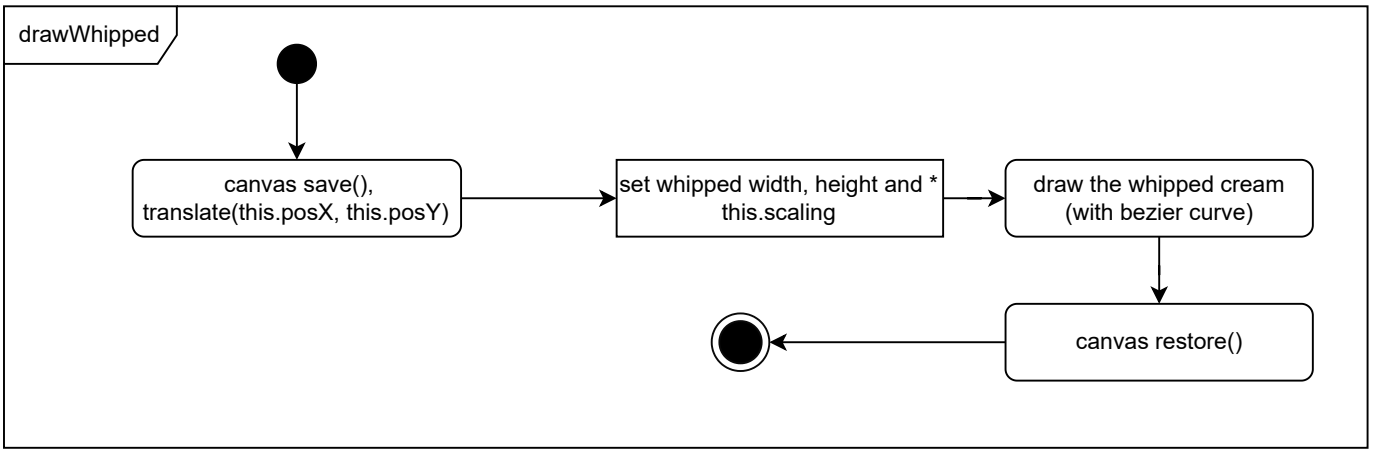


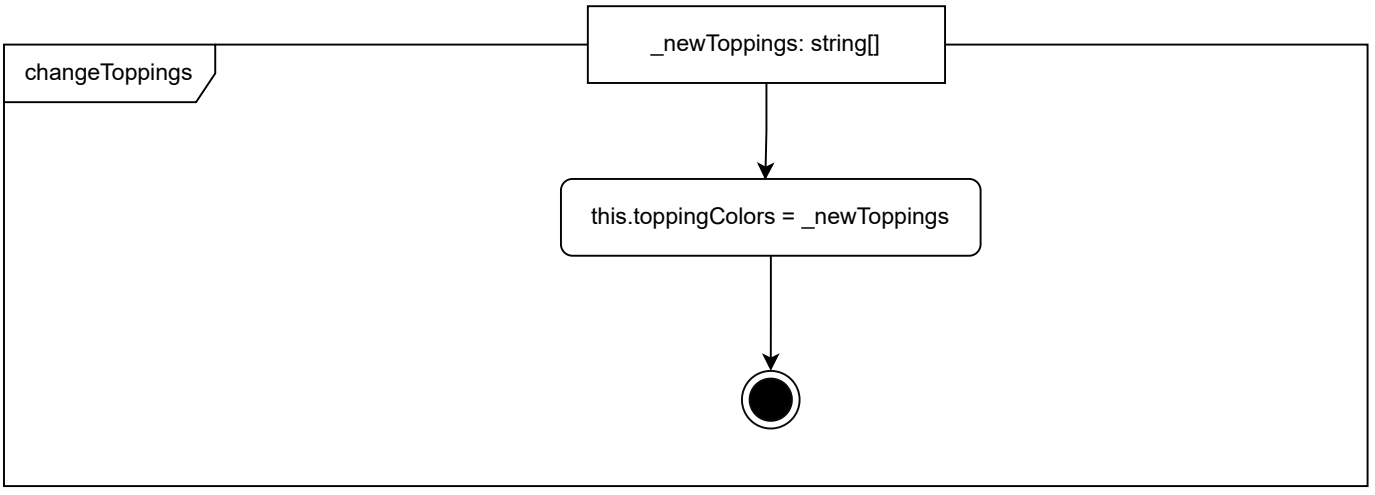


DisplayIcecream: (2)

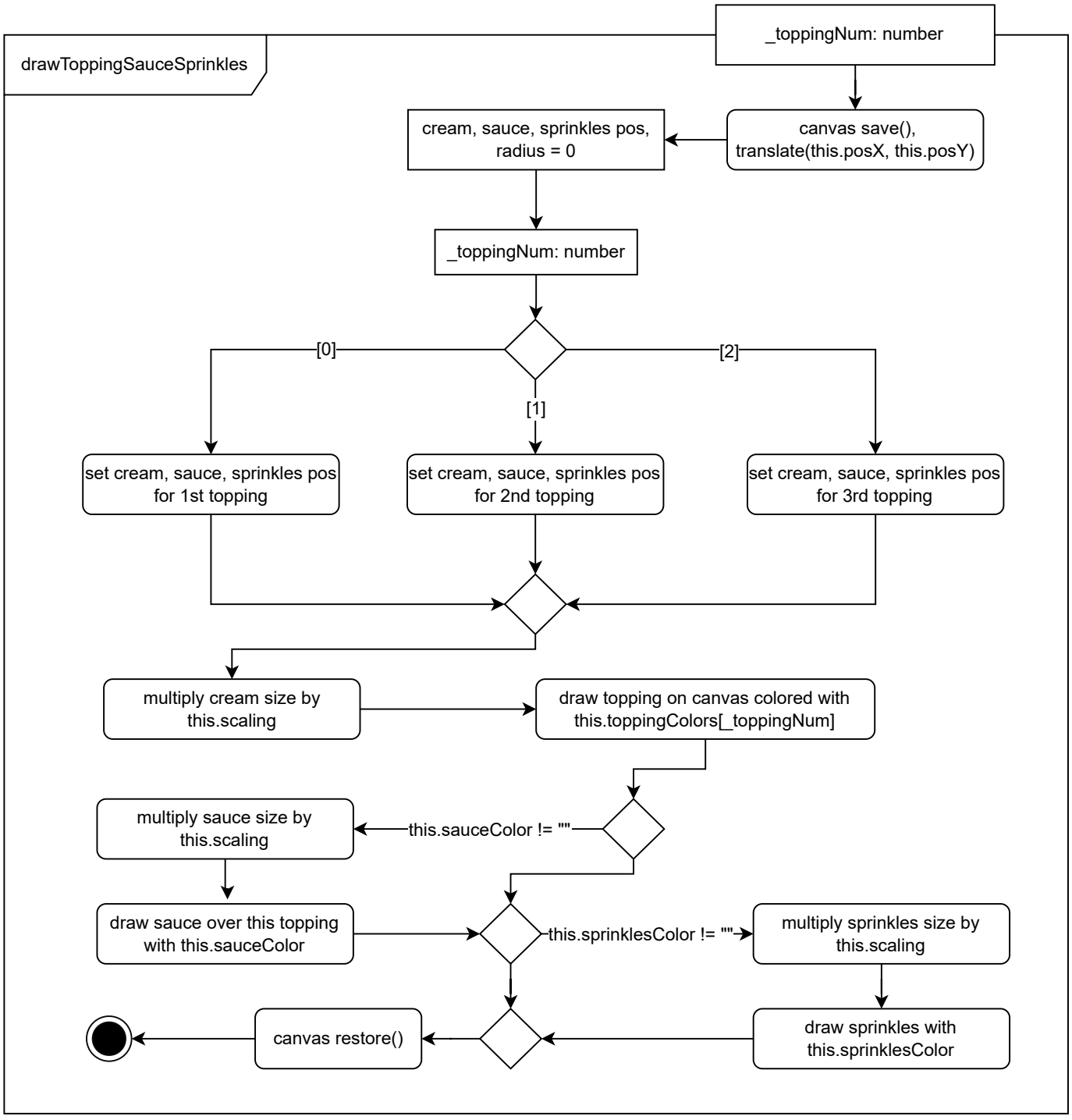


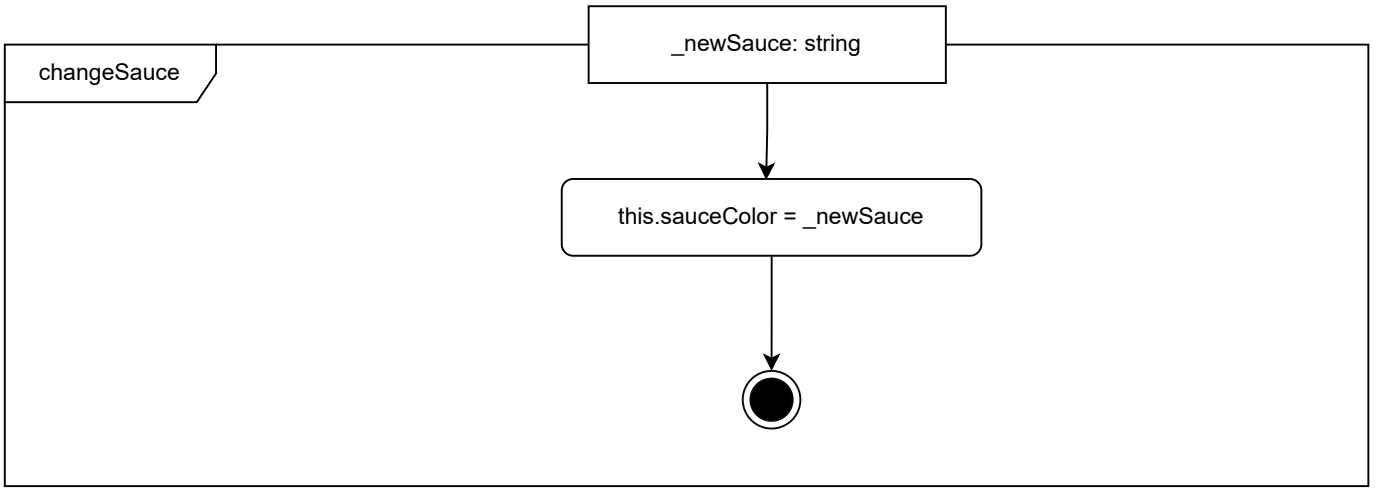




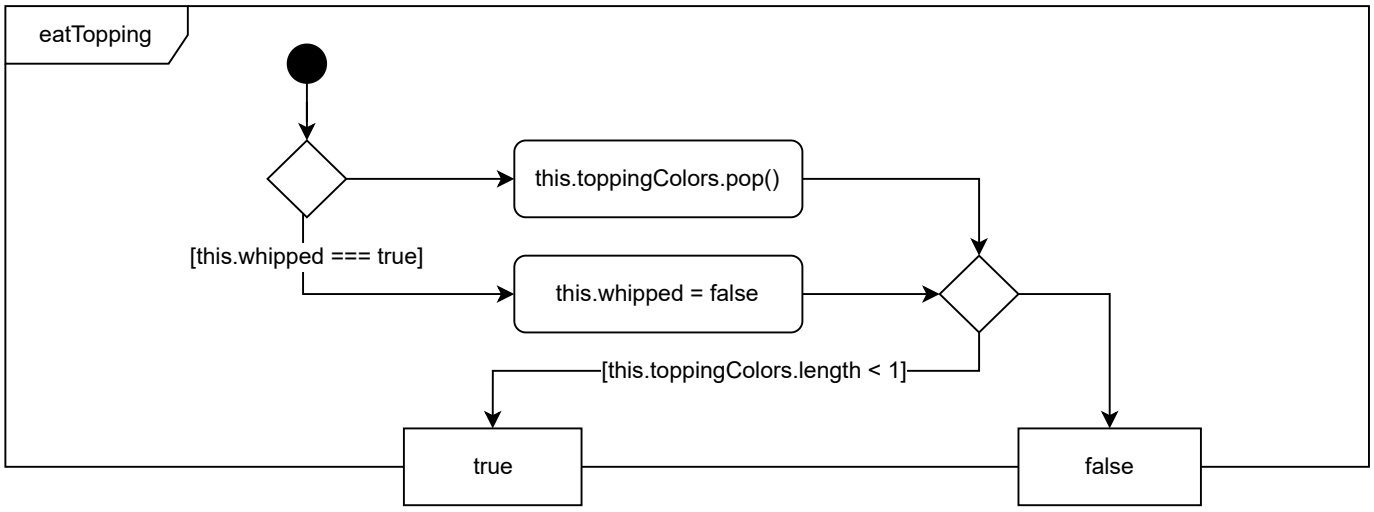


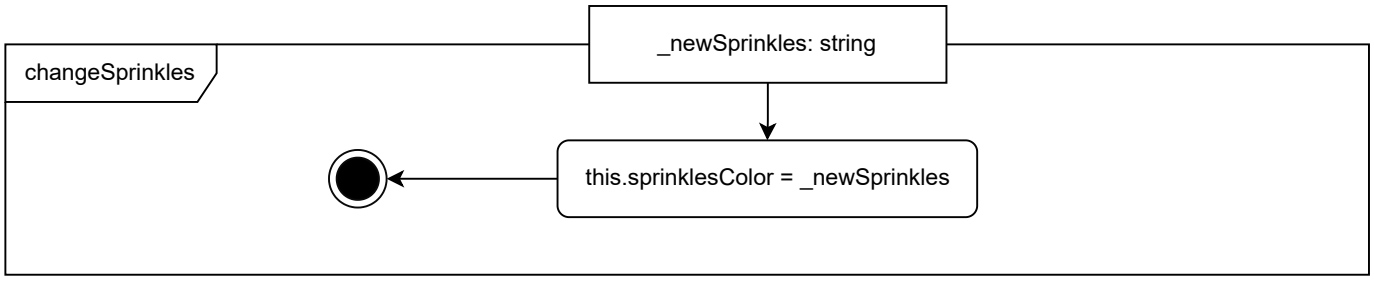
DisplayIcecream: (3)

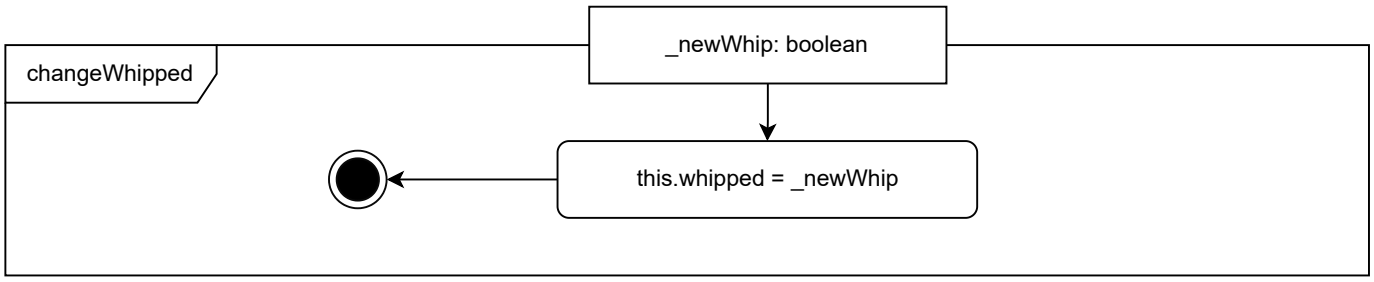


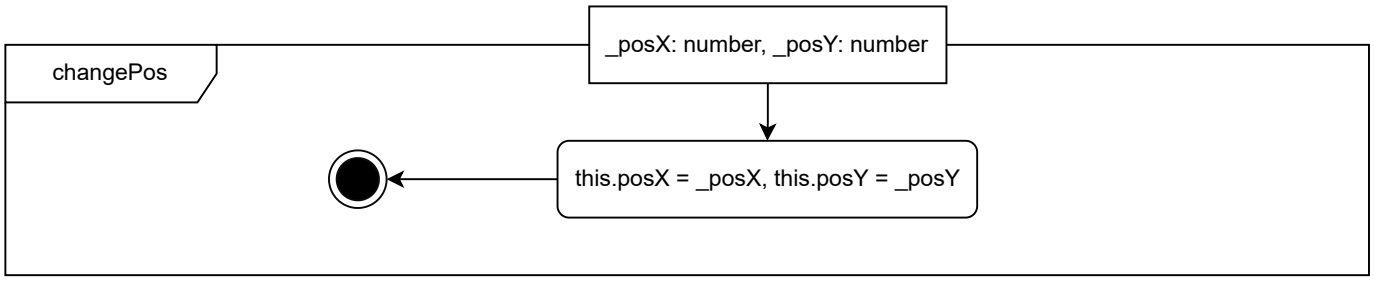


DisplayIcecream: (4)

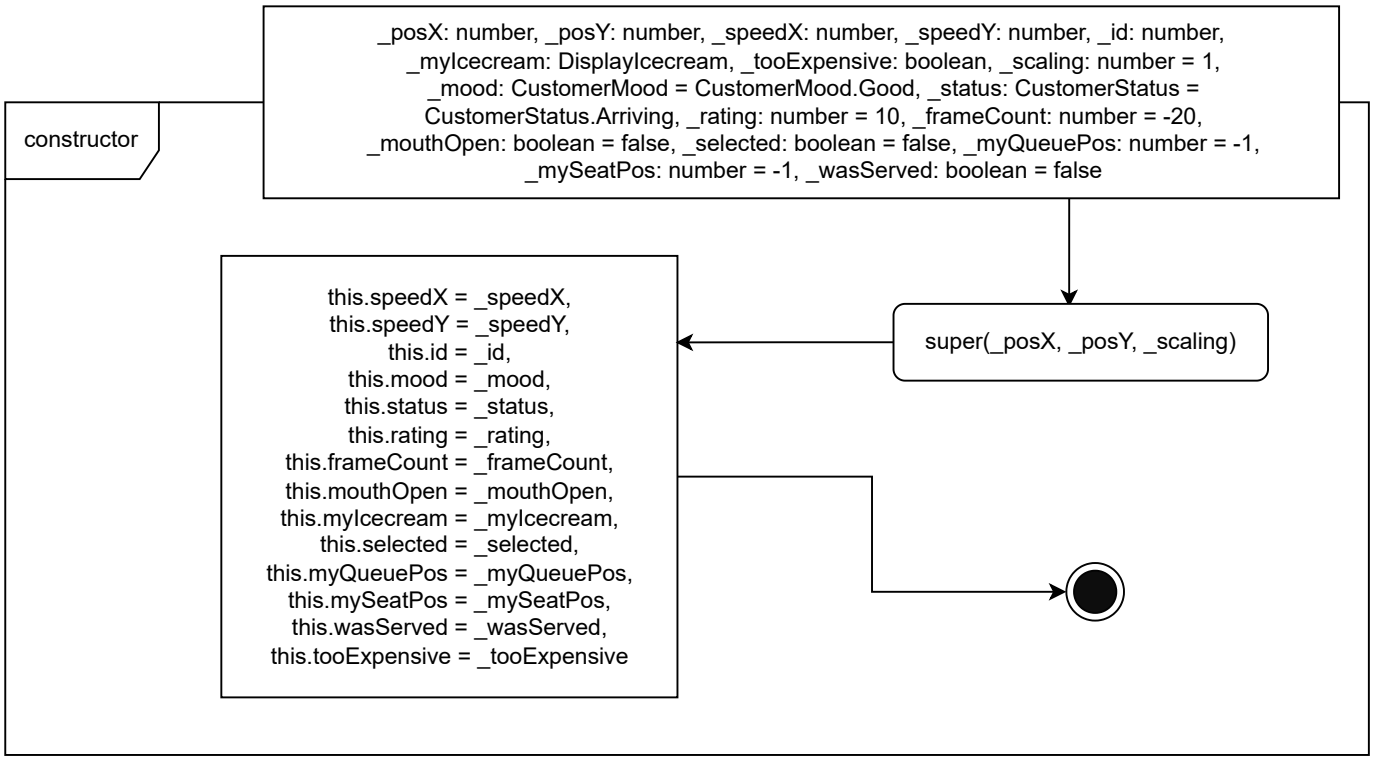


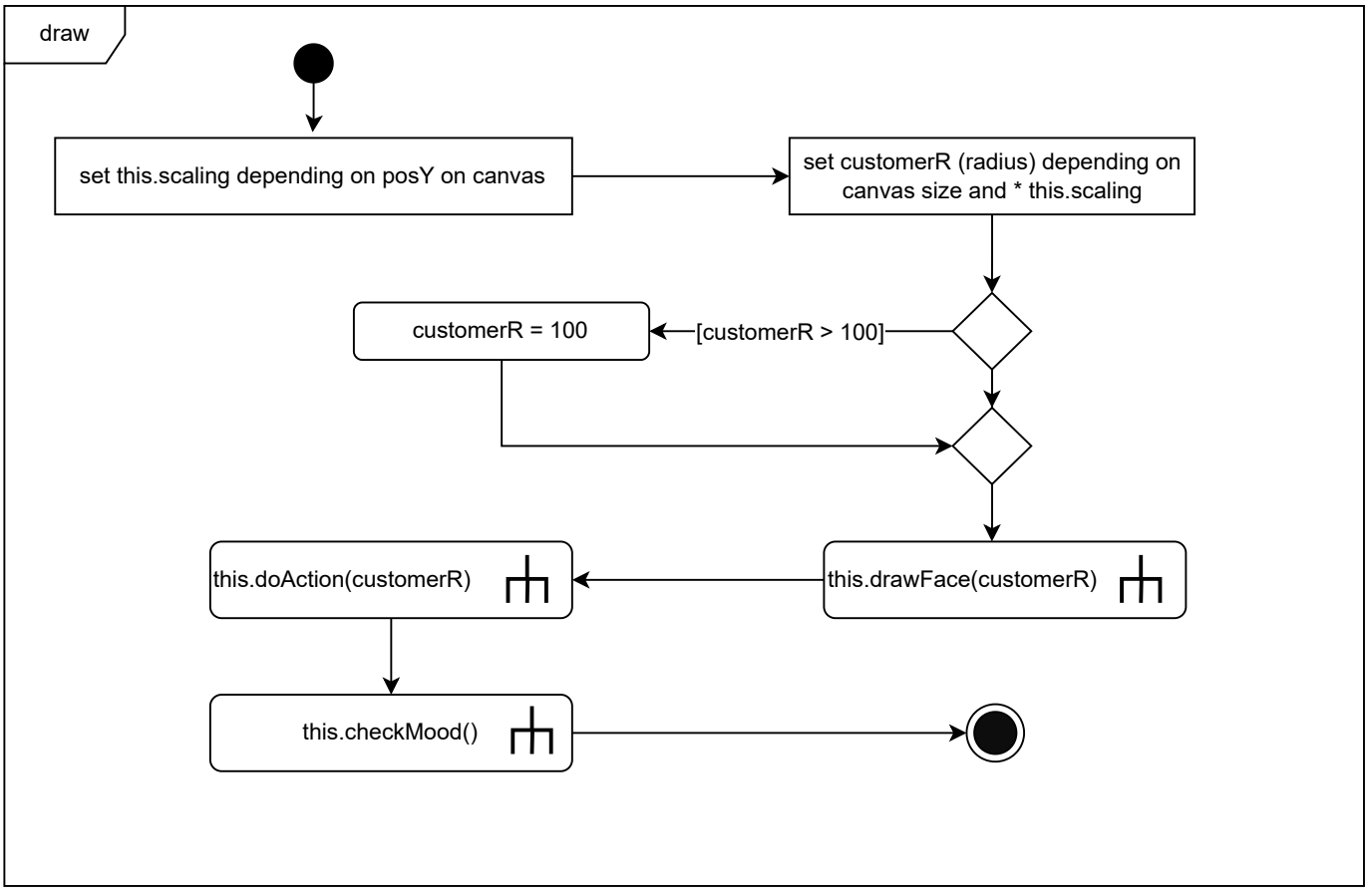




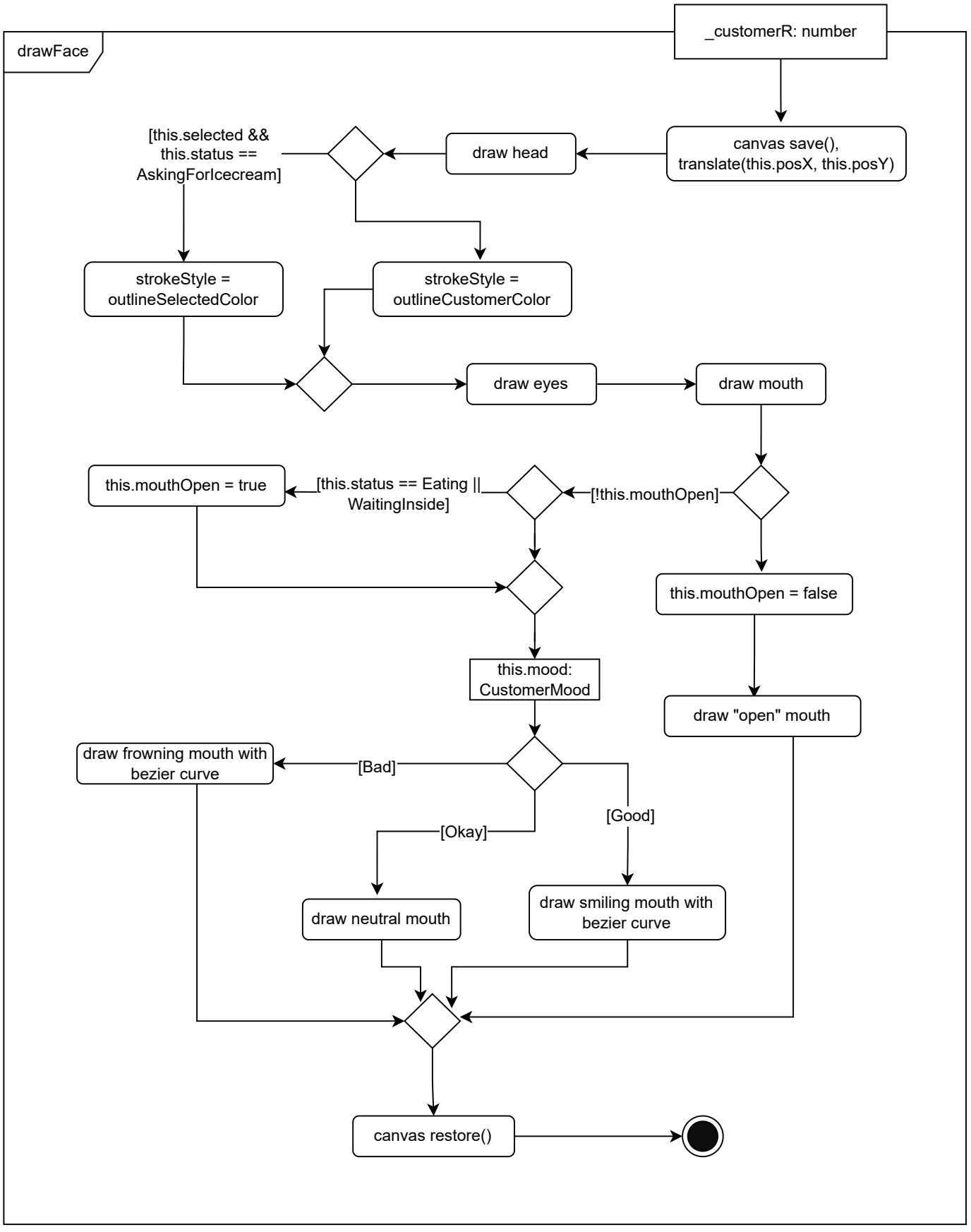


Customer: (1)

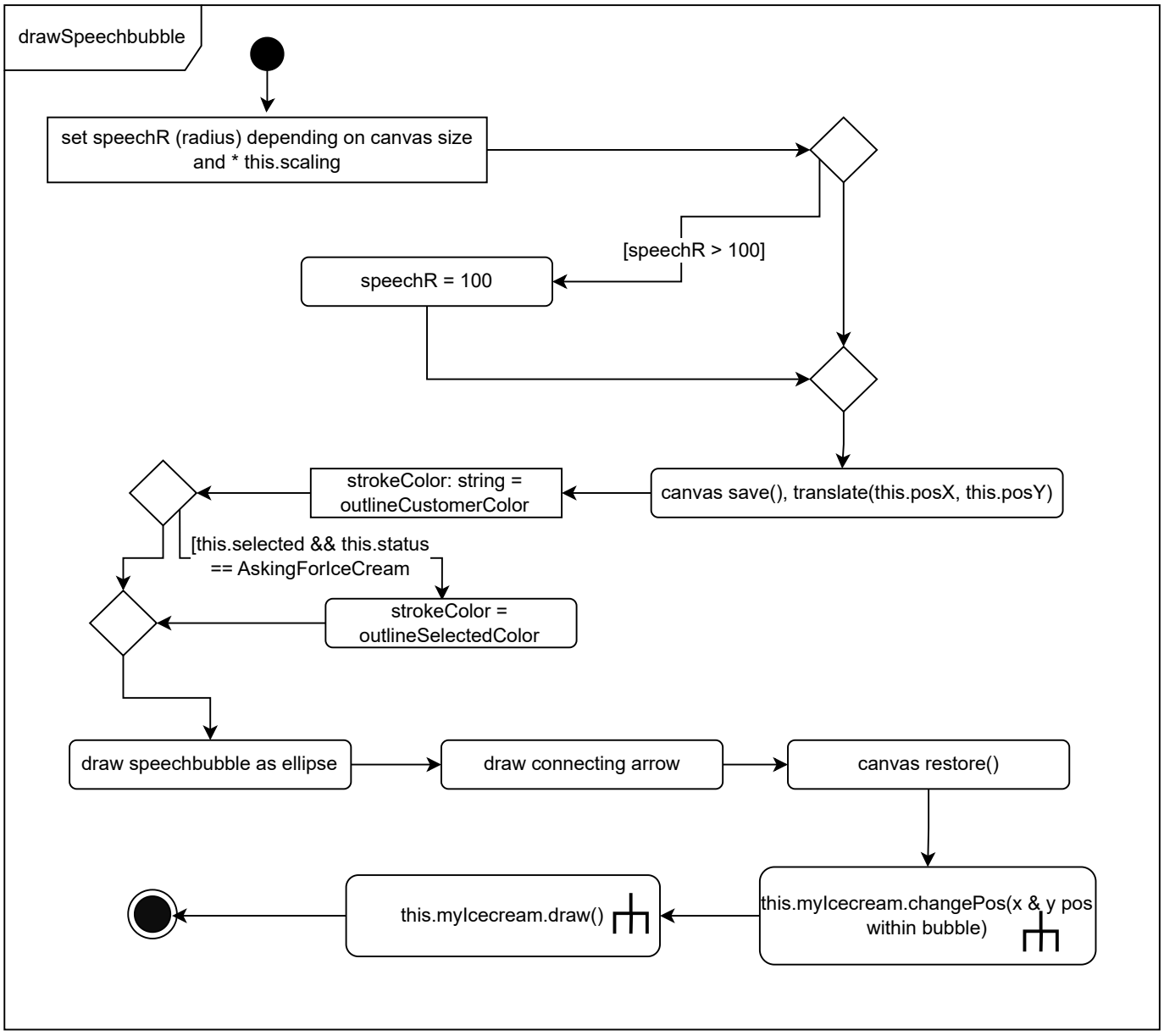

constructor
_posX: number, _posY: number, _speedX: number, _speedY: number, _id: number,
_myIcecream: DisplayIcecream, _tooExpensive: boolean, _scaling: number = 1,
_mood: CustomerMood = CustomerMood.Good, _status: CustomerStatus =
CustomerStatus.Arriving, _rating: number = 10, _frameCount: number = -20,
_mouthOpen: boolean = false, _selected: boolean = false, _myQueuePos: number = -1,
_mySeatPos: number = -1, _wasServed: boolean = false
super(_posX, _posY, _scaling)
this.speedX = _speedX,
this.speedY = _speedY,
this.id = _id,
this.mood = _mood,
this.status = _status,
this.rating = _rating,
this.frameCount = _frameCount,
this.mouthOpen = _mouthOpen,
this.myIcecream = _myIcecream,
this.selected = _selected,
this.myQueuePos = _myQueuePos,
this.mySeatPos = _mySeatPos,
this.wasServed = _wasServed,
this.tooExpensive = _tooExpensive

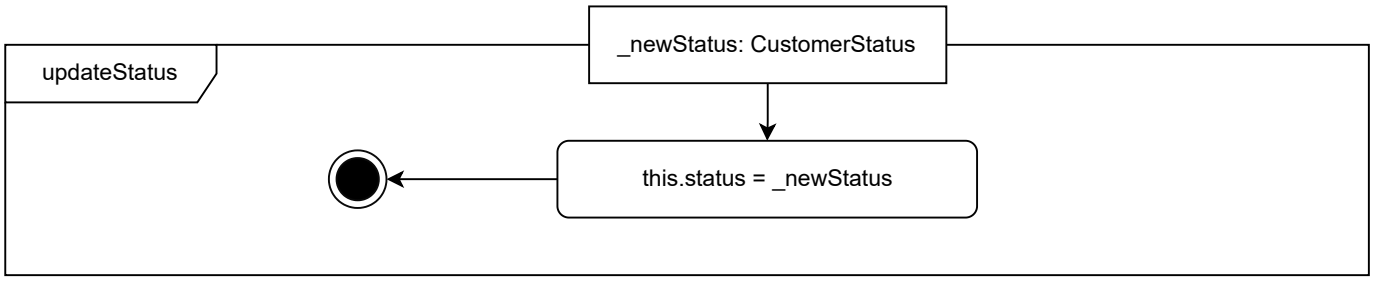

draw
set this.scaling depending on posY on canvas
set customerR (radius) depending on
canvas size and * this.scaling
customerR = 100
[customerR > 100]
this.doAction(customerR)
this.drawFace(customerR)
this.checkMood()

Customer: (2)

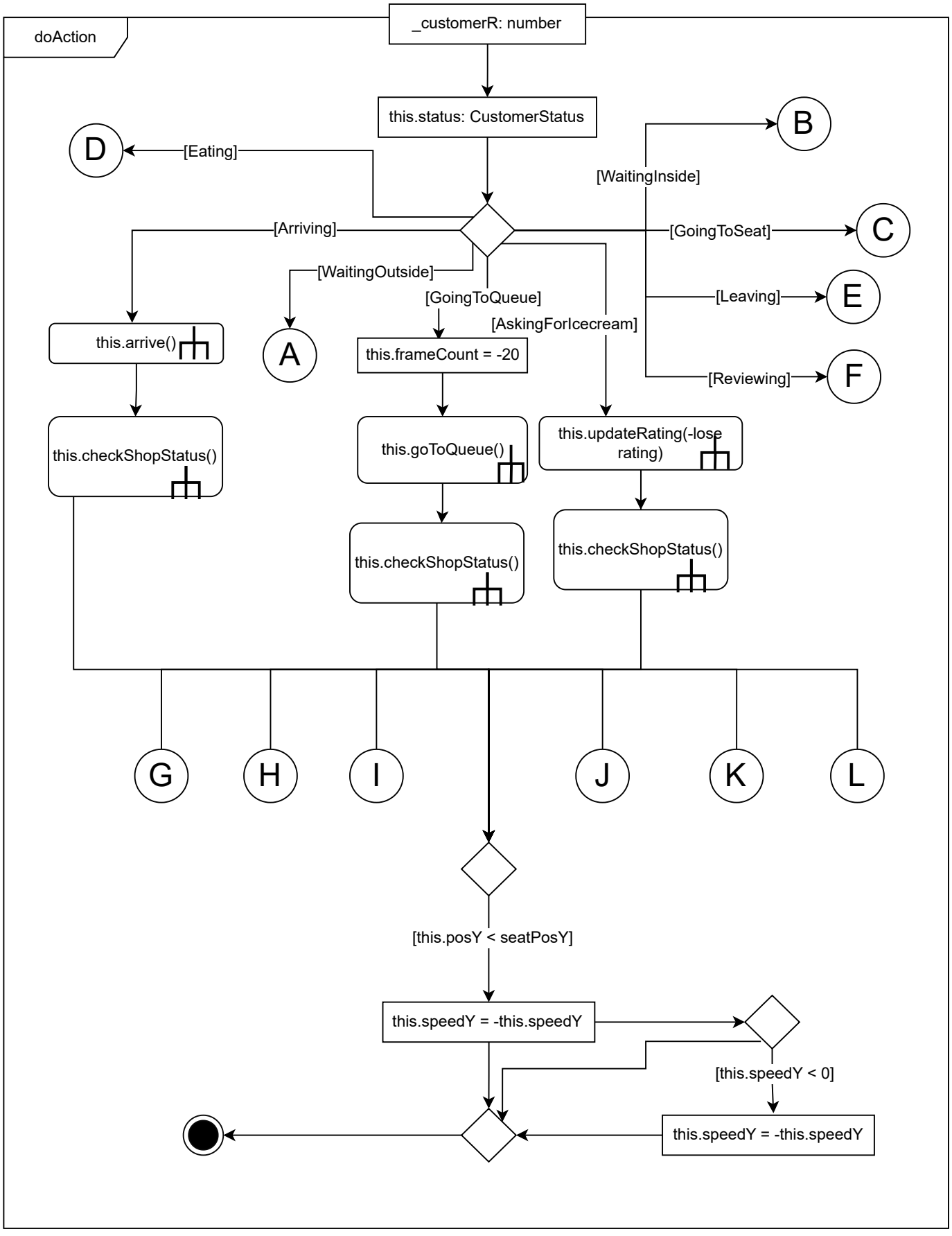

drawFace
_customerR: number
canvas save(), translate(this.posX, this.posY)
draw head
[this.selected && this.status == AskingForIcecream]
strokeStyle = outlineSelectedColor
strokeStyle = outlineCustomerColor
draw eyes
draw mouth
[!this.mouthOpen]
this.mouthOpen = false
[this.status == Eating || WaitingInside]
this.mouthOpen = true
this.mood: CustomerMood
draw "open" mouth
[Bad]
draw frowning mouth with bezier curve
[Okay]
[Good]
draw neutral mouth
draw smiling mouth with bezier curve
canvas restore()

Customer: (3)

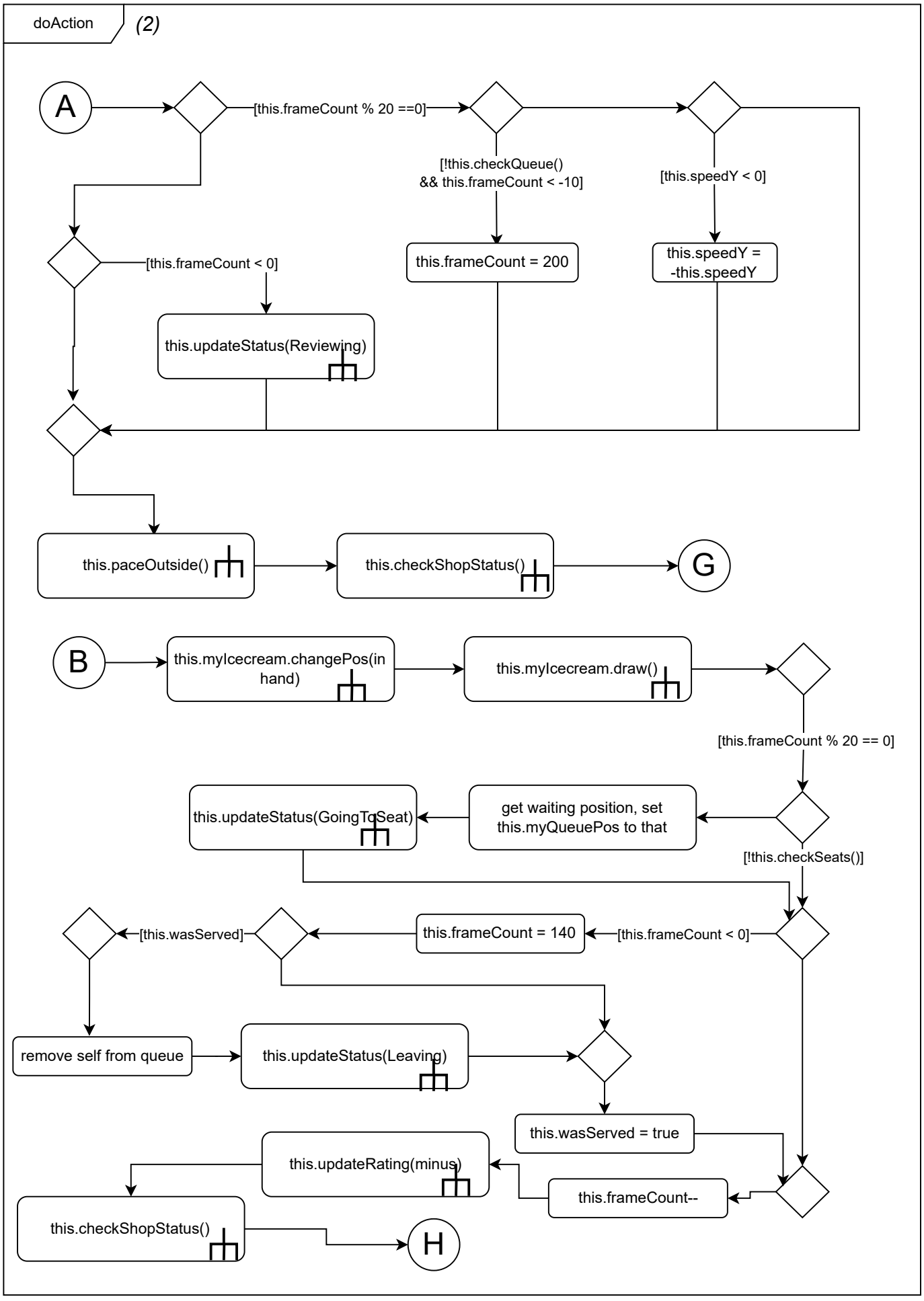

drawSpeechbubble
set speechR (radius) depending on canvas size and * this.scaling
[speechR > 100]
speechR = 100
canvas save(), translate(this.posX, this.posY)
strokeColor: string = outlineCustomerColor
[this.selected && this.status == AskingForIceCream
strokeColor = outlineSelectedColor
draw speechbubble as ellipse
draw connecting arrow
canvas restore()
this.myIcecream.changePos(x & y pos within bubble)
this.myIcecream.draw()


updateStatus
_newStatus: CustomerStatus
this.status = _newStatus

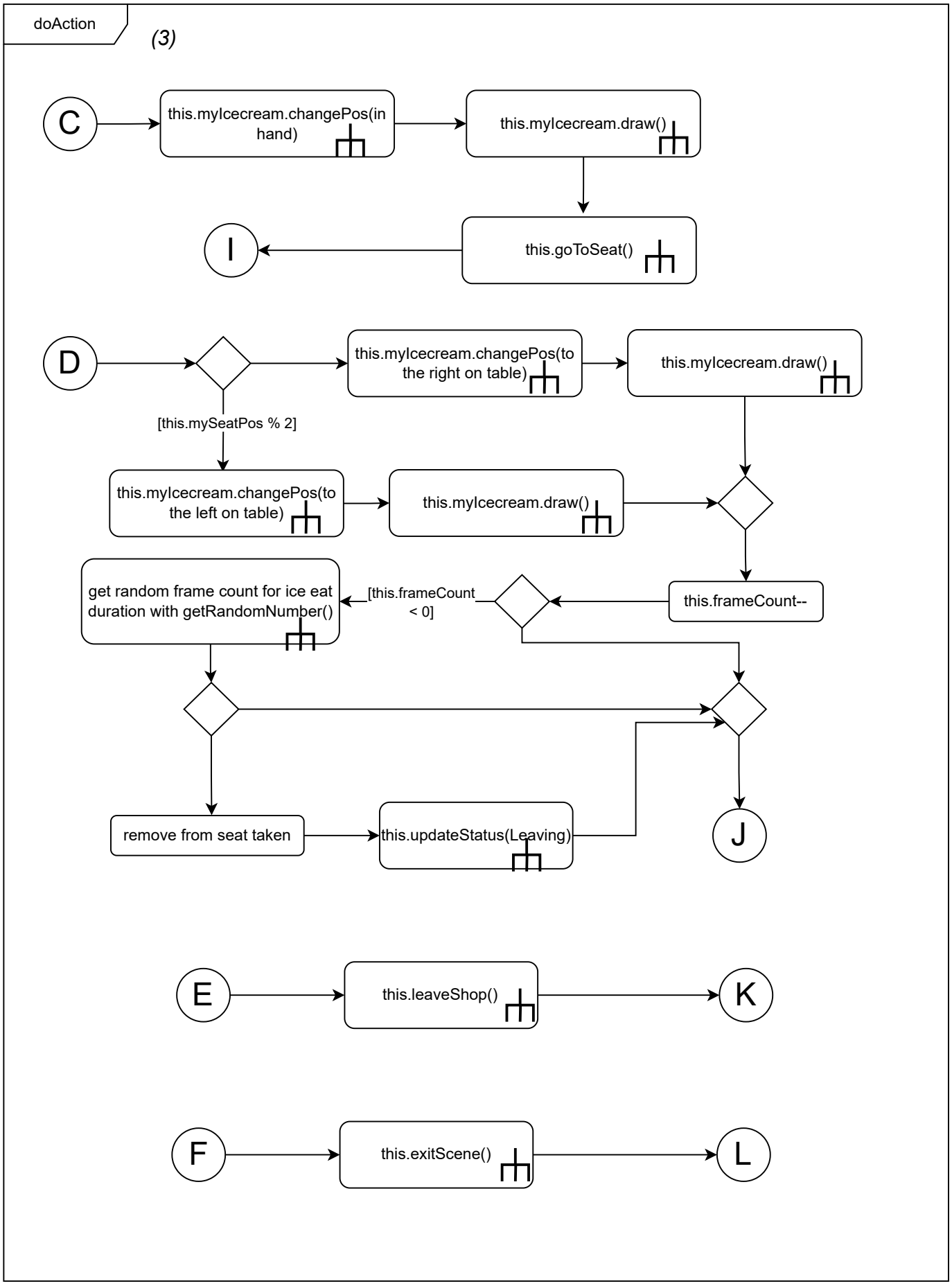
Customer: (4)


doAction
_customerR: number
this.status: CustomerStatus
D
[Eating]
B
[WaitingInside]
[Arriving]
[GoingToSeat]
C
[WaitingOutside]
[GoingToQueue]
[Leaving]
E
[AskingForIcecream]
this.arrive()
A
this.frameCount = -20
[Reviewing]
F
this.checkShopStatus()
this.goToQueue()
this.updateRating(-lose rating)
this.checkShopStatus()
this.checkShopStatus()
G
H
I
J
K
L
[this.posY < seatPosY]
this.speedY = -this.speedY
[this.speedY < 0]
this.speedY = -this.speedY

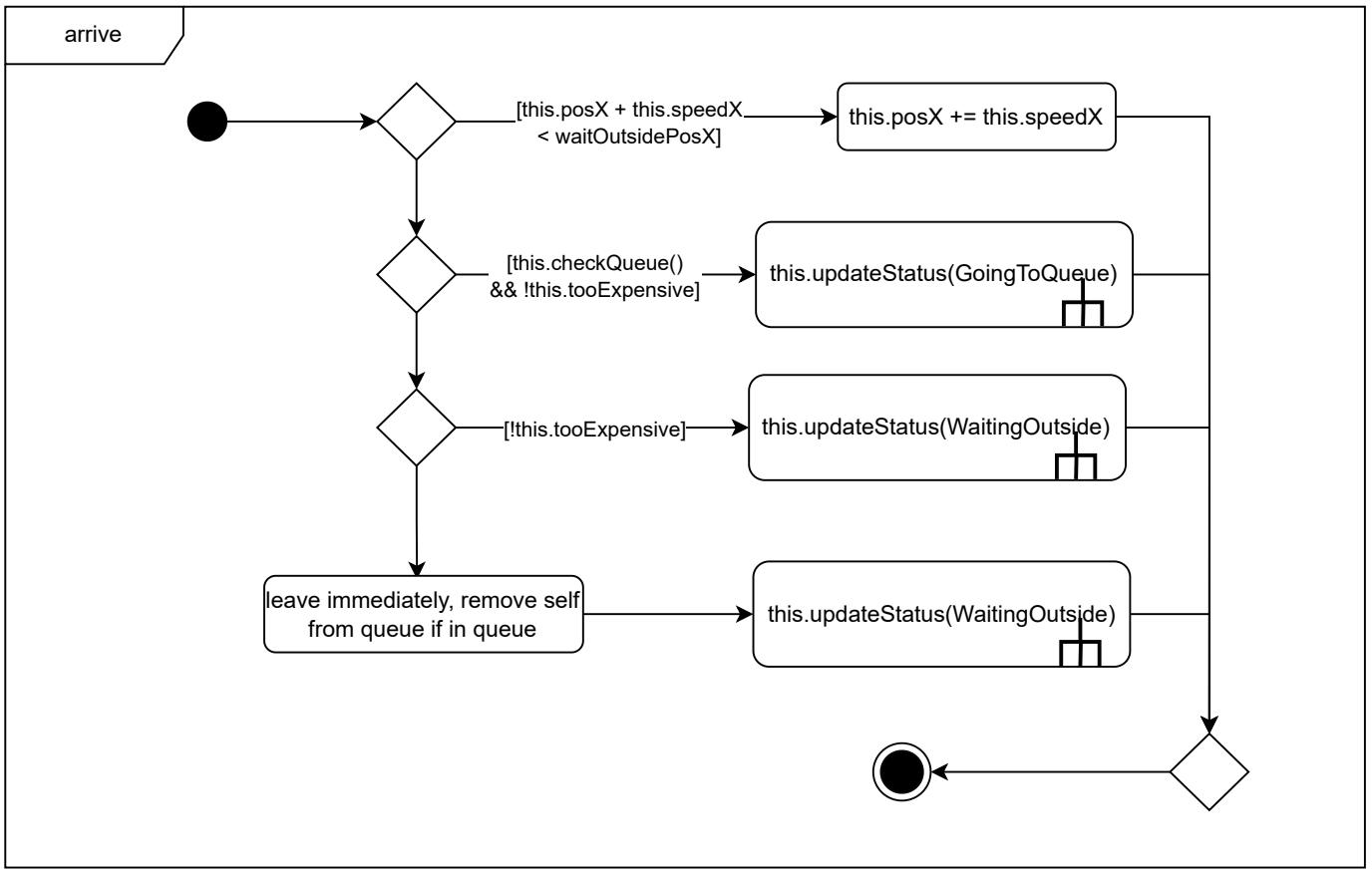
Customer: (5)

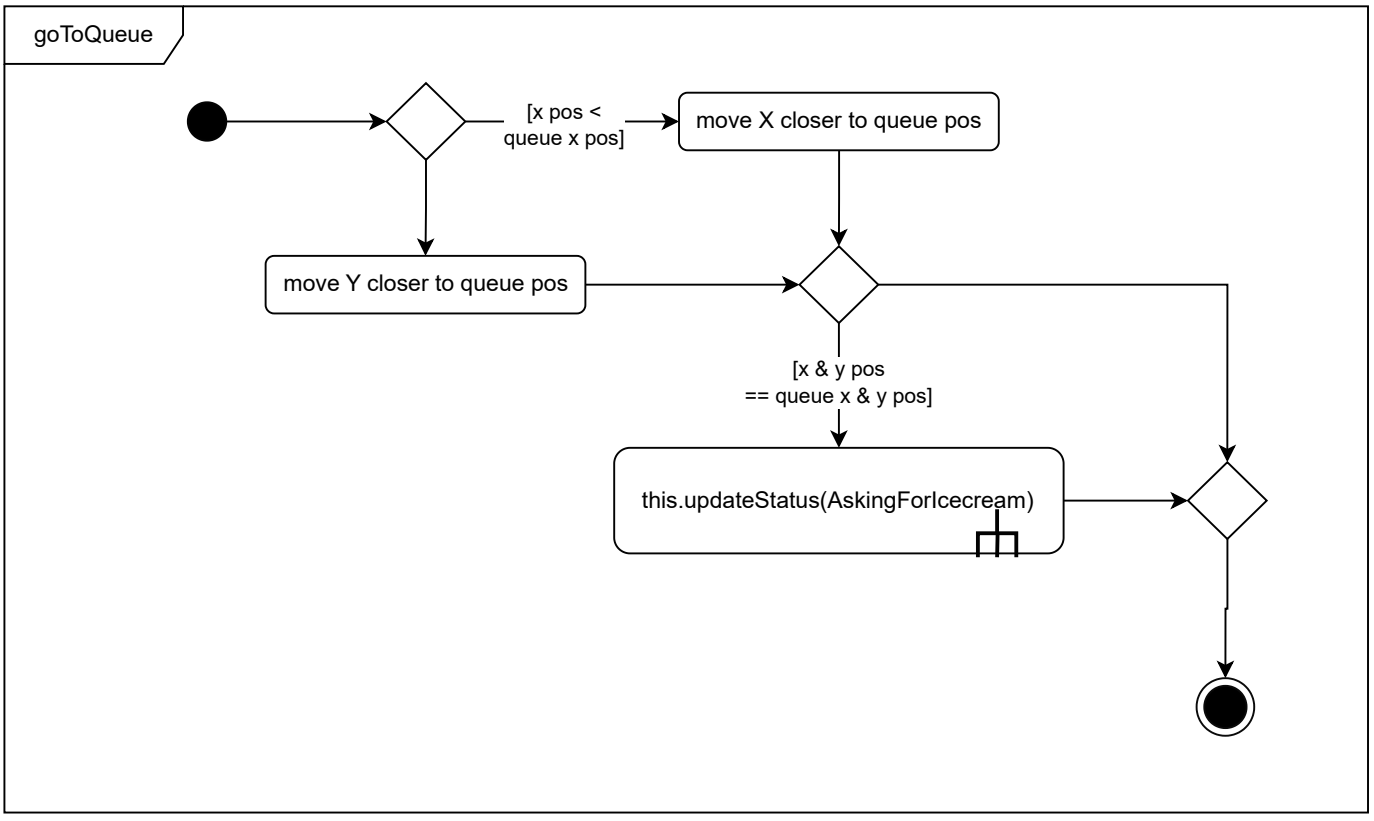

doAction
(2)
A
[this.frameCount % 20 ==0]
[!this.checkQueue()
&& this.frameCount < -10]
[this.speedY < 0]
[this.frameCount < 0]
this.frameCount = 200
this.speedY =
-this.speedY
this.updateStatus(Reviewing)
this.paceOutside()
this.checkShopStatus()
G
B
this.myIcecream.changePos(in
hand)
this.myIcecream.draw()
[this.frameCount % 20 == 0]
this.updateStatus(GoingToSeat)
get waiting position, set
this.myQueuePos to that
[!this.checkSeats()]
[this.wasServed]
this.frameCount = 140
[this.frameCount < 0]
remove self from queue
this.updateStatus(Leaving)
this.wasServed = true
this.updateRating(minus)
this.frameCount--
this.checkShopStatus()
H

Customer: (6)

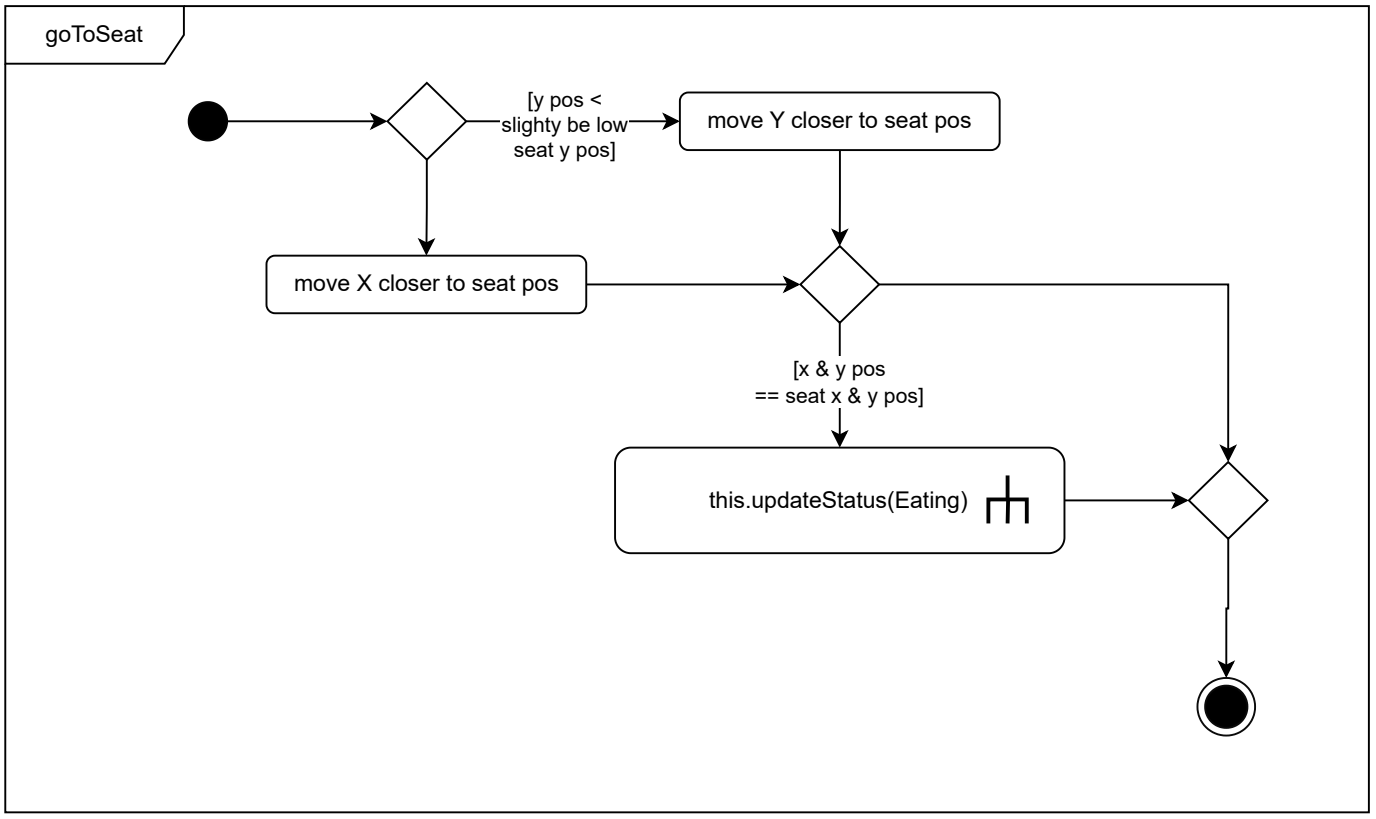

doAction
(3)
C
this.myIcecream.changePos(in hand)
this.myIcecream.draw()
this.goToSeat()
I
D
this.myIcecream.changePos(to the right on table)
this.myIcecream.draw()
[this.mySeatPos % 2]
this.myIcecream.changePos(to the left on table)
this.myIcecream.draw()
this.frameCount--
[this.frameCount < 0]
get random frame count for ice eat duration with getRandomNumber()
remove from seat taken
this.updateStatus(Leaving)
J
E
this.leaveShop()
K
F
this.exitScene()
L

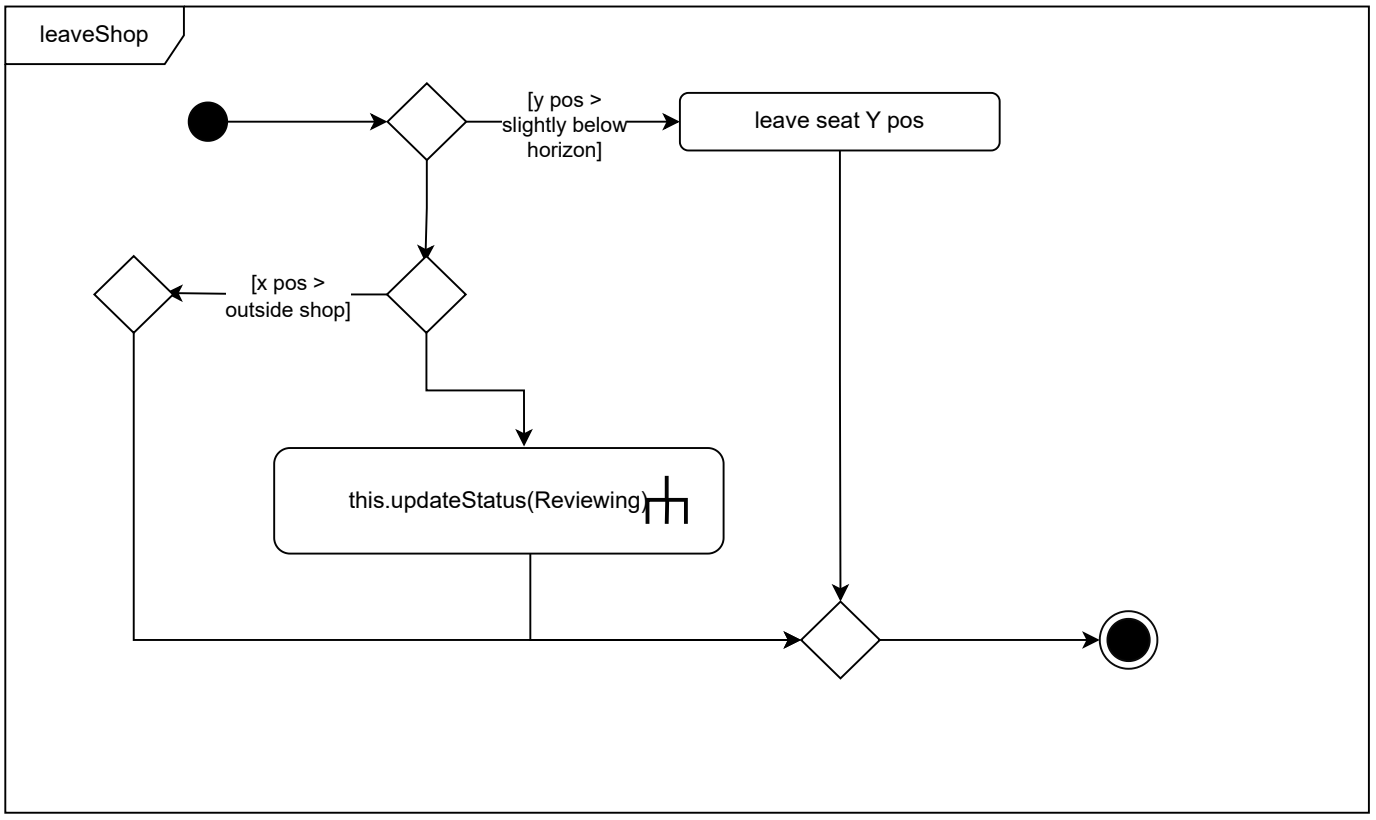
Customer: (7)


arrive
[this.posX + this.speedX
< waitOutsidePosX]
this.posX += this.speedX
[this.checkQueue()
&& !this.tooExpensive]
this.updateStatus(GoingToQueue)
[!this.tooExpensive]
this.updateStatus(WaitingOutside)
leave immediately, remove self
from queue if in queue
this.updateStatus(WaitingOutside)

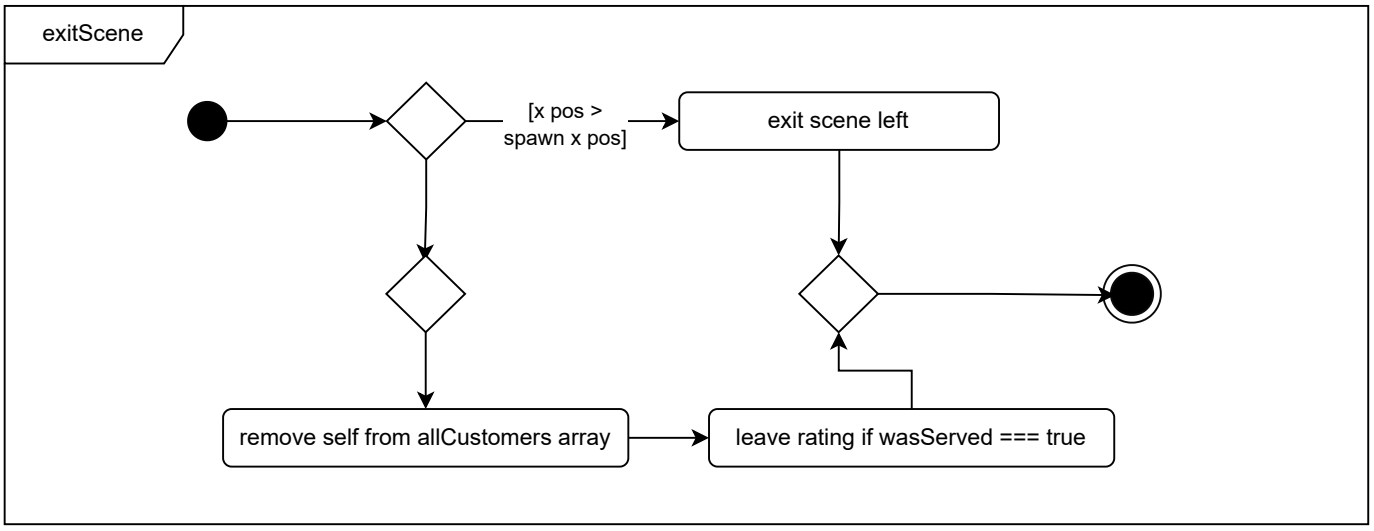

goToQueue
[x pos <
queue x pos]
move X closer to queue pos
move Y closer to queue pos
[x & y pos
== queue x & y pos]
this.updateStatus(AskingForIcecream)

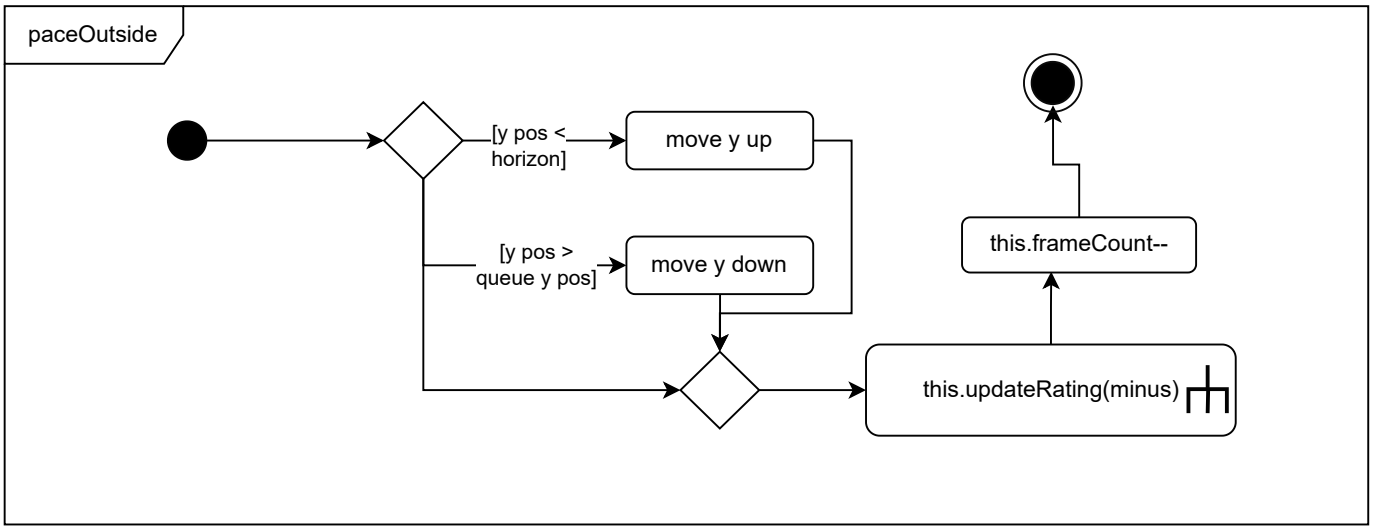
Customer: (8)

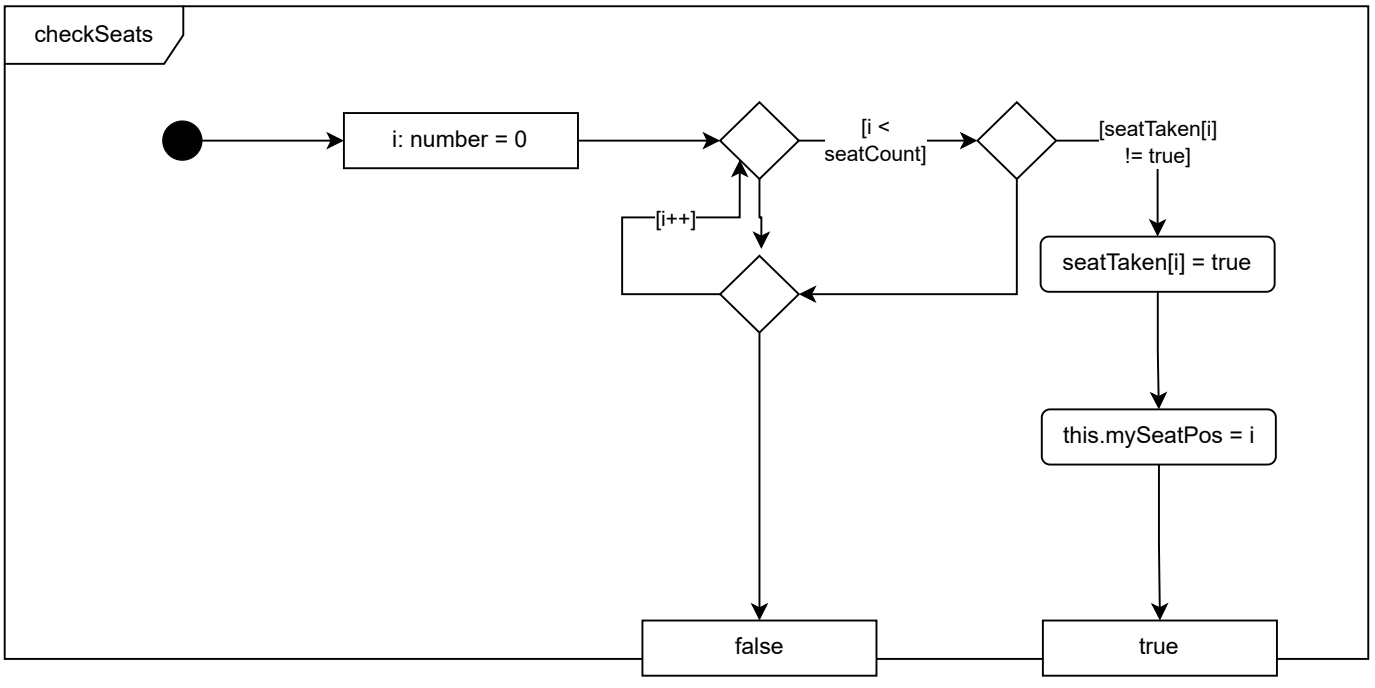

goToSeat
[y pos < slighty be low seat y pos]
move Y closer to seat pos
move X closer to seat pos
[x & y pos == seat x & y pos]
this.updateStatus(Eating)


leaveShop
[y pos > slightly below horizon]
leave seat Y pos
[x pos > outside shop]
this.updateStatus(Reviewing)

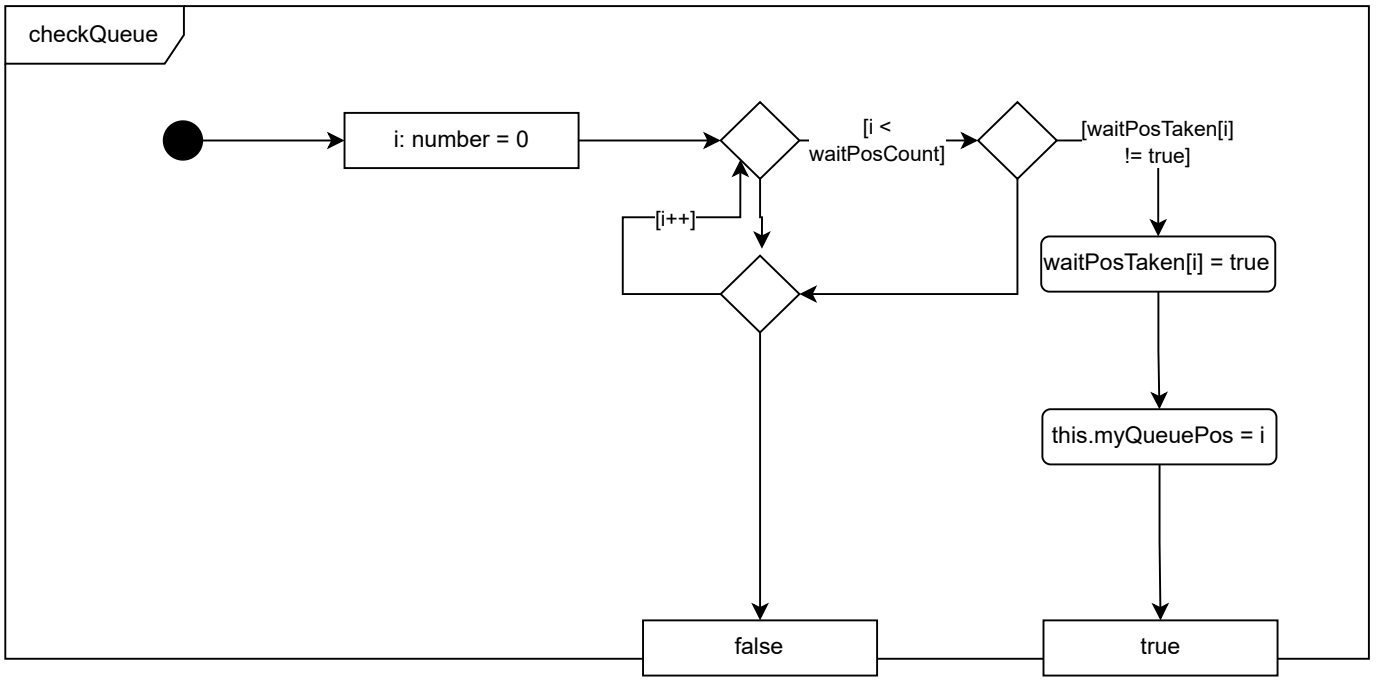
Customer: (9)

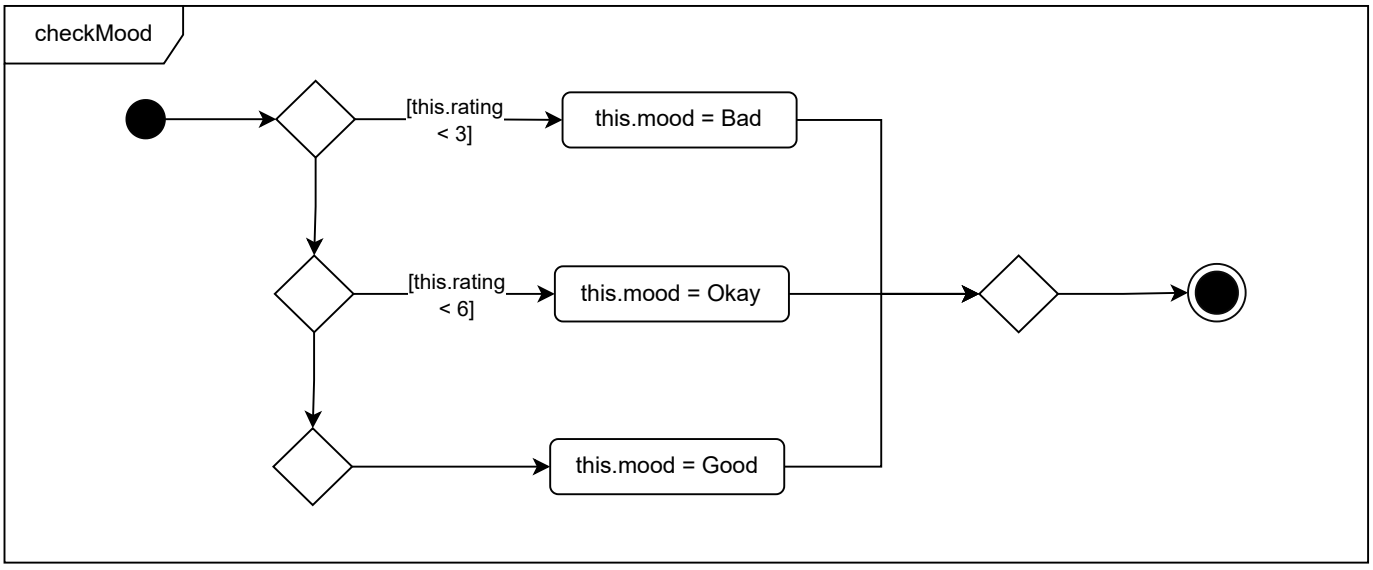

exitScene
[x pos > spawn x pos]
exit scene left
remove self from allCustomers array
leave rating if wasServed === true

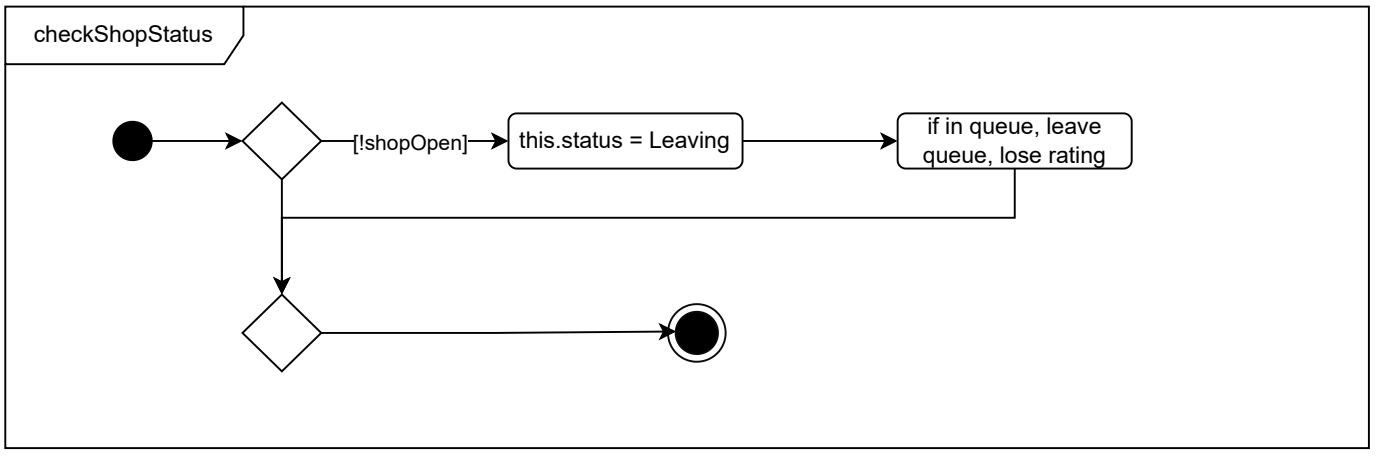

paceOutside
[y pos < horizon]
move y up
[y pos > queue y pos]
move y down
this.frameCount--
this.updateRating(minus)


checkSeats
i: number = 0
[i < seatCount]
[seatTaken[i] != true]
[i++]
seatTaken[i] = true
this.mySeatPos = i
false
true

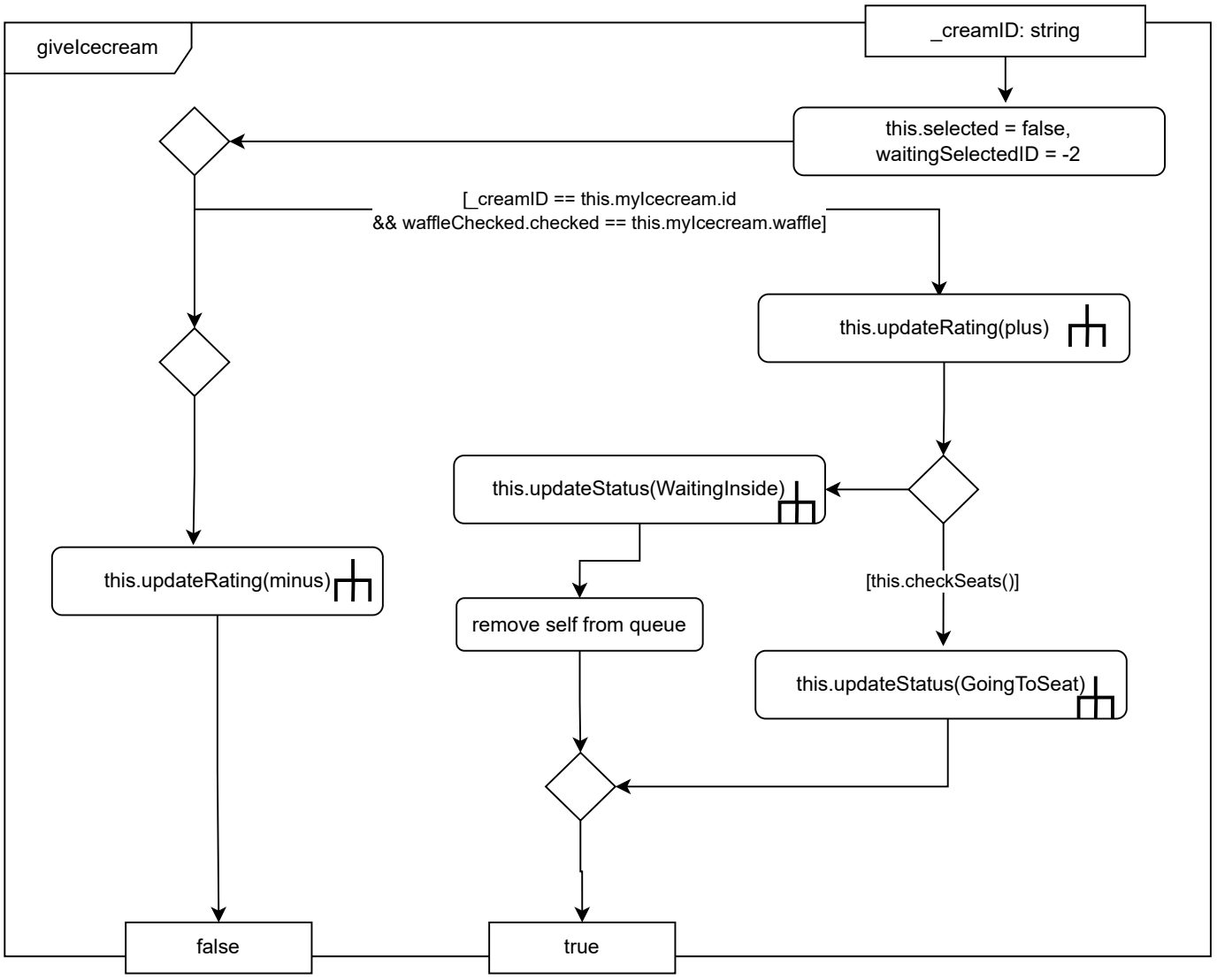
Customer: (10)

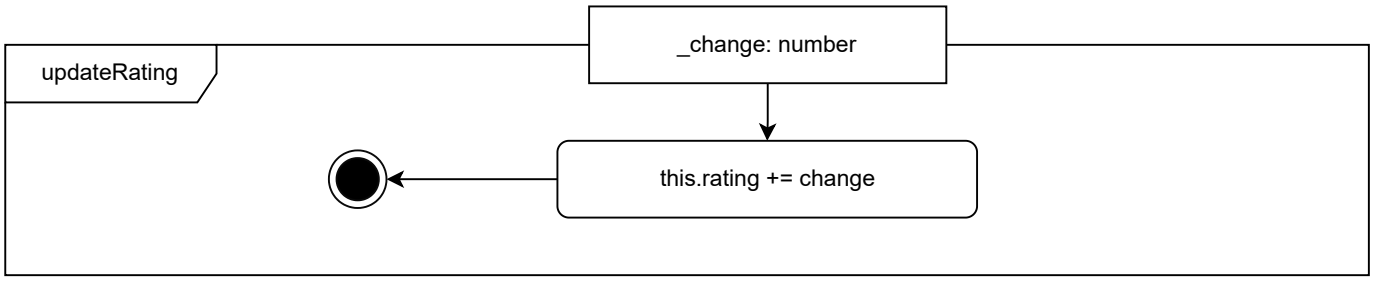

checkQueue
i: number = 0
[i <
waitPosCount]
[waitPosTaken[i]
!= true]
[i++]
waitPosTaken[i] = true
this.myQueuePos = i
false
true


checkMood
[this.rating
< 3]
this.mood = Bad
[this.rating
< 6]
this.mood = Okay
this.mood = Good


checkShopStatus
[!shopOpen]
this.status = Leaving
if in queue, leave
queue, lose rating

Customer: (11)


giveIcecream
_creamID: string
this.selected = false,
waitingSelectedID = -2
[_creamID == this.myIcecream.id
&& waffleChecked.checked == this.myIcecream.waffle]
this.updateRating(plus)
this.updateStatus(WaitingInside)
this.updateRating(minus)
remove self from queue
[this.checkSeats()]
this.updateStatus(GoingToSeat)
false
true


updateRating
_change: number
this.rating += change