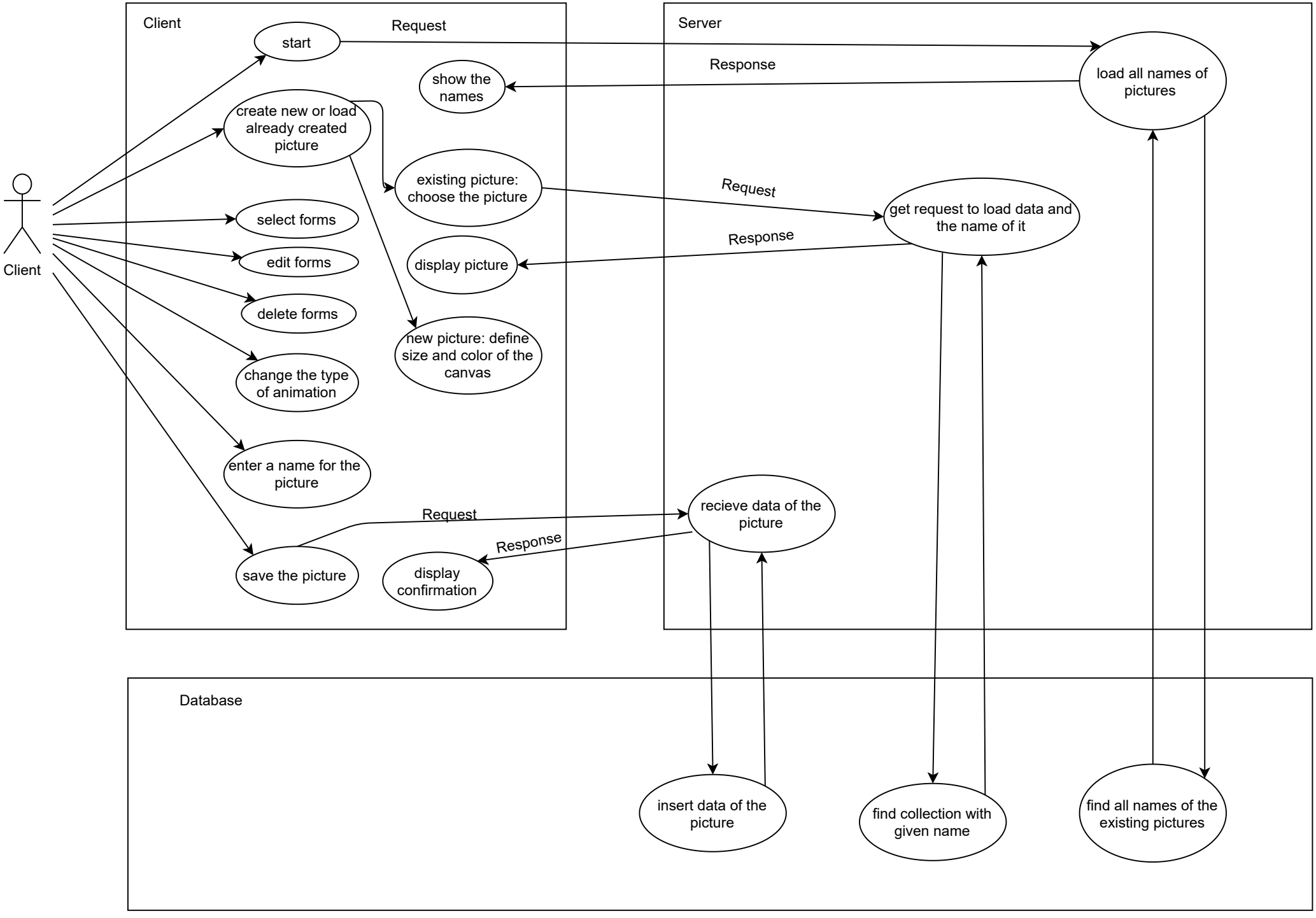
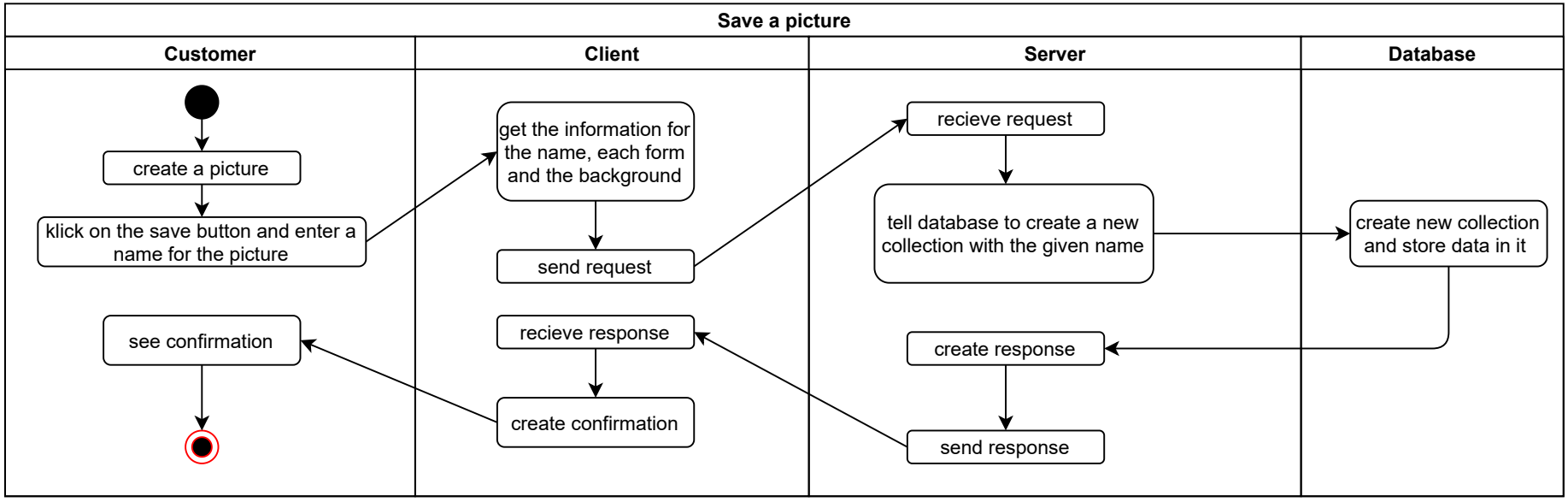
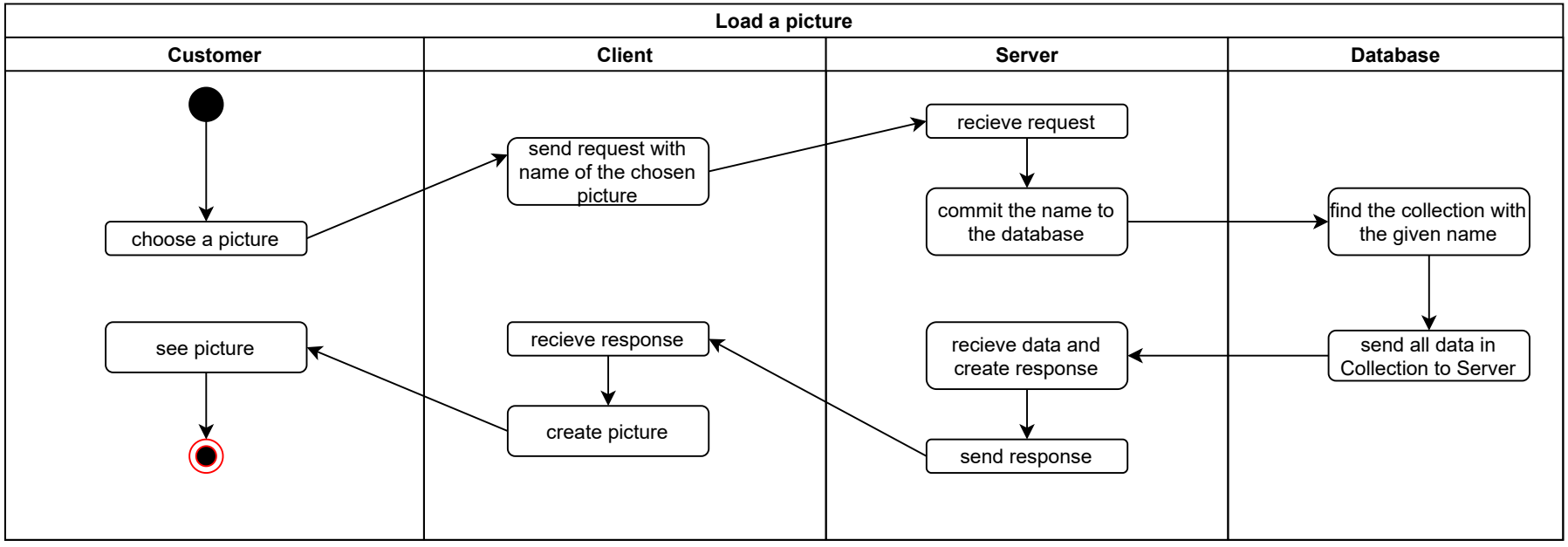
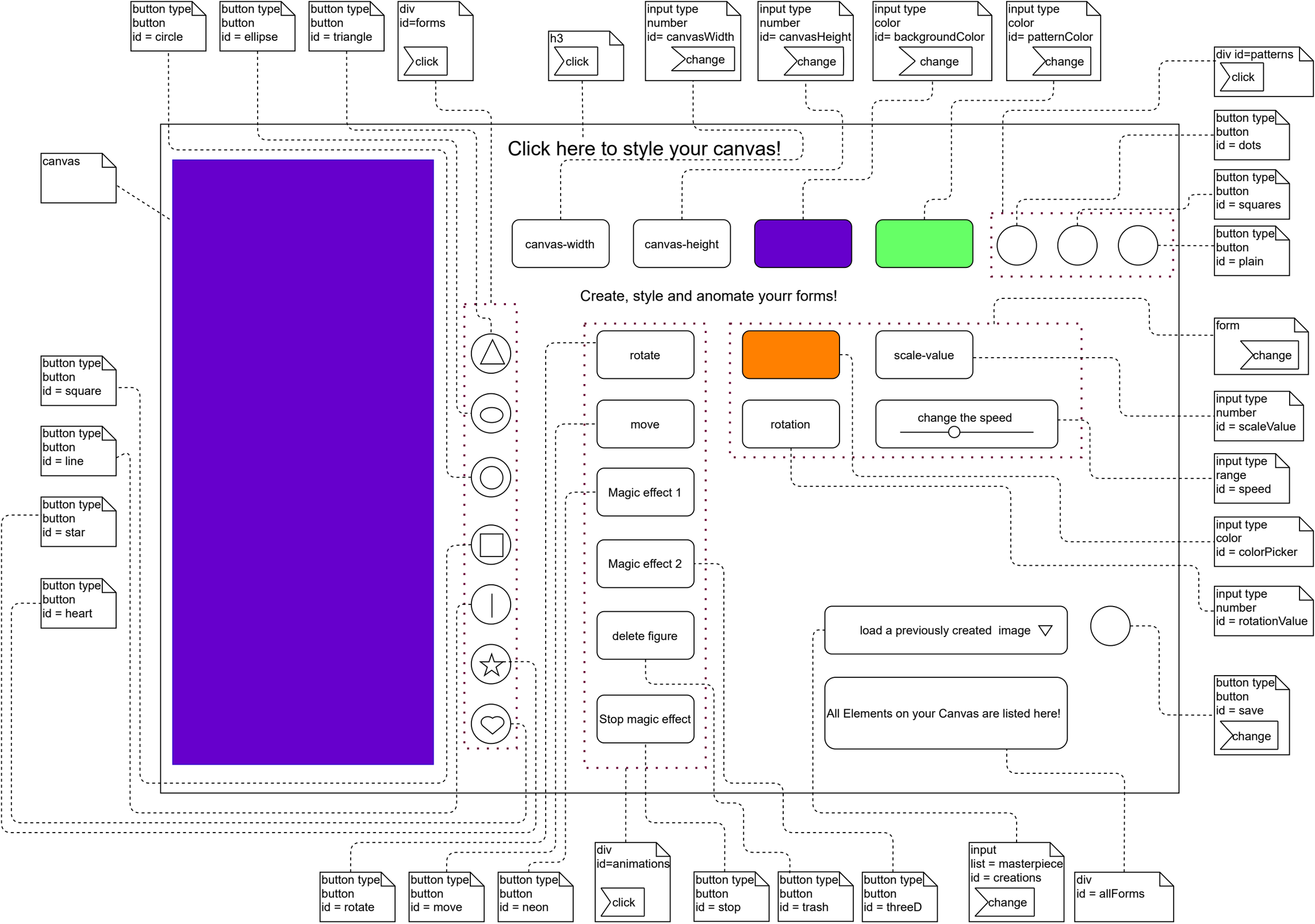
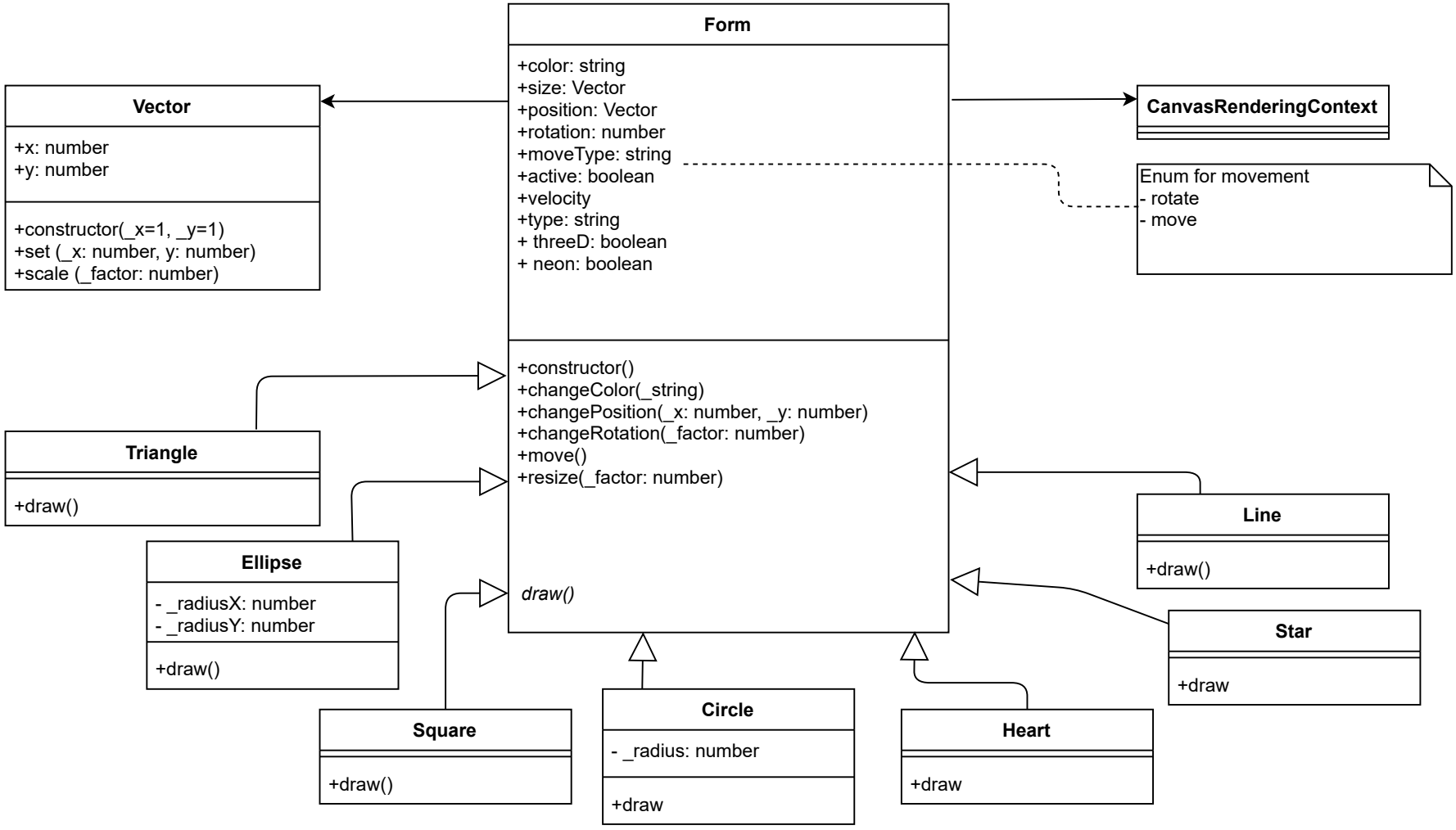

Client
Request
start
show the names
create new or load already created picture
existing picture: choose the picture
select forms
edit forms
delete forms
new picture: define size and color of the canvas
change the type of animation
enter a name for the picture
save the picture
display confirmation
Client
Server
Response
load all names of pictures
Request
get request to load data and the name of it
Response
display picture
recieve data of the picture
Request
Response
Database
insert data of the picture
find collection with given name
find all names of the existing pictures

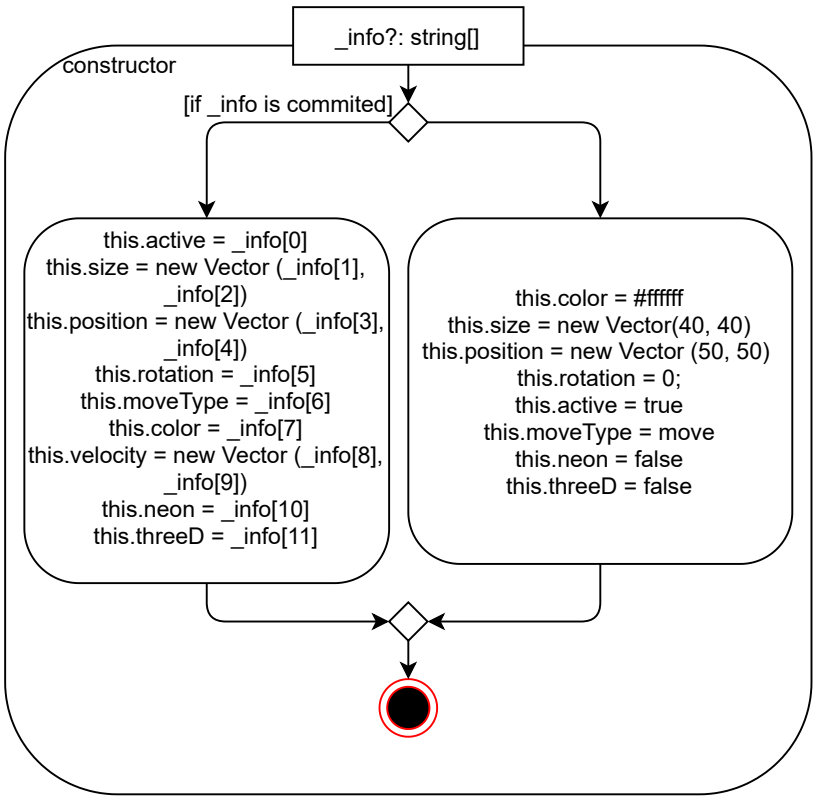

Save a picture
Customer
Client
Server
Database
create a picture
klick on the save button and enter a name for the picture
see confirmation
get the information for the name, each form and the background
send request
recieve response
create confirmation
recieve request
tell database to create a new collection with the given name
create response
send response
create new collection and store data in it

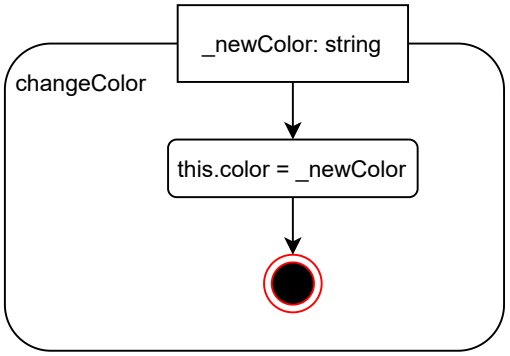

Load a picture
Customer
Client
Server
Database
choose a picture
see picture
send request with name of the chosen picture
recieve response
create picture
recieve request
commit the name to the database
recieve data and create response
send response
find the collection with the given name
send all data in Collection to Server

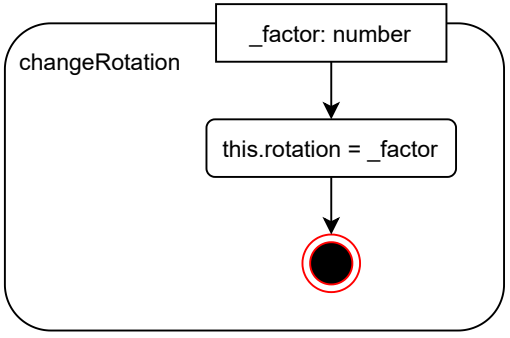

button type
button
id = circle
button type
button
id = ellipse
button type
button
id = triangle
div
id=forms
click
h3
click
input type
number
id= canvasWidth
change
input type
number
id= canvasHeight
change
input type
color
id= backgroundColor
change
input type
color
id= patternColor
change
div id=patterns
click
canvas
Click here to style your canvas!
button type
button
id = dots
button type
button
id = squares
button type
button
id = plain
canvas-width
canvas-height
Create, style and anomate yourr forms!
form
change
button type
button
id = square
button type
button
id = line
button type
button
id = star
button type
button
id = heart
rotate
move
Magic effect 1
Magic effect 2
delete figure
Stop magic effect
scale-value
rotation
change the speed
input type
number
id = scaleValue
input type
range
id = speed
input type
color
id = colorPicker
input type
number
id = rotationValue
load a previously created image
All Elements on your Canvas are listed here!
button type
button
id = save
change
button type
button
id = rotate
button type
button
id = move
button type
button
id = neon
div
id=animations
click
button type
button
id = stop
button type
button
id = trash
button type
button
id = threeD
input
list = masterpiece
id = creations
change
div
id = allForms

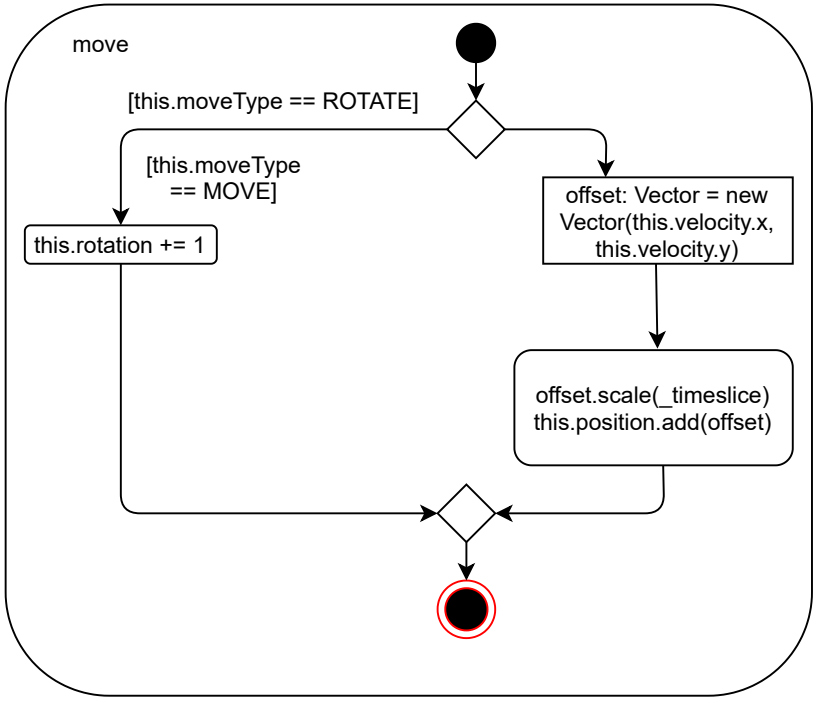

Form
+color: string
+size: Vector
+position: Vector
+rotation: number
+moveType: string
+active: boolean
+velocity
+type: string
+ threeD: boolean
+ neon: boolean
+constructor()
+changeColor(_string)
+changePosition(_x: number, _y: number)
+changeRotation(_factor: number)
+move()
+resize(_factor: number)
draw()
Vector
+x: number
+y: number
+constructor(_x=1, _y=1)
+set (_x: number, y: number)
+scale (_factor: number)
CanvasRenderingContext
Enum for movement
- rotate
- move
Triangle
+draw()
Ellipse
- _radiusX: number
- _radiusY: number
+draw()
Square
+draw()
Circle
- _radius: number
+draw
Heart
+draw
Line
+draw()
Star
+draw

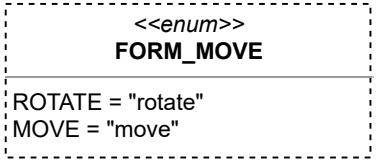
AD Form

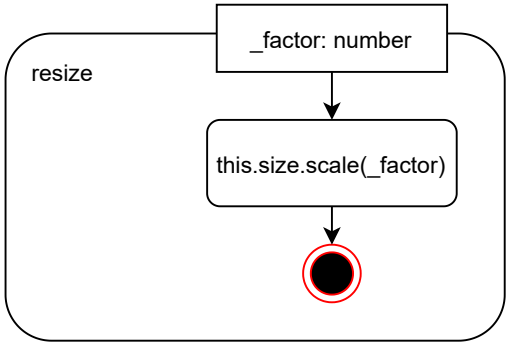

_info?: string[]
constructor
[if _info is commited]
this.active = _info[0]
this.size = new Vector (_info[1], _info[2])
this.position = new Vector (_info[3], _info[4])
this.rotation = _info[5]
this.moveType = _info[6]
this.color = _info[7]
this.velocity = new Vector (_info[8], _info[9])
this.neon = _info[10]
this.threeD = _info[11]
this.color = #ffffff
this.size = new Vector(40, 40)
this.position = new Vector (50, 50)
this.rotation = 0;
this.active = true
this.moveType = move
this.neon = false
this.threeD = false

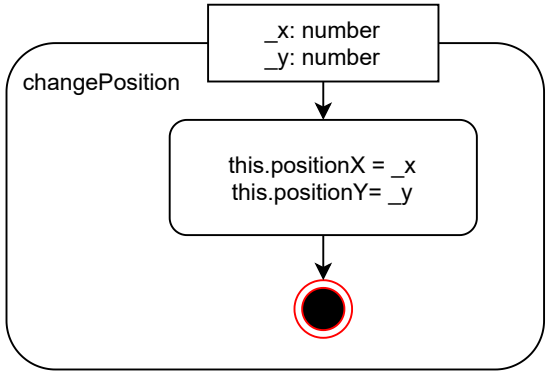

_newColor: string
changeColor
this.color = _newColor

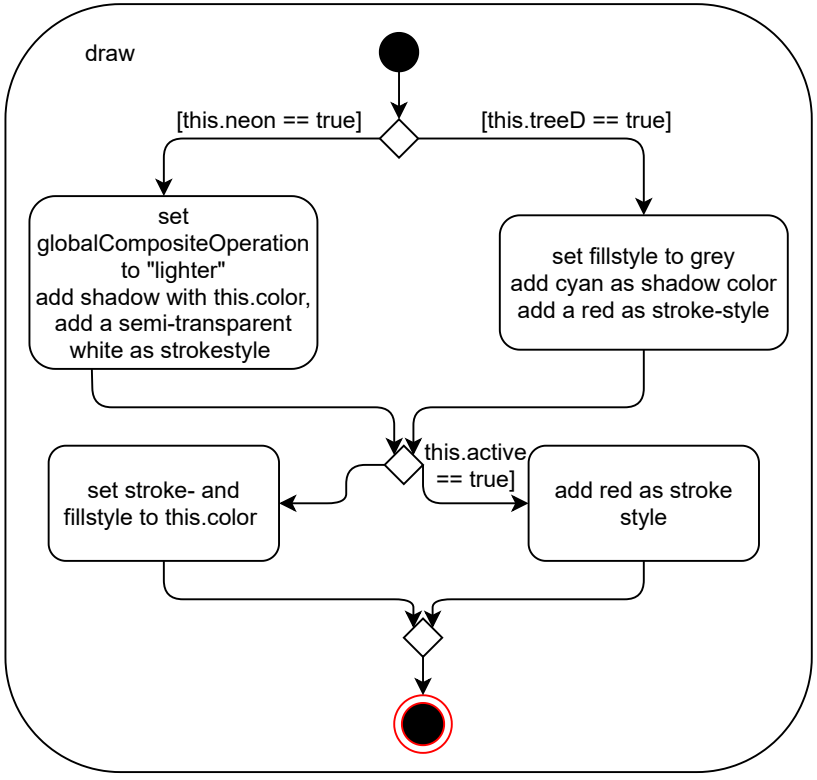

_factor: number
changeRotation
this.rotation = _factor


move
[this.moveType == ROTATE]
[this.moveType == MOVE]
this.rotation += 1
offset: Vector = new Vector(this.velocity.x, this.velocity.y)
offset.scale(_timeslice)
this.position.add(offset)

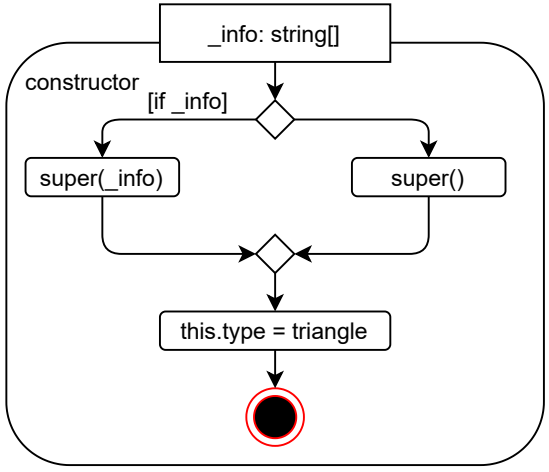

<<enum>>
FORM_MOVE
ROTATE = "rotate"
MOVE = "move"

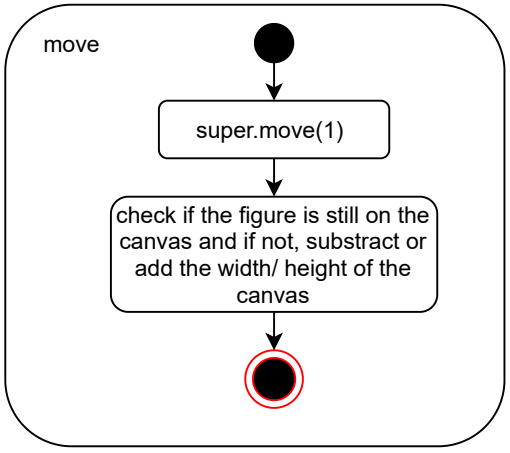

_factor: number
resize
this.size.scale(_factor)

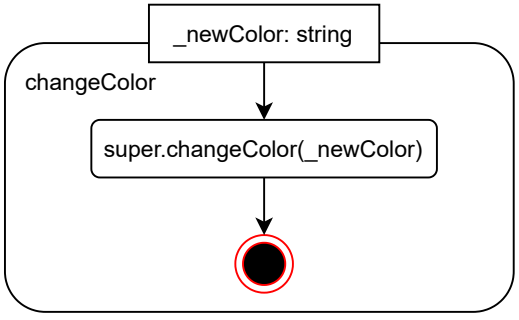

_x: number
_y: number
changePosition
this.positionX = _x
this.positionY= _y

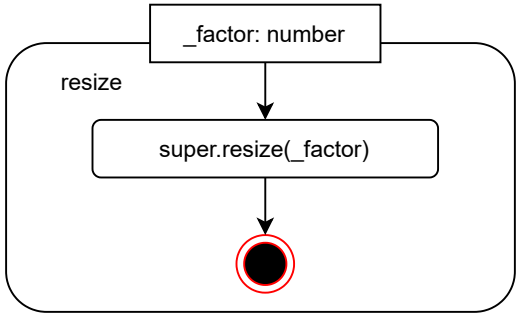

draw
[this.neon == true]
[this.treeD == true]
set globalCompositeOperation to "lighter"
add shadow with this.color,
add a semi-transparent white as strokestyle
set fillstyle to grey
add cyan as shadow color
add a red as stroke-style
this.active == true]
set stroke- and fillstyle to this.color
add red as stroke style

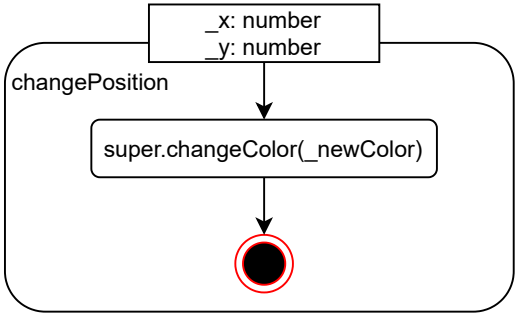
AD Triangle

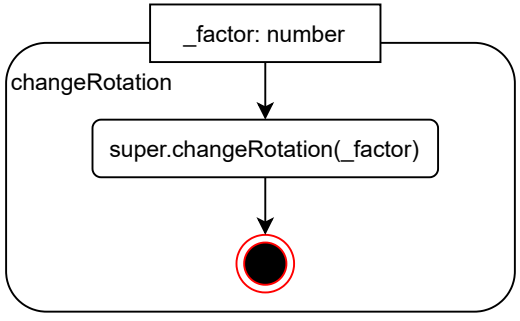

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = triangle

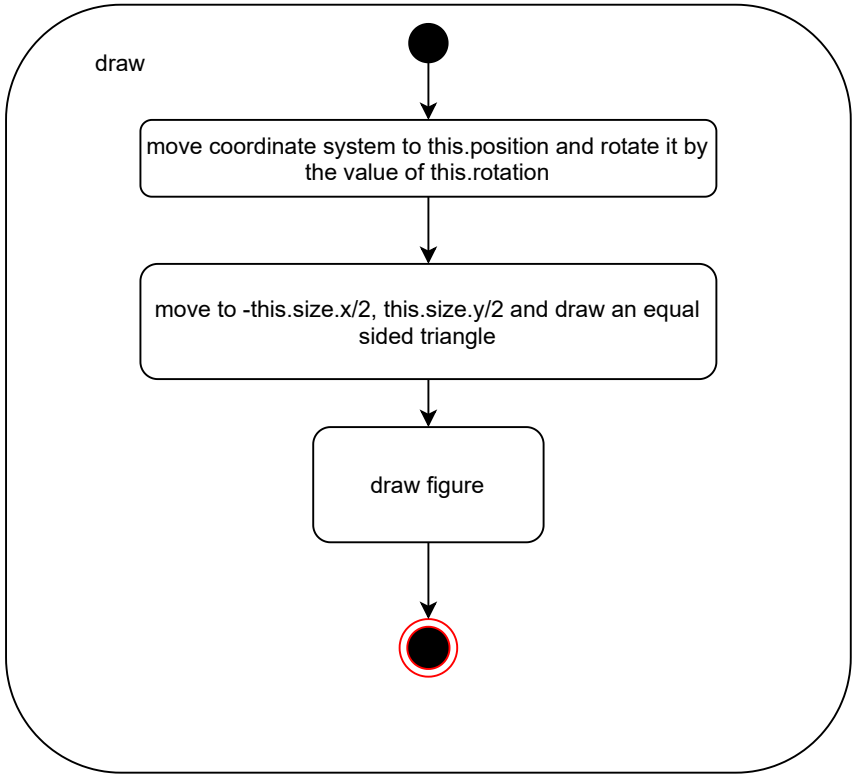

move
super.move(1)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


_newColor: string
changeColor
super.changeColor(_newColor)

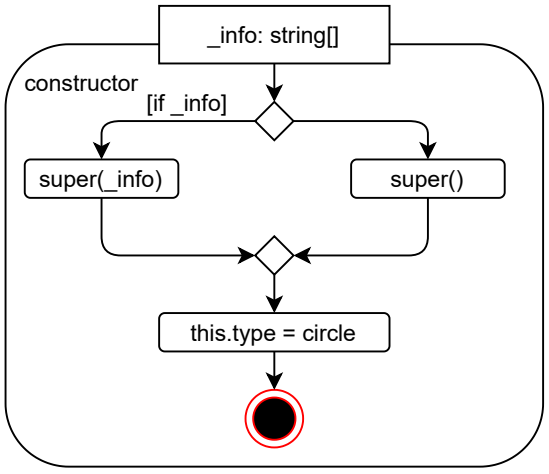

_factor: number
resize
super.resize(_factor)

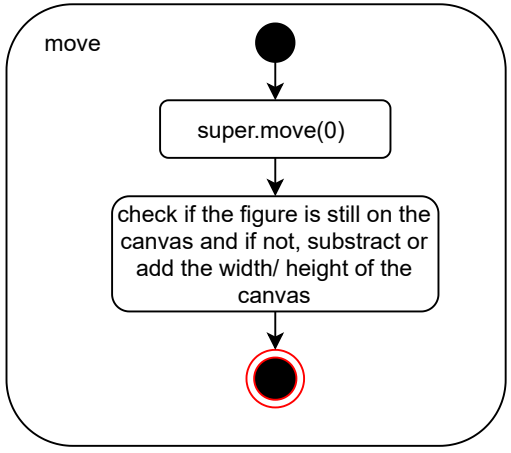

_x: number
_y: number
changePosition
super.changeColor(_newColor)

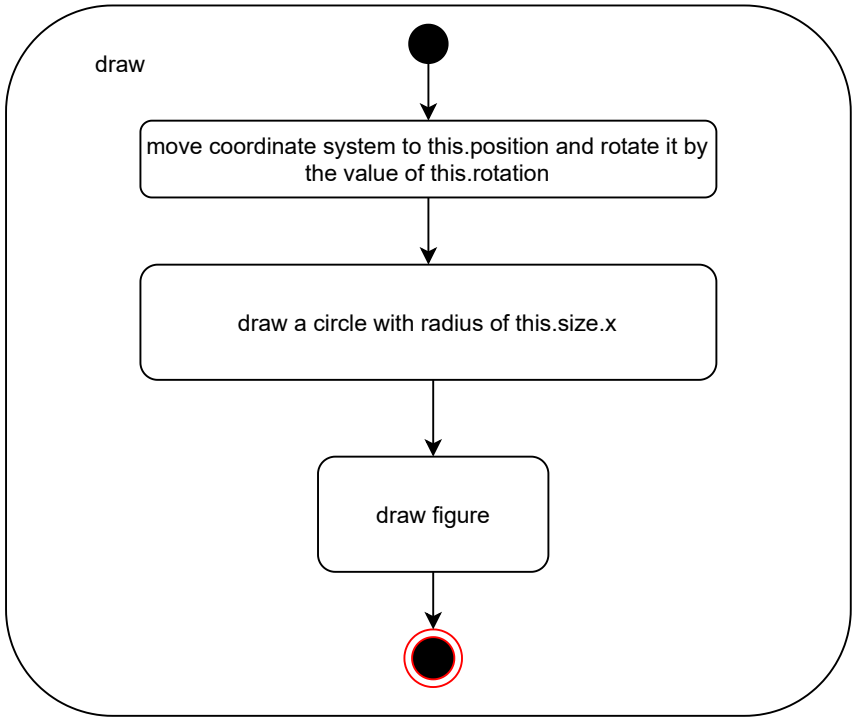

_factor: number
changeRotation
super.changeRotation(_factor)

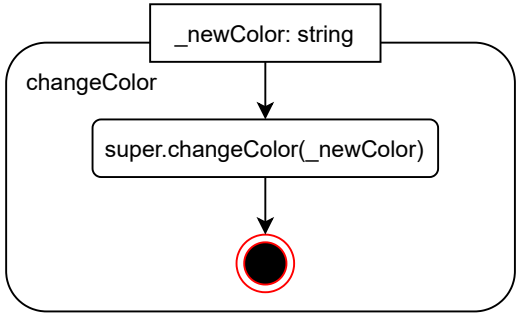

draw
move coordinate system to this.position and rotate it by the value of this.rotation
move to -this.size.x/2, this.size.y/2 and draw an equal sided triangle
draw figure

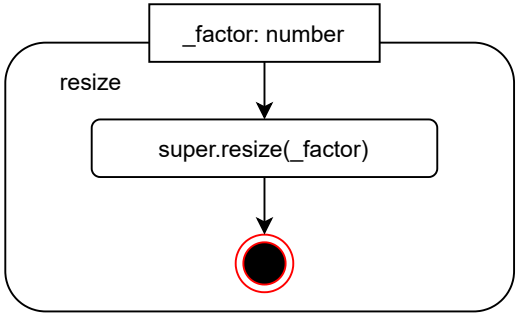
AD Circle

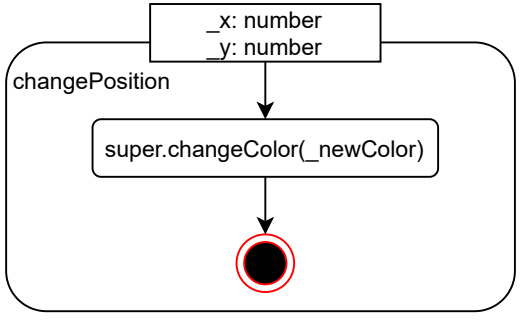

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = circle

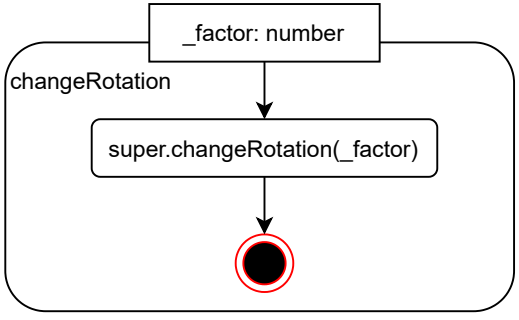

move
super.move(0)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a circle with radius of this.size.x
draw figure

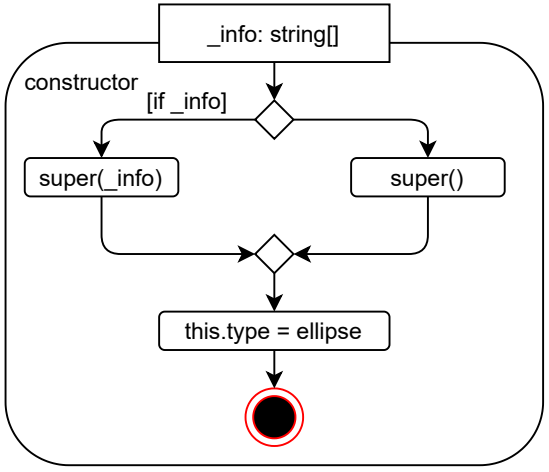

_newColor: string
changeColor
super.changeColor(_newColor)

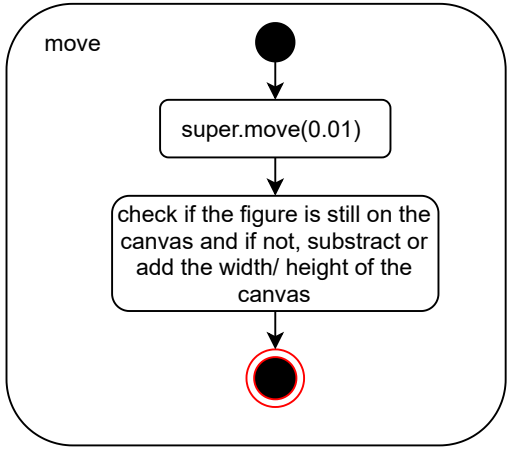

_factor: number
resize
super.resize(_factor)

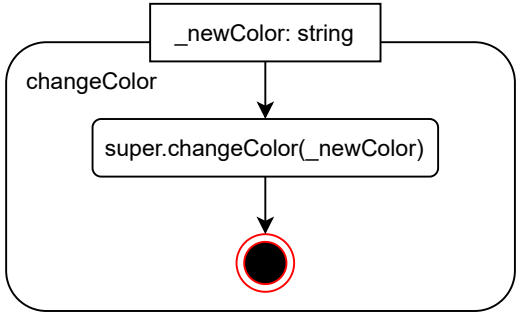

_x: number
_y: number
changePosition
super.changeColor(_newColor)

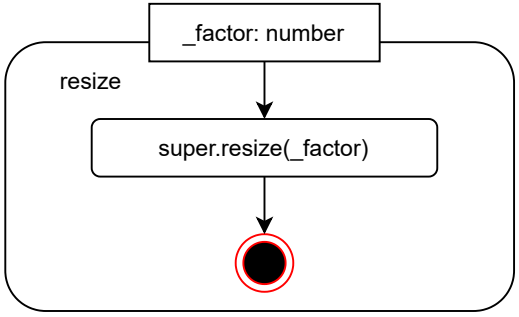

_factor: number
changeRotation
super.changeRotation(_factor)

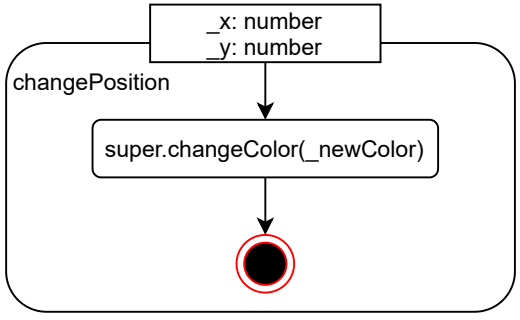
AD Ellipse

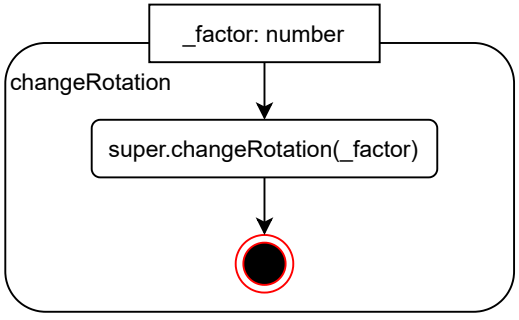

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = ellipse

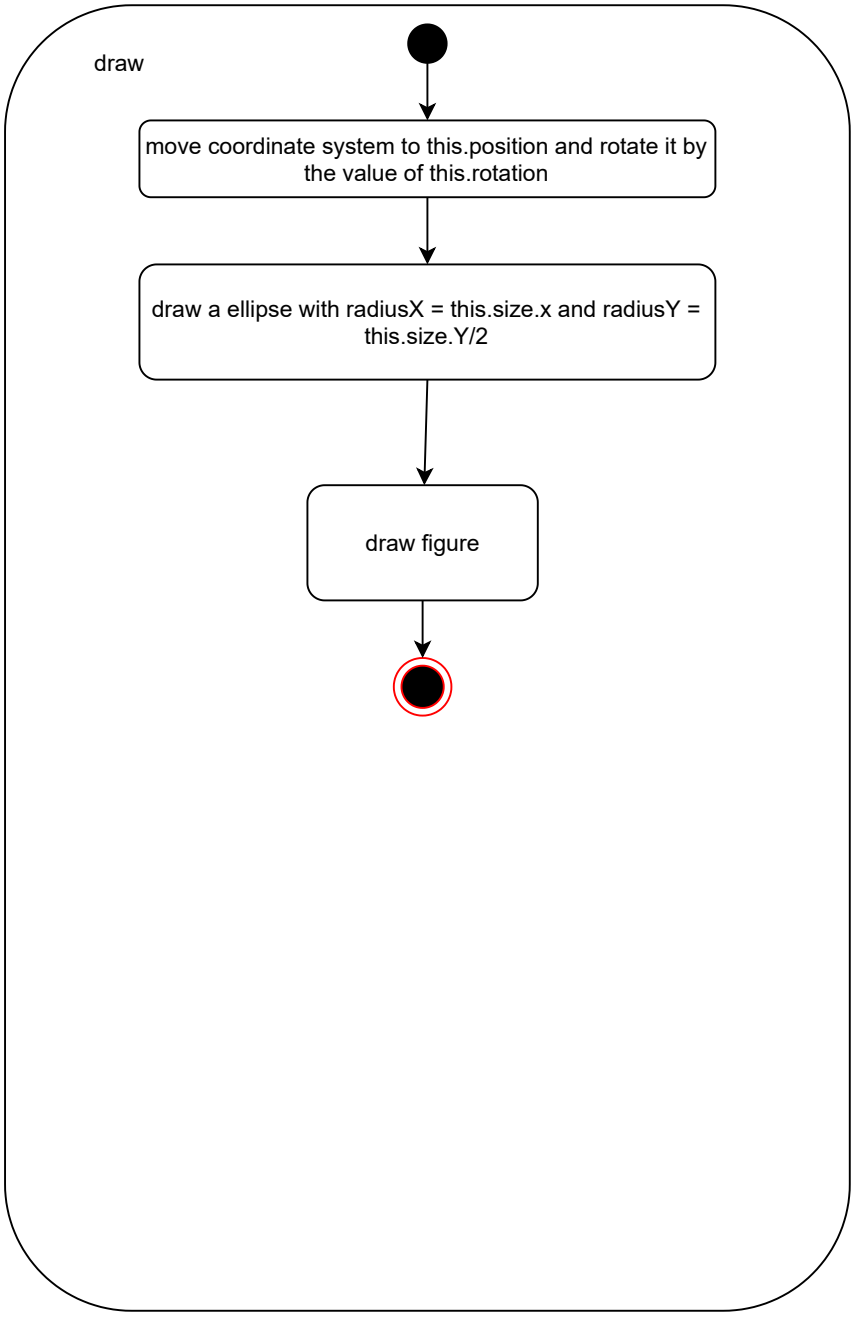

move
super.move(0.01)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


_newColor: string
changeColor
super.changeColor(_newColor)

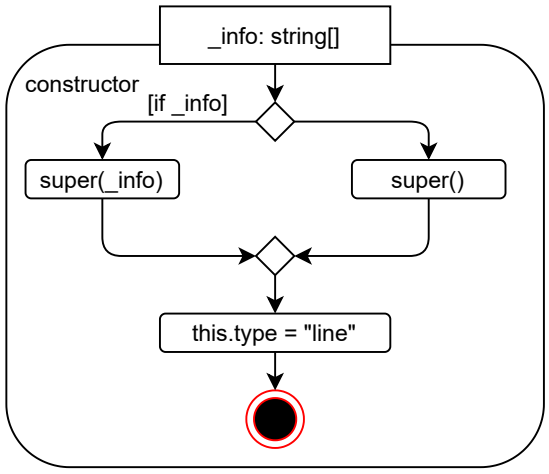

_factor: number
resize
super.resize(_factor)

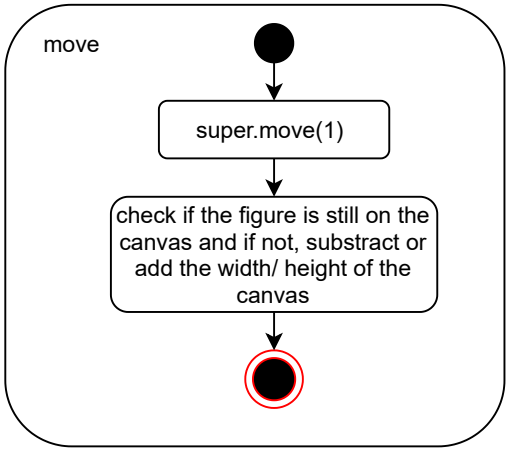

_x: number
_y: number
changePosition
super.changeColor(_newColor)

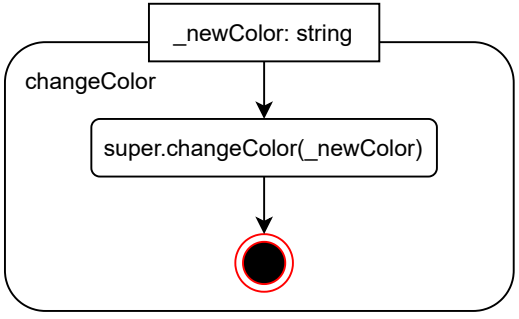

_factor: number
changeRotation
super.changeRotation(_factor)

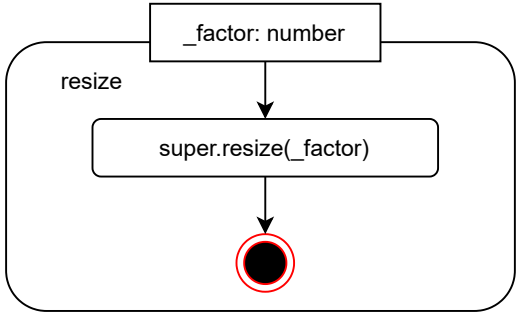

draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a ellipse with radiusX = this.size.x and radiusY = this.size.Y/2
draw figure

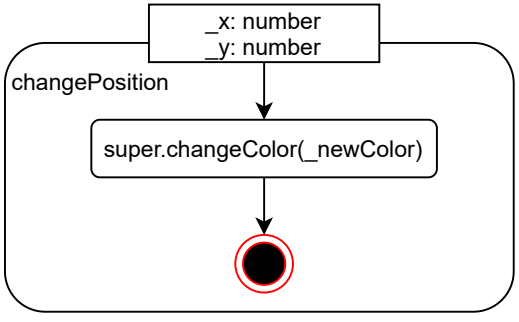
AD Line

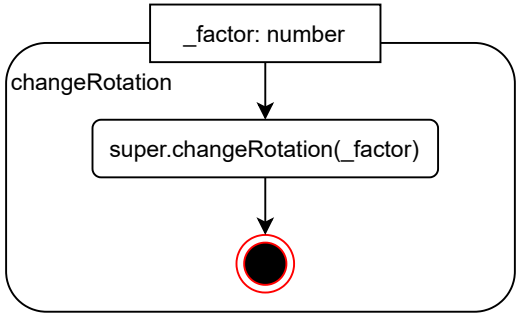

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = "line"

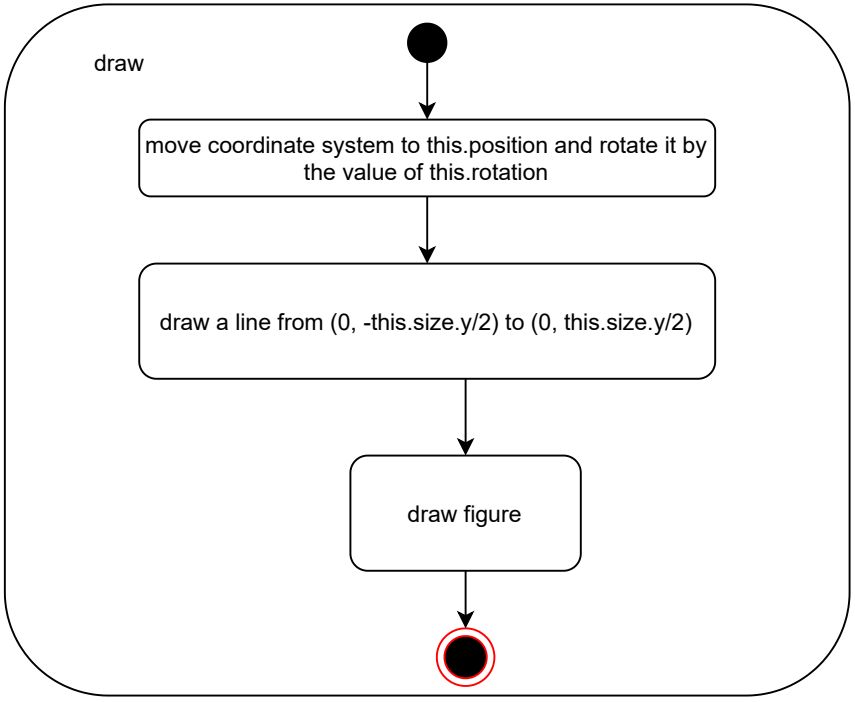

move
super.move(1)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


_newColor: string
changeColor
super.changeColor(_newColor)

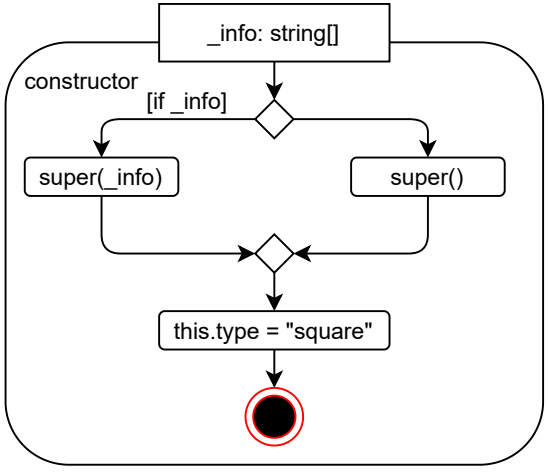

_factor: number
resize
super.resize(_factor)

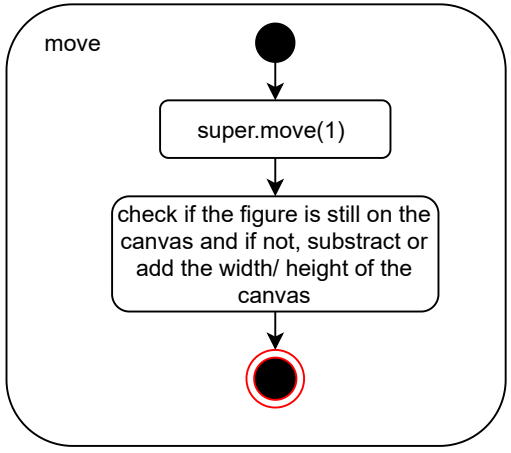

_x: number
_y: number
changePosition
super.changeColor(_newColor)

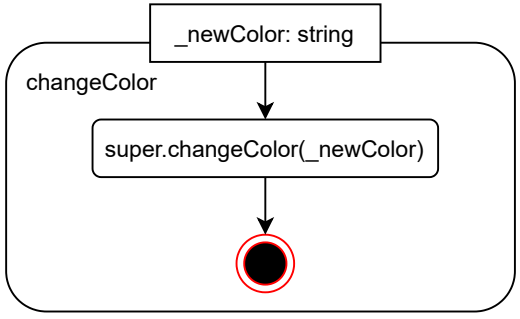

_factor: number
changeRotation
super.changeRotation(_factor)

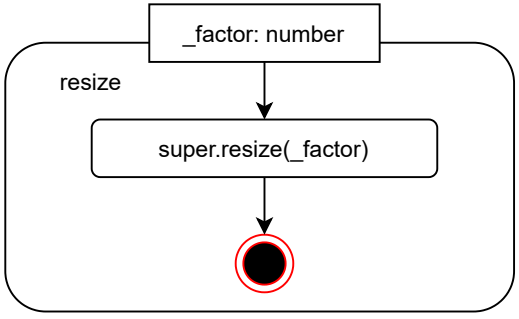

draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a line from (0, -this.size.y/2) to (0, this.size.y/2)
draw figure

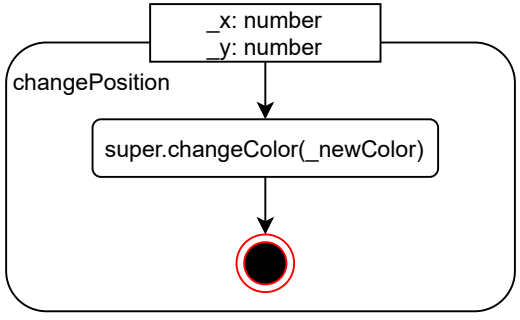
AD Square

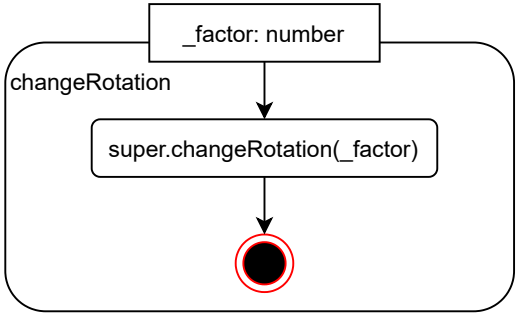

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = "square"

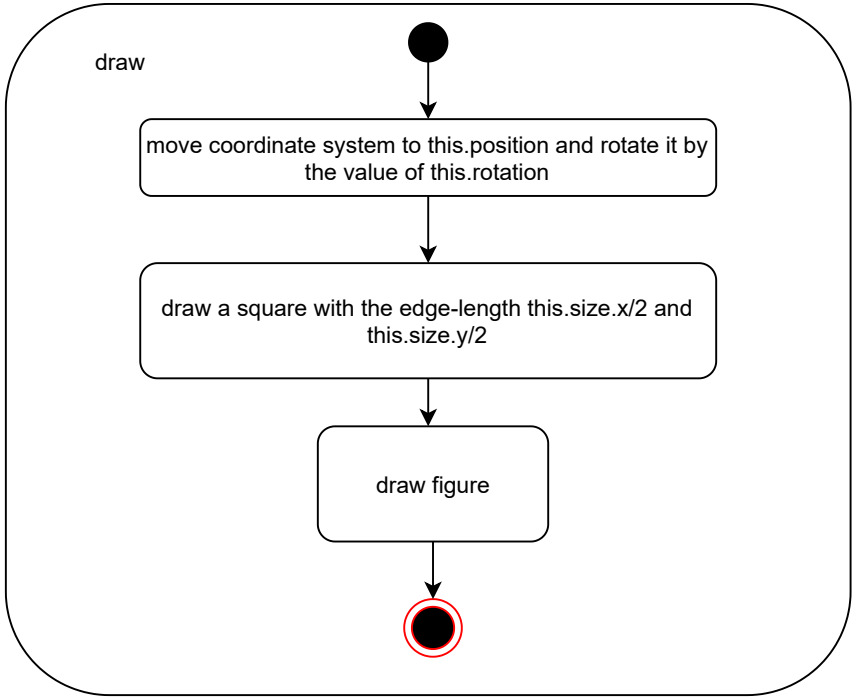

move
super.move(1)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


_newColor: string
changeColor
super.changeColor(_newColor)

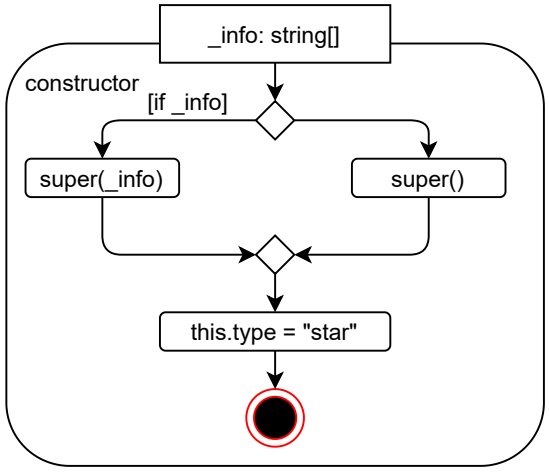

_factor: number
resize
super.resize(_factor)

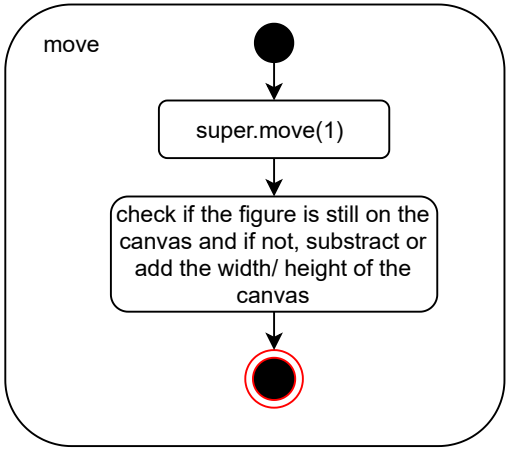

_x: number
_y: number
changePosition
super.changeColor(_newColor)

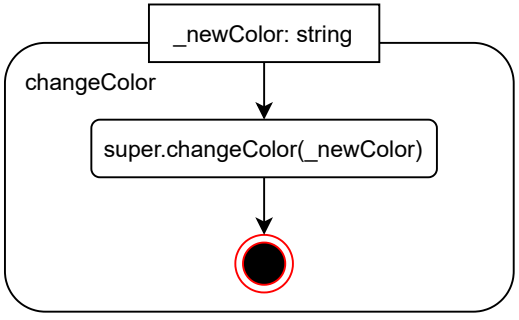

_factor: number
changeRotation
super.changeRotation(_factor)

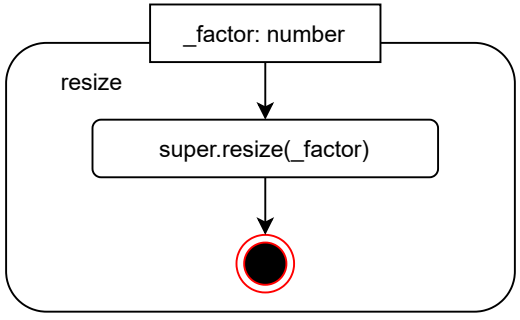

draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a square with the edge-length this.size.x/2 and this.size.y/2
draw figure

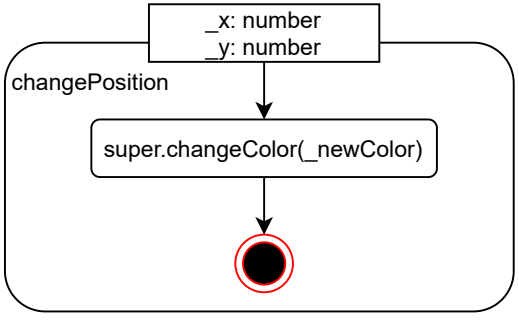
AD Star

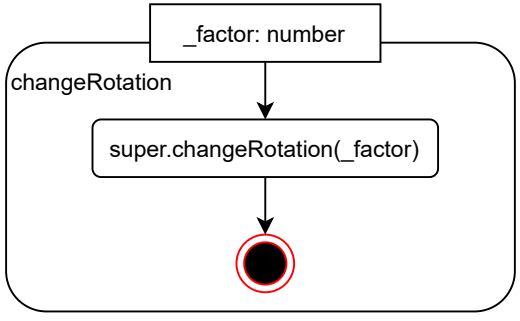

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = "star"

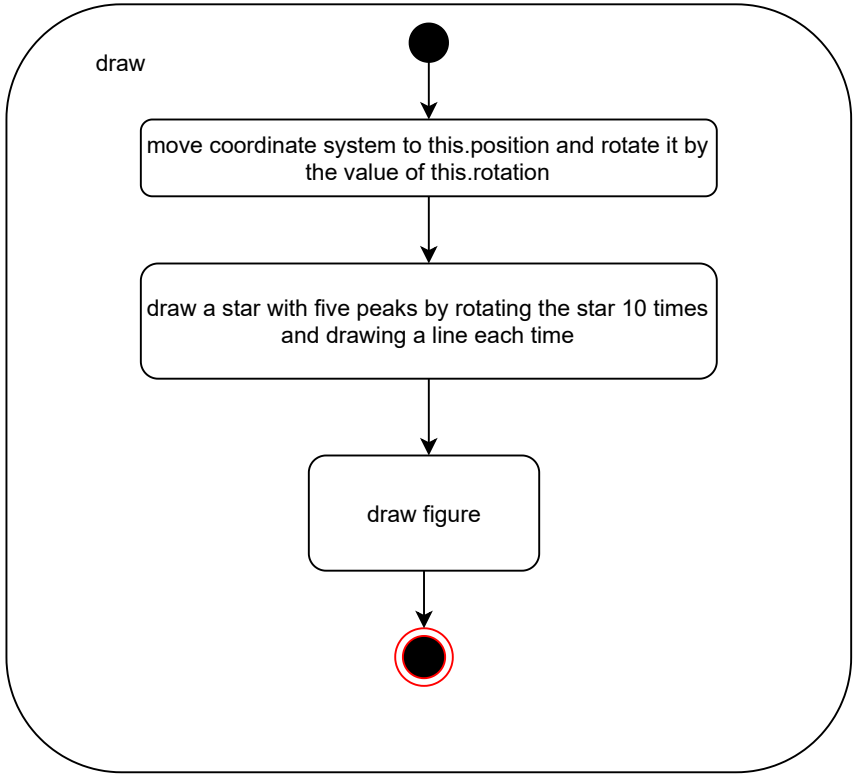

move
super.move(1)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


_newColor: string
changeColor
super.changeColor(_newColor)

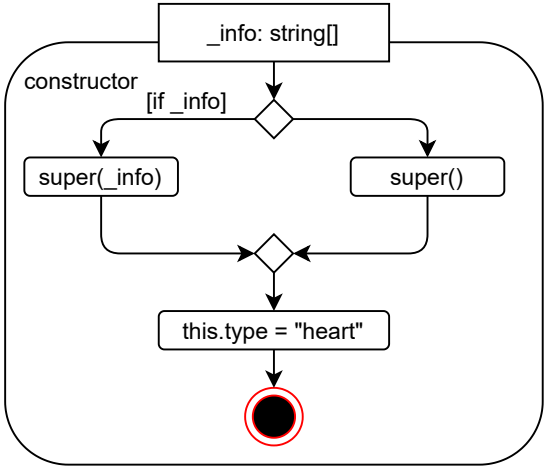

_factor: number
resize
super.resize(_factor)

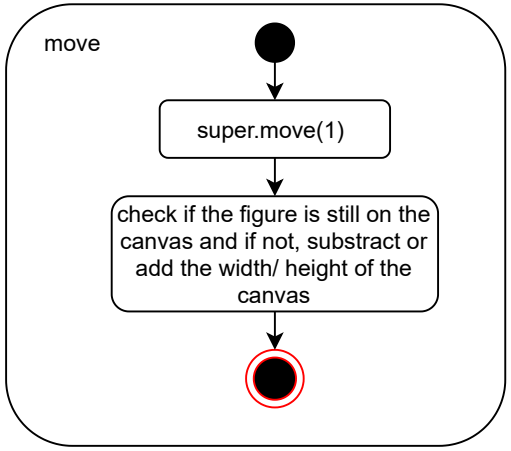

_x: number
_y: number
changePosition
super.changeColor(_newColor)

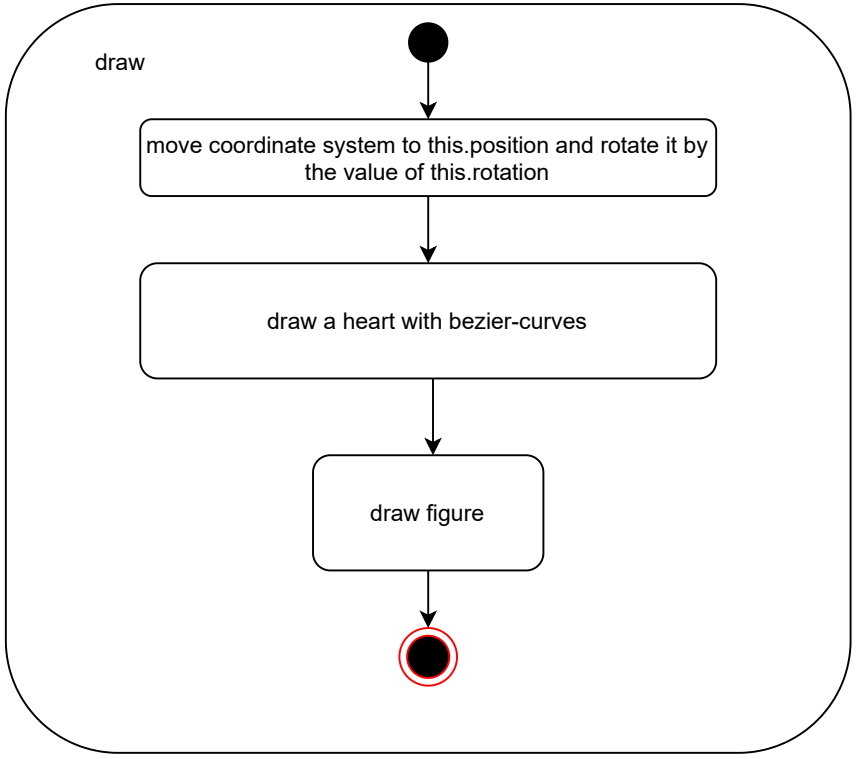

_factor: number
changeRotation
super.changeRotation(_factor)

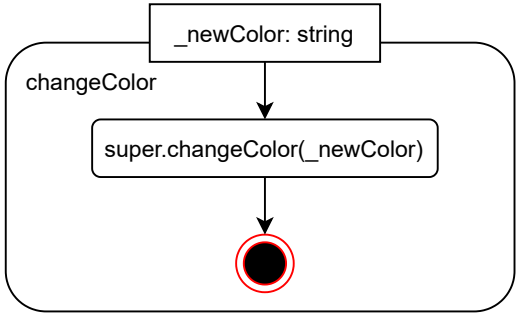

draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a star with five peaks by rotating the star 10 times and drawing a line each time
draw figure

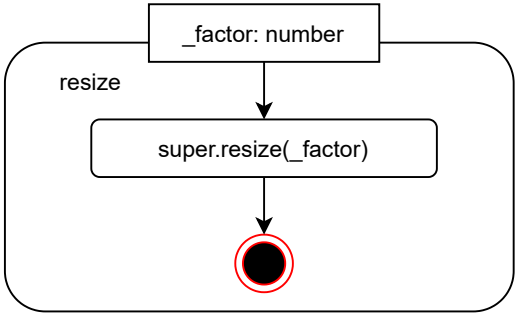
AD Heart

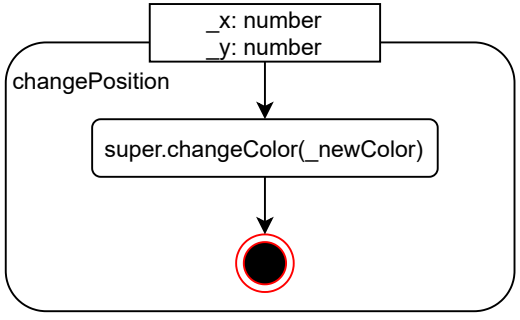

_info: string[]
constructor
[if _info]
super(_info)
super()
this.type = "heart"

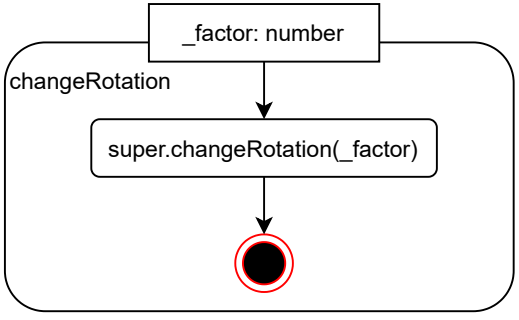

move
super.move(1)
check if the figure is still on the canvas and if not, substract or add the width/ height of the canvas


draw
move coordinate system to this.position and rotate it by the value of this.rotation
draw a heart with bezier-curves
draw figure

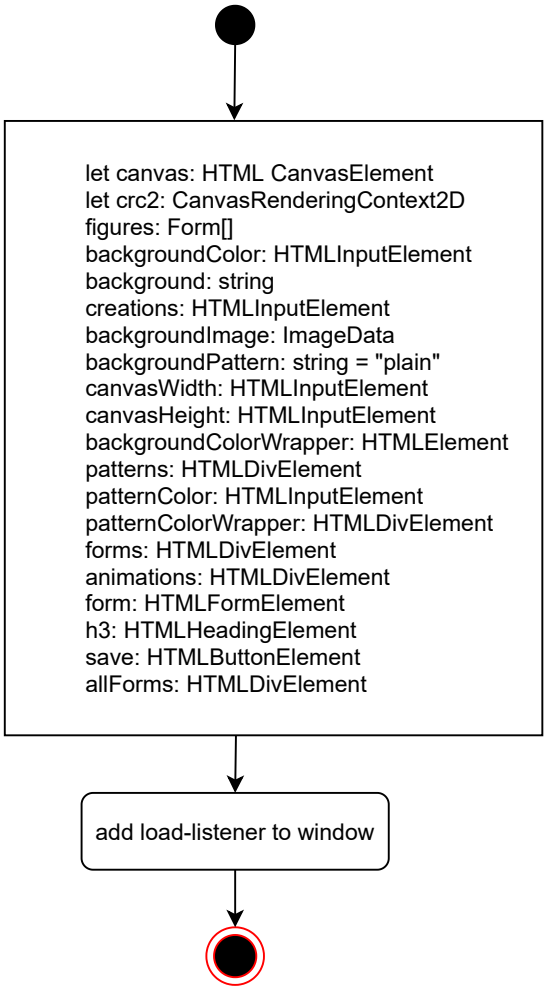

_newColor: string
changeColor
super.changeColor(_newColor)

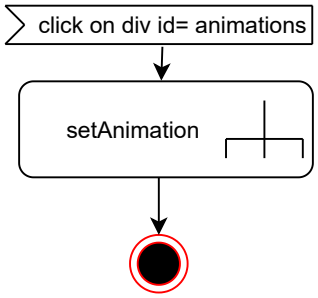

_factor: number
resize
super.resize(_factor)

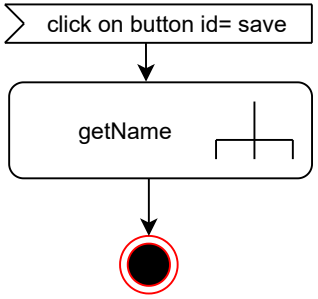

_x: number
_y: number
changePosition
super.changeColor(_newColor)

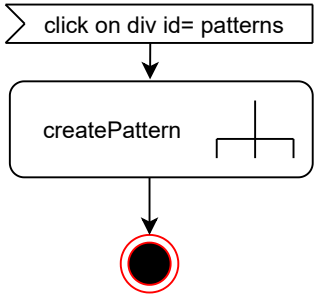

_factor: number
changeRotation
super.changeRotation(_factor)

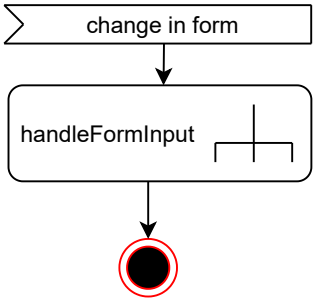
AD Main

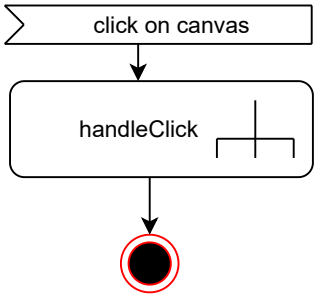

let canvas: HTML CanvasElement
let crc2: CanvasRenderingContext2D
figures: Form[]
backgroundColor: HTMLInputElement
background: string
creations: HTMLInputElement
backgroundImage: ImageData
backgroundPattern: string = "plain"
canvasWidth: HTMLInputElement
canvasHeight: HTMLInputElement
backgroundColorWrapper: HTMLElement
patterns: HTMLDivElement
patternColor: HTMLInputElement
patternColorWrapper: HTMLDivElement
forms: HTMLDivElement
animations: HTMLDivElement
form: HTMLFormElement
h3: HTMLHeadingElement
save: HTMLButtonElement
allForms: HTMLDivElement
add load-listener to window

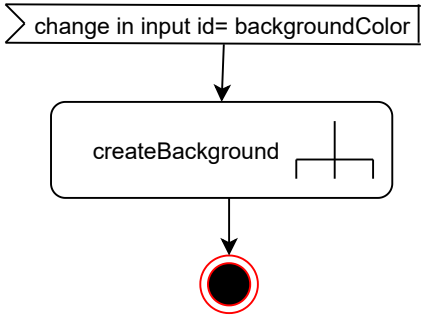

click on div id= animations
setAnimation

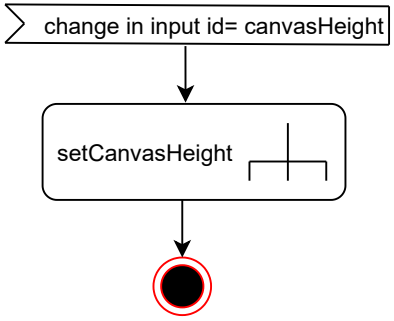

click on button id= save
getName

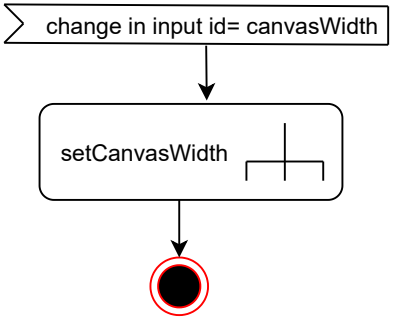

click on div id= patterns
createPattern

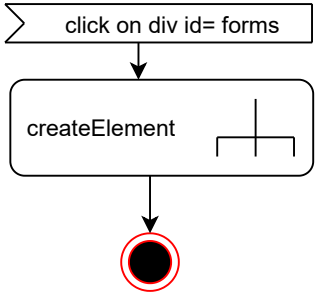

change in form
handleFormInput

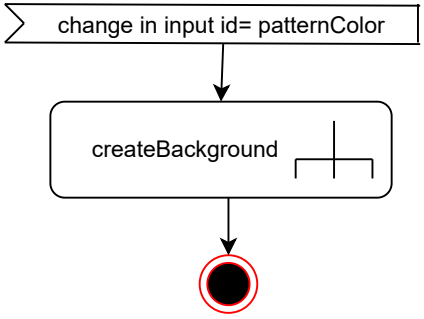

click on canvas
handleClick

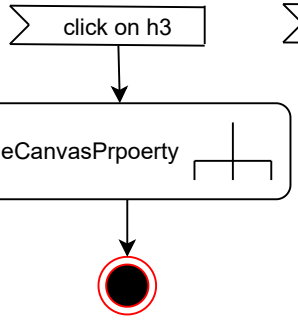

change in input id= backgroundColor
createBackground

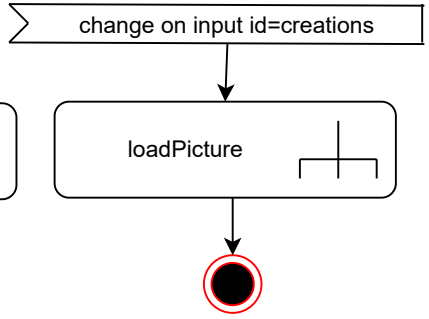

change in input id= canvasHeight
setCanvasHeight

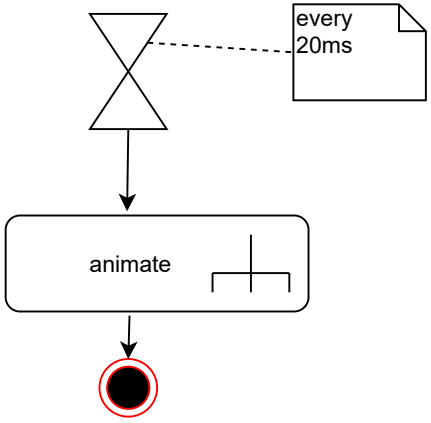

change in input id= canvasWidth
setCanvasWidth

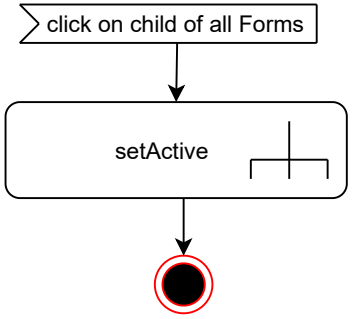

click on div id= forms
createElement

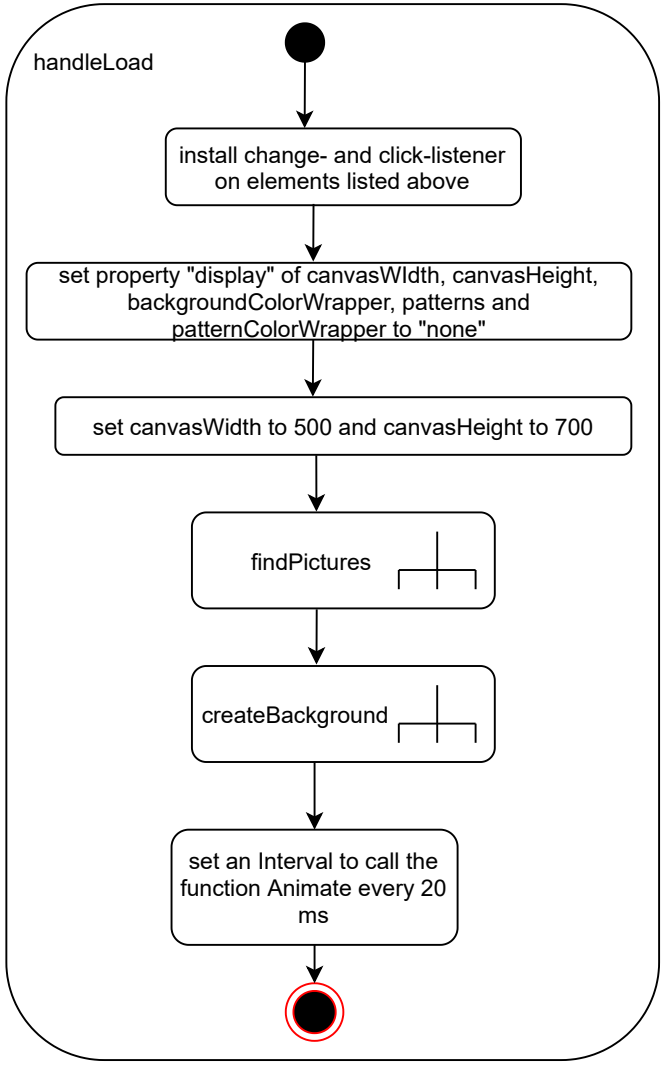

change in input id= patternColor
createBackground

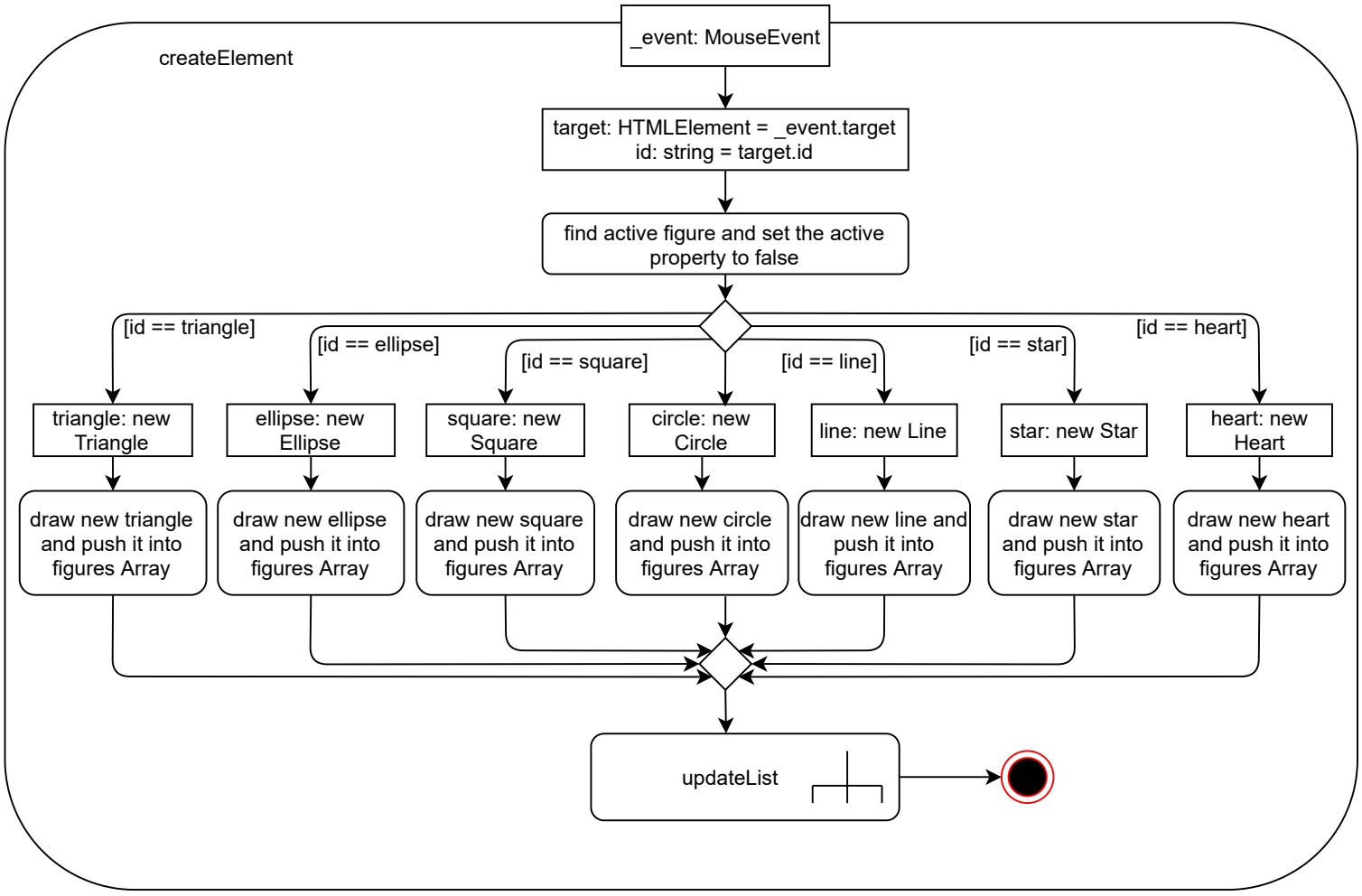

click on h3
toggleCanvasPrpoerty

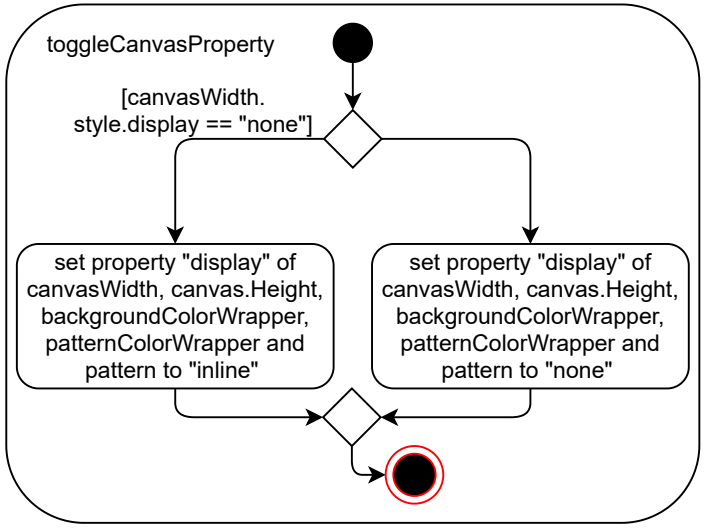

change on input id=creations
loadPicture

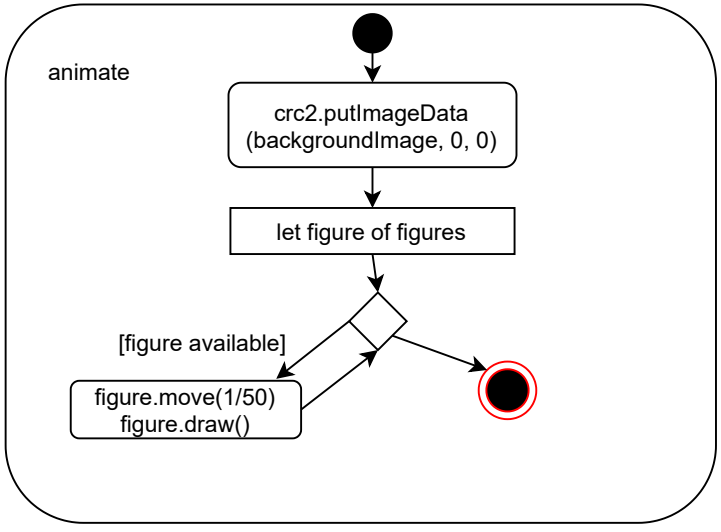

every
20ms
animate

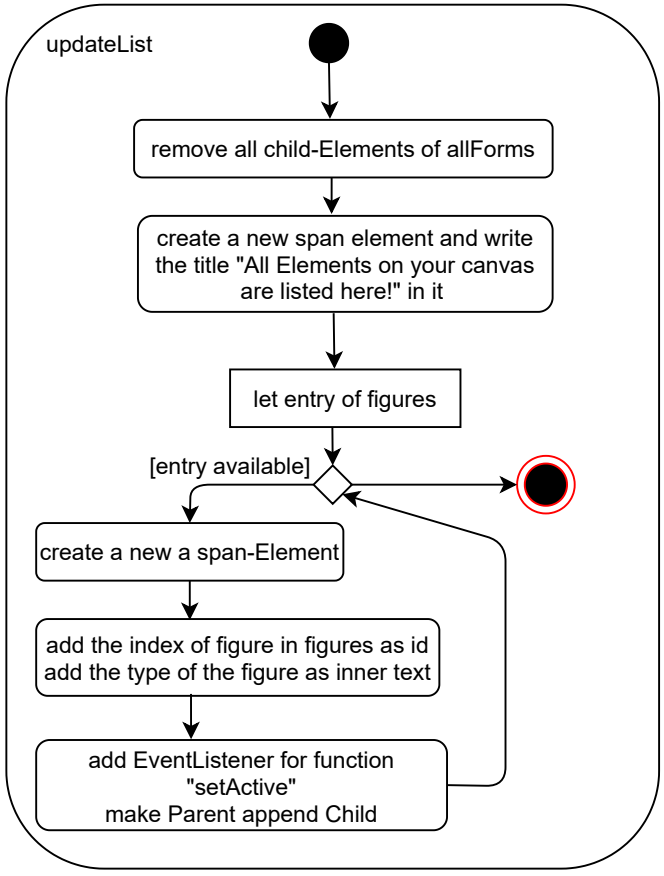

click on child of all Forms
setActive

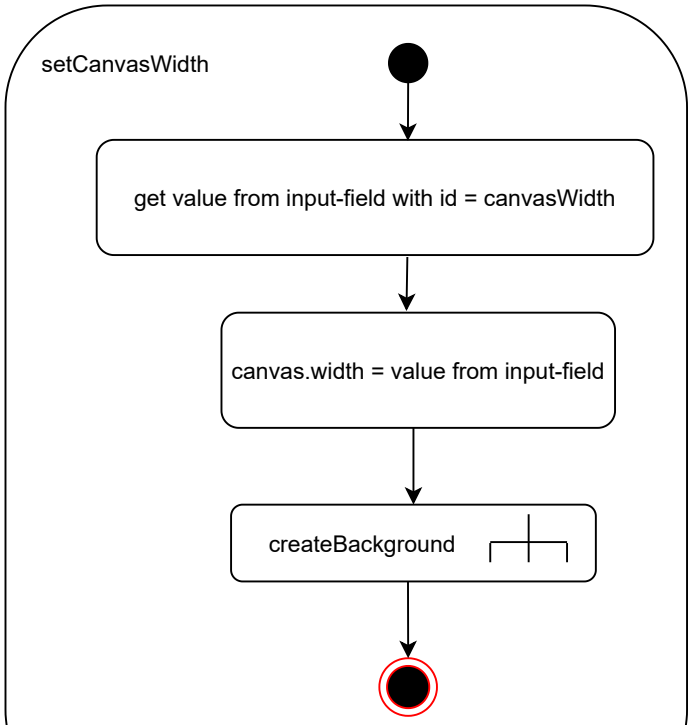

handleLoad
install change- and click-listener on elements listed above
set property "display" of canvasWIdth, canvasHeight, backgroundColorWrapper, patterns and patternColorWrapper to "none"
set canvasWidth to 500 and canvasHeight to 700
findPictures
createBackground
set an Interval to call the function Animate every 20 ms

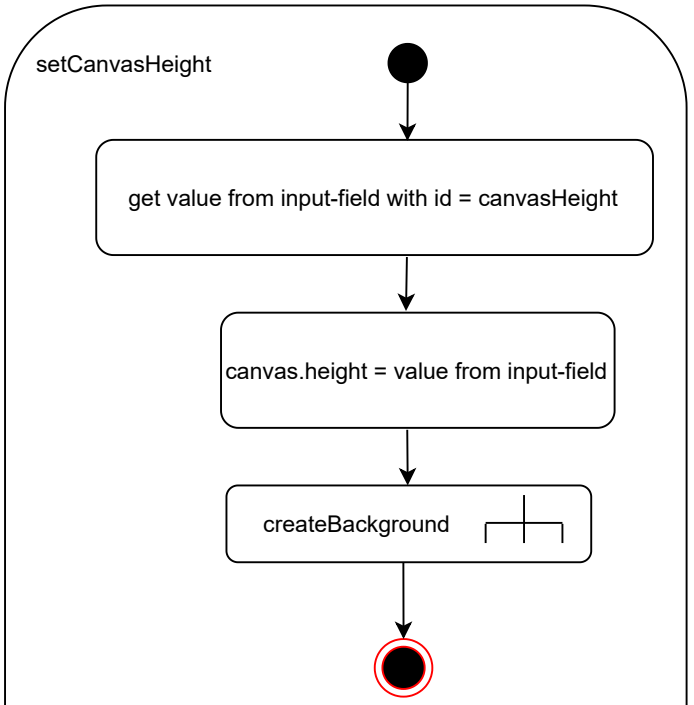

createElement
_event: MouseEvent
target: HTMLElement = _event.target
id: string = target.id
find active figure and set the active property to false
[id == triangle]
[id == ellipse]
[id == square]
[id == line]
[id == star]
[id == heart]
triangle: new Triangle
ellipse: new Ellipse
square: new Square
circle: new Circle
line: new Line
star: new Star
heart: new Heart
draw new triangle and push it into figures Array
draw new ellipse and push it into figures Array
draw new square and push it into figures Array
draw new circle and push it into figures Array
draw new line and push it into figures Array
draw new star and push it into figures Array
draw new heart and push it into figures Array
updateList

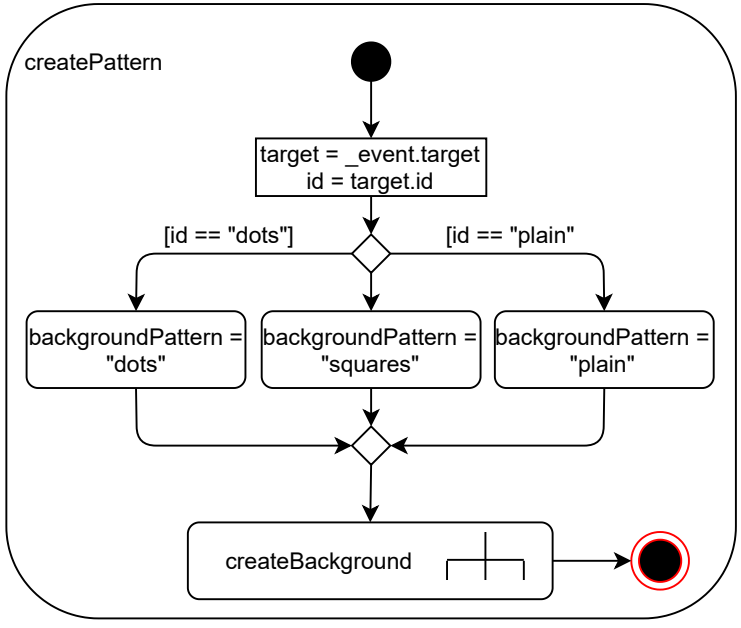

toggleCanvasProperty
[canvasWidth.
style.display == "none"]
set property "display" of canvasWidth, canvas.Height, backgroundColorWrapper, patternColorWrapper and pattern to "inline"
set property "display" of canvasWidth, canvas.Height, backgroundColorWrapper, patternColorWrapper and pattern to "none"

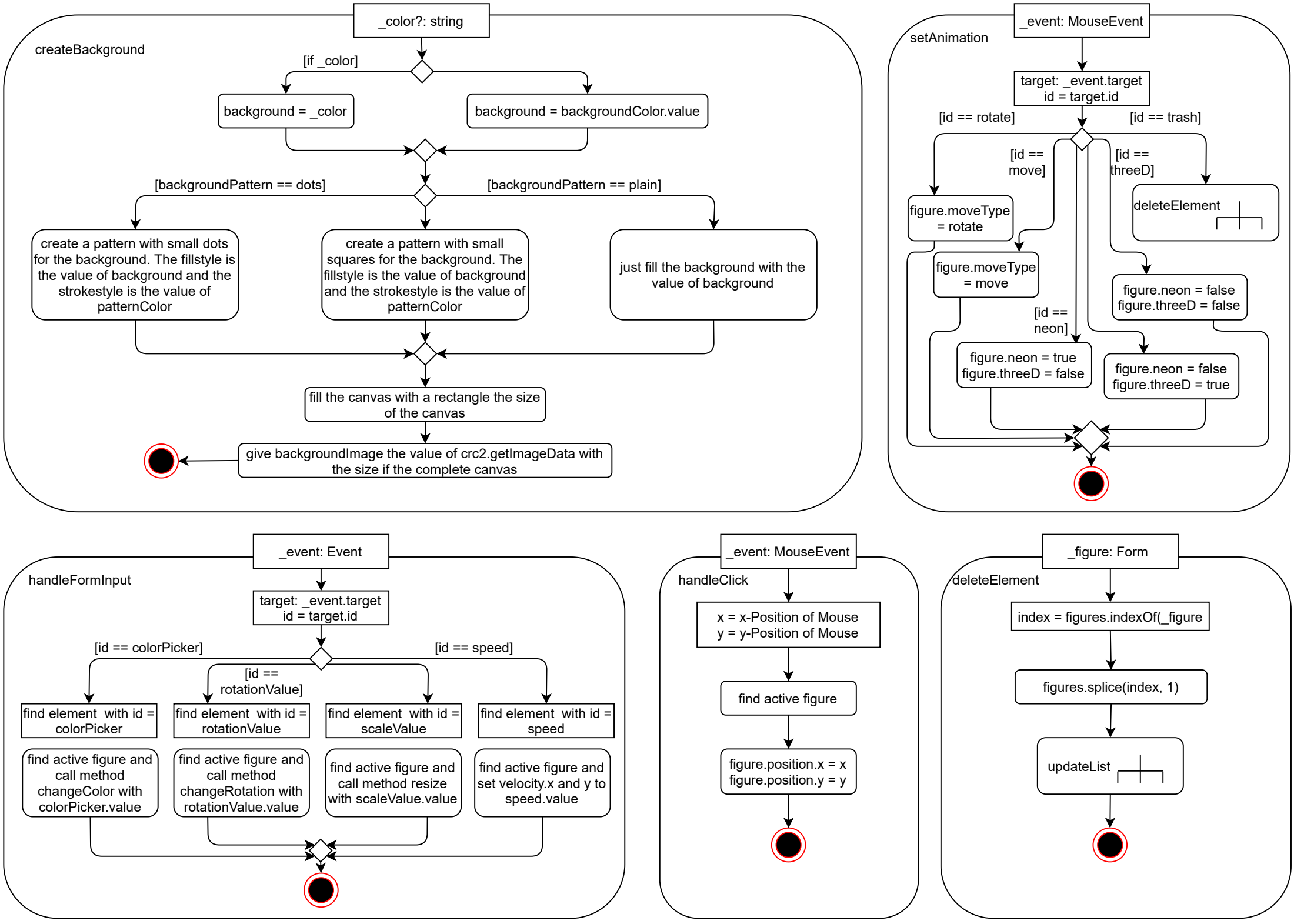

animate
crc2.putImageData (backgroundImage, 0, 0)
let figure of figures
[figure available]
figure.move(1/50)
figure.draw()

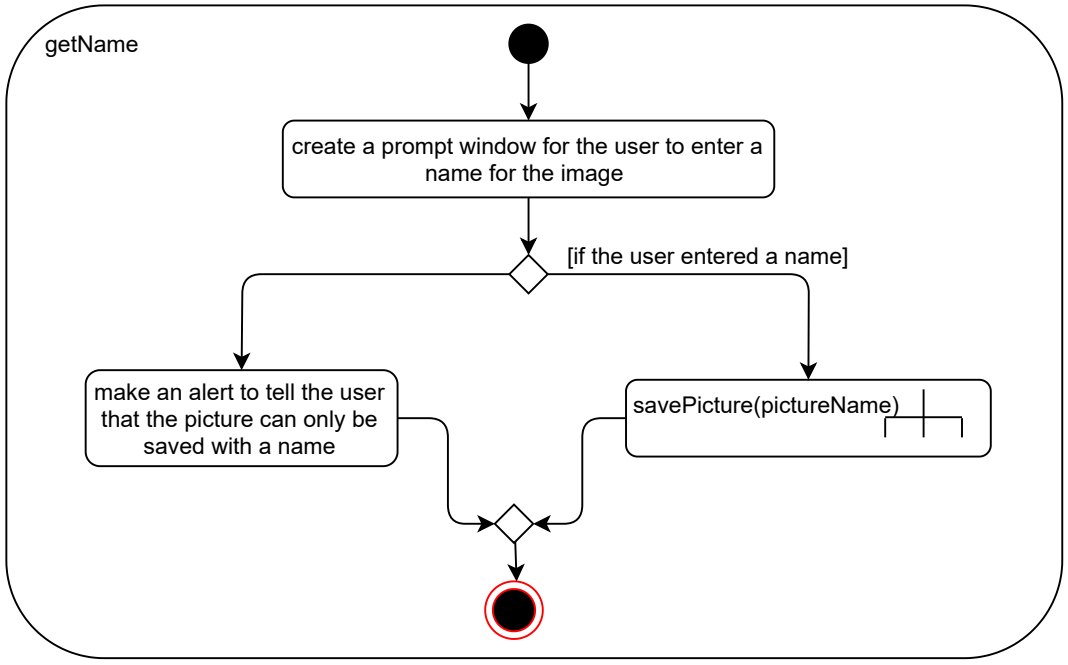

updateList
remove all child-Elements of allForms
create a new span element and write the title "All Elements on your canvas are listed here!" in it
let entry of figures
[entry available]
create a new a span-Element
add the index of figure in figures as id
add the type of the figure as inner text
add EventListener for function "setActive"
make Parent append Child

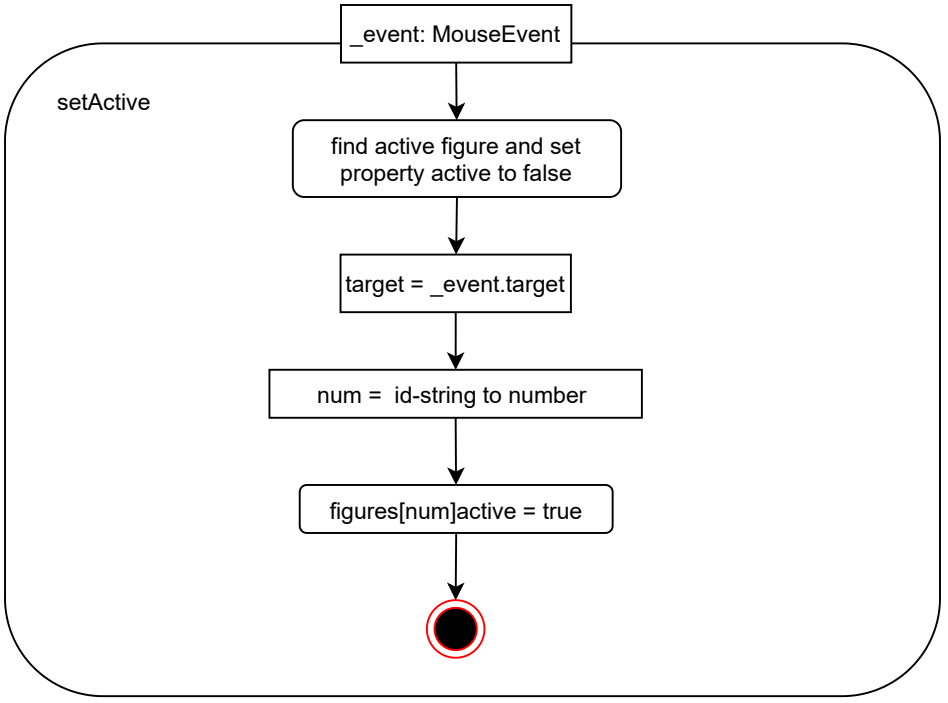

setCanvasWidth
get value from input-field with id = canvasWidth
canvas.width = value from input-field
createBackground


setCanvasHeight
get value from input-field with id = canvasHeight
canvas.height = value from input-field
createBackground


createPattern
target = _event.target
id = target.id
[id == "dots"]
[id == "plain"
backgroundPattern = "dots"
backgroundPattern = "squares"
backgroundPattern = "plain"
createBackground


createBackground
_color?: string
[if _color]
background = _color
background = backgroundColor.value
[backgroundPattern == dots]
[backgroundPattern == plain]
create a pattern with small dots for the background. The fillstyle is the value of background and the strokestyle is the value of patternColor
create a pattern with small squares for the background. The fillstyle is the value of background and the strokestyle is the value of patternColor
just fill the background with the value of background
fill the canvas with a rectangle the size of the canvas
give backgroundImage the value of crc2.getImageData with the size if the complete canvas
setAnimation
_event: MouseEvent
target: _event.target
id = target.id
[id == rotate]
[id == trash]
[id == move]
[id == threeD]
figure.moveType = rotate
deleteElement
figure.moveType = move
figure.neon = false
figure.threeD = false
[id == neon]
figure.neon = true
figure.threeD = false
figure.neon = false
figure.threeD = true
handleFormInput
_event: Event
target: _event.target
id = target.id
[id == colorPicker]
[id == rotationValue]
[id == speed]
find element with id = colorPicker
find element with id = rotationValue
find element with id = scaleValue
find element with id = speed
find active figure and call method changeColor with colorPicker.value
find active figure and call method changeRotation with rotationValue.value
find active figure and call method resize with scaleValue.value
find active figure and set velocity.x and y to speed.value
handleClick
_event: MouseEvent
x = x-Position of Mouse
y = y-Position of Mouse
find active figure
figure.position.x = x
figure.position.y = y
deleteElement
_figure: Form
index = figures.indexOf(_figure
figures.splice(index, 1)
updateList

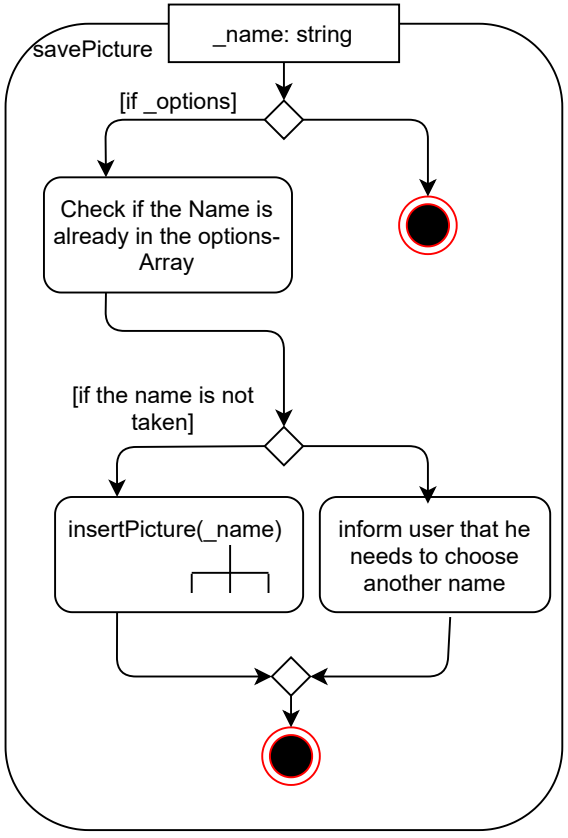

getName
create a prompt window for the user to enter a name for the image
[if the user entered a name]
make an alert to tell the user that the picture can only be saved with a name
savePicture(pictureName)

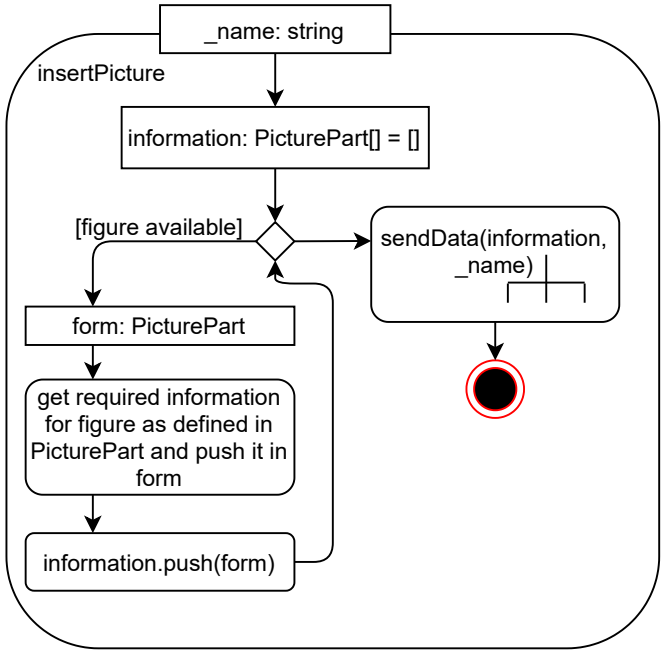

setActive
_event: MouseEvent
find active figure and set property active to false
target = _event.target
num = id-string to number
figures[num]active = true

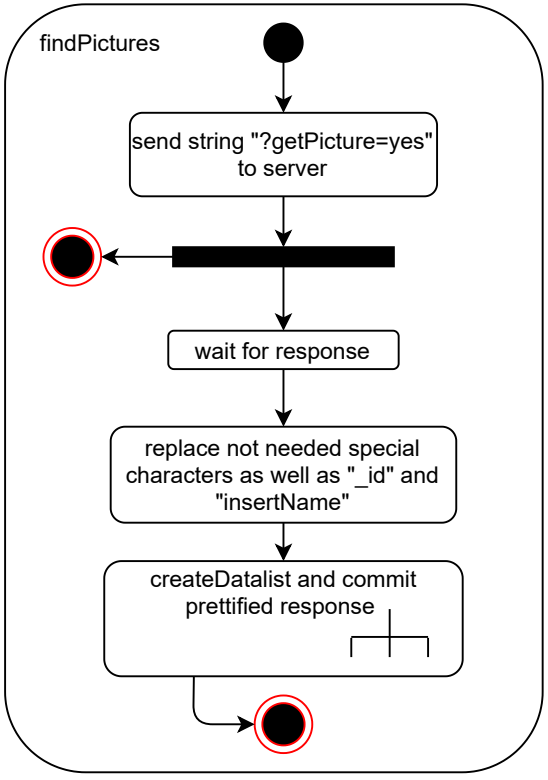
AD connectServer

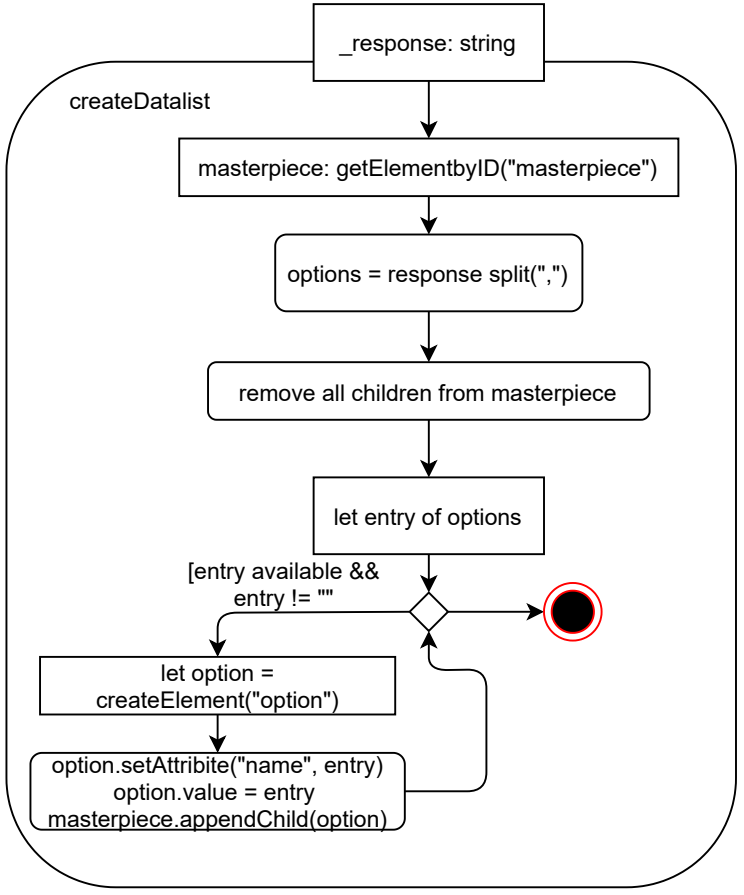
PicturePart
active: boolean size: Vector positionX: number positionY: number rotation: number moveType: FORM_MOVE color: string velocity: Vector neon: boolean threeD: boolean type: string

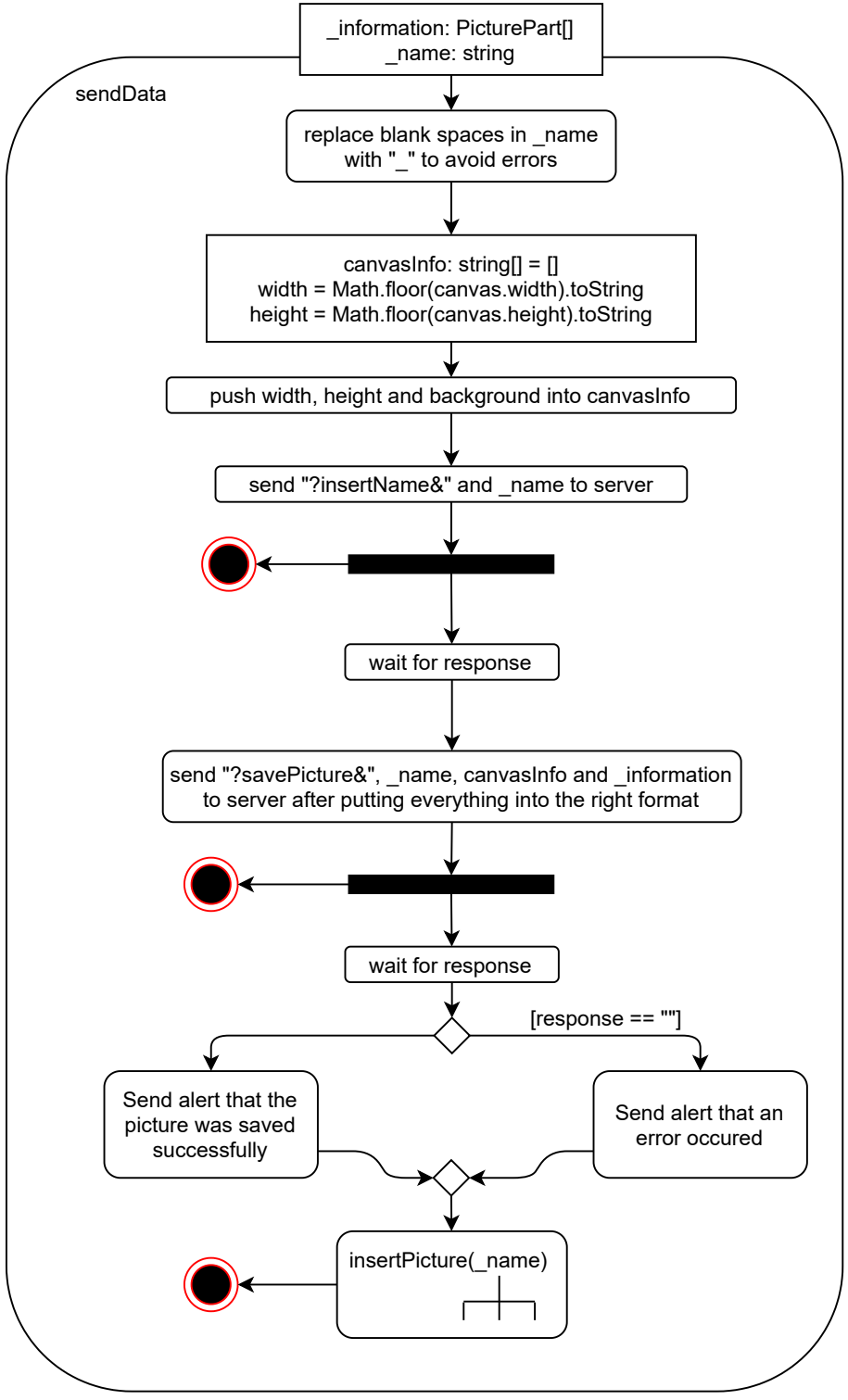
Globals
url: string = Link to Server
options: string[] = []

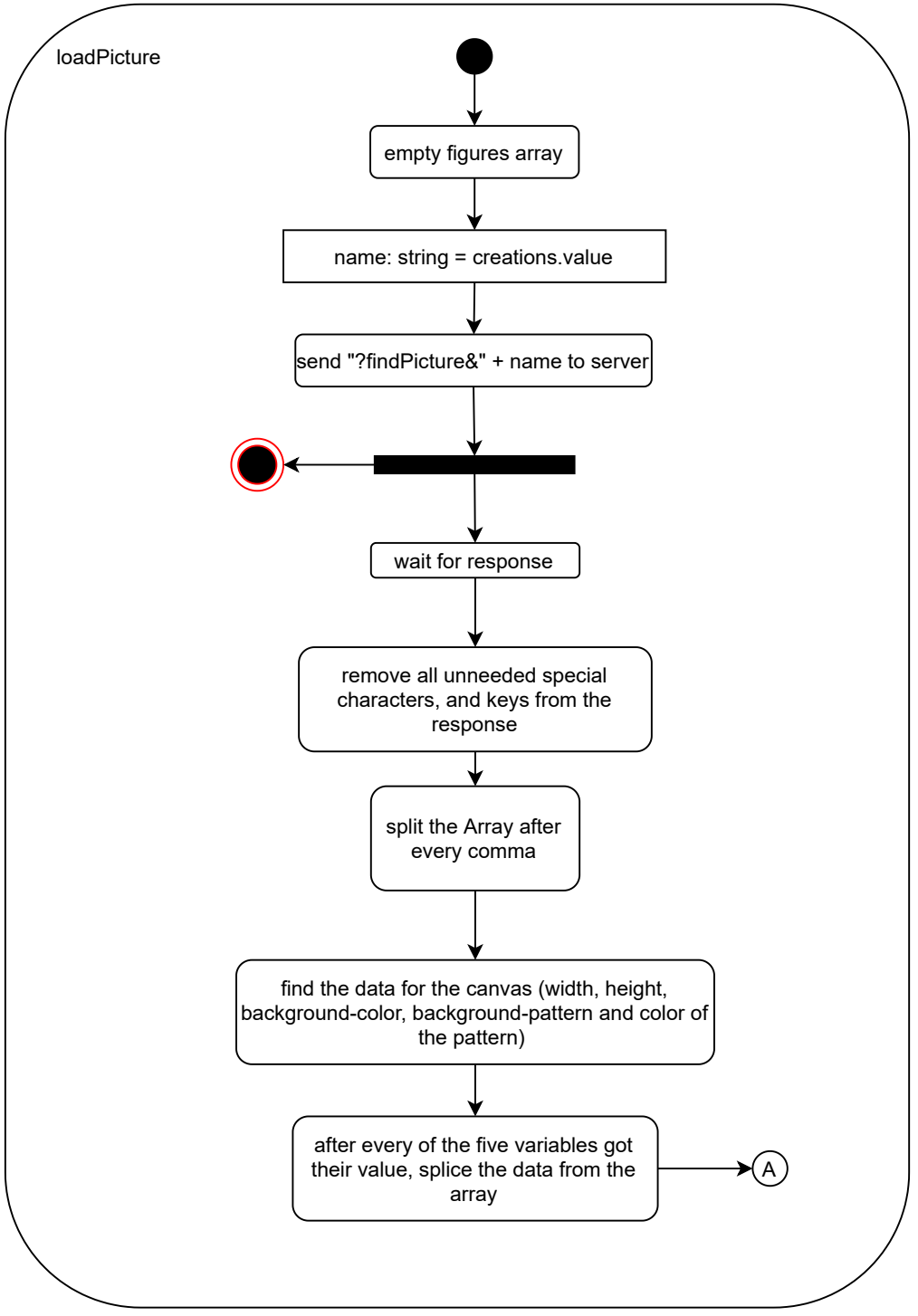


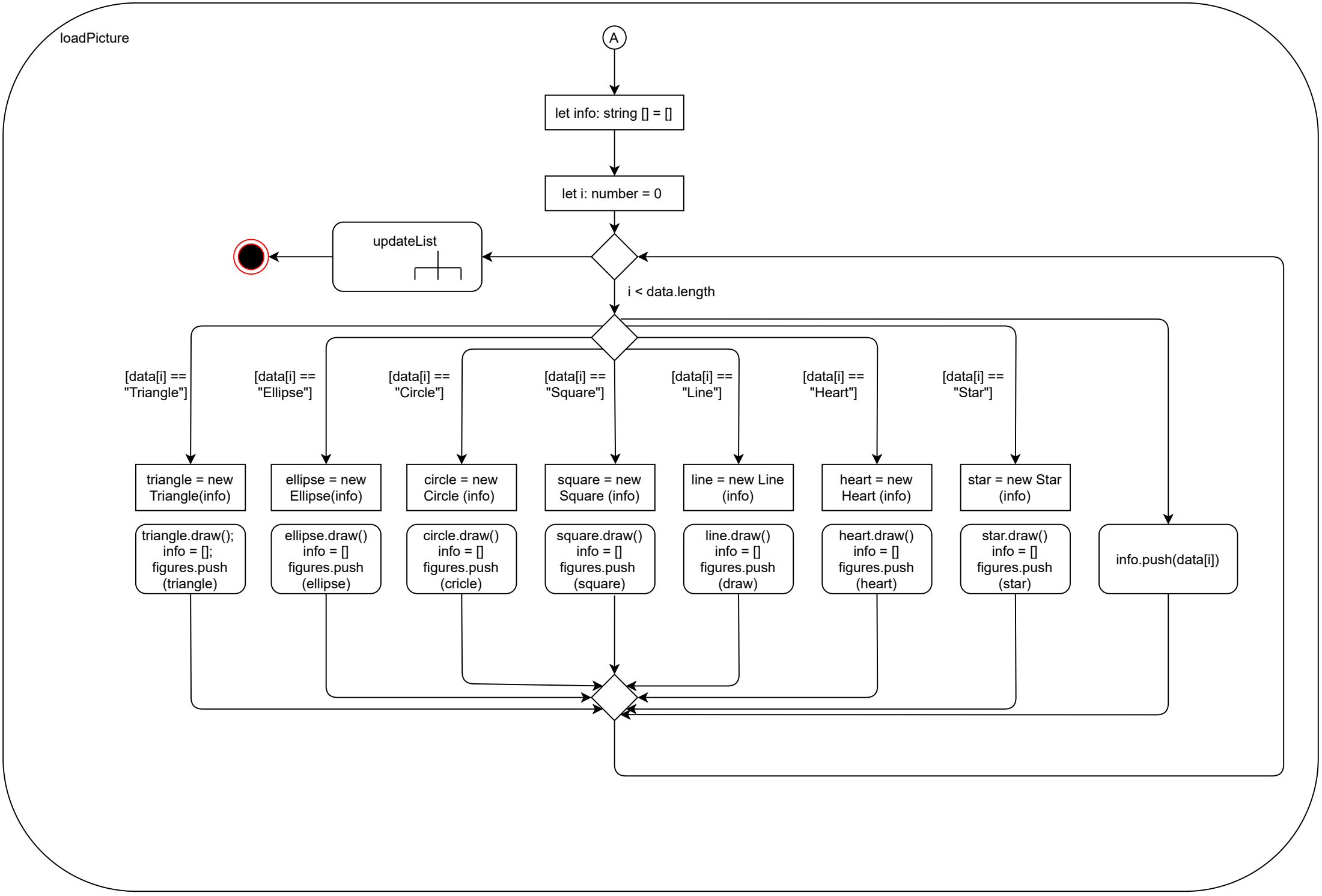




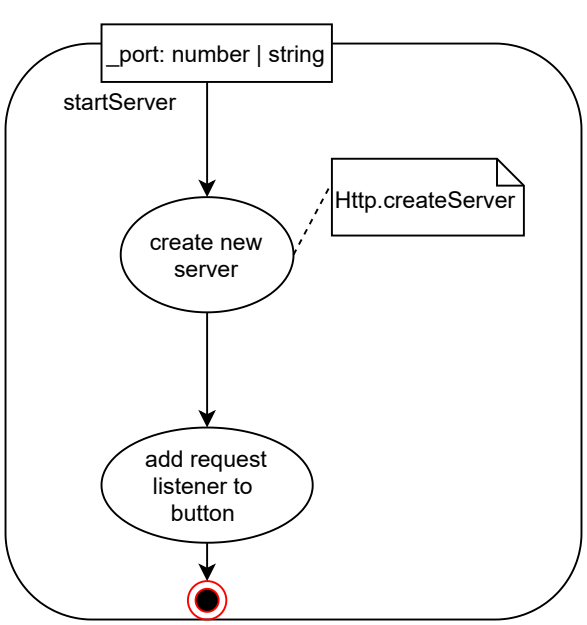


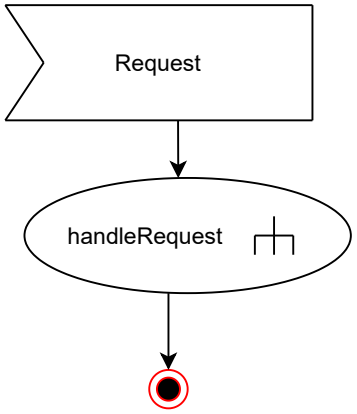

_information: PicturePart[]
_name: string
sendData
replace blank spaces in _name with "_" to avoid errors
canvasInfo: string[] = []
width = Math.floor(canvas.width).toString
height = Math.floor(canvas.height).toString
push width, height and background into canvasInfo
send "?insertName&" and _name to server
wait for response
send "?savePicture&", _name, canvasInfo and _information to server after putting everything into the right format
wait for response
[response == ""]
Send alert that the picture was saved successfully
Send alert that an error occured
insertPicture(_name)

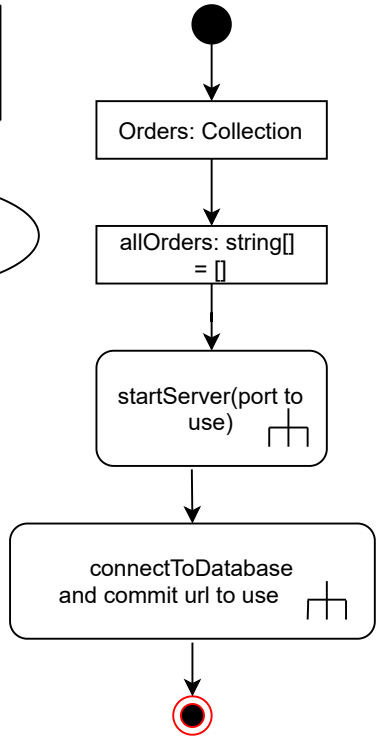

loadPicture
empty figures array
name: string = creations.value
send "?findPicture&" + name to server
wait for response
remove all unneeded special characters, and keys from the response
split the Array after every comma
find the data for the canvas (width, height, background-color, background-pattern and color of the pattern)
after every of the five variables got their value, splice the data from the array
A

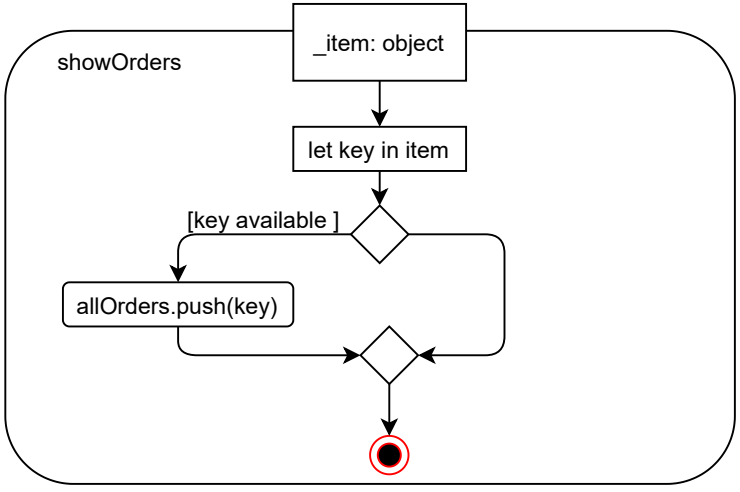

loadPicture
A
let info: string [] = []
let i: number = 0
updateList
i < data.length
[data[i] == "Triangle"]
[data[i] == "Ellipse"]
[data[i] == "Circle"]
[data[i] == "Square"]
[data[i] == "Line"]
[data[i] == "Heart"]
[data[i] == "Star"]
triangle = new Triangle(info)
ellipse = new Ellipse(info)
circle = new Circle (info)
square = new Square (info)
line = new Line (info)
heart = new Heart (info)
star = new Star (info)
triangle.draw(); info = []; figures.push (triangle)
ellipse.draw() info = [] figures.push (ellipse)
circle.draw() info = [] figures.push (cricle)
square.draw() info = [] figures.push (square)
line.draw() info = [] figures.push (draw)
heart.draw() info = [] figures.push (heart)
star.draw() info = [] figures.push (star)
info.push(data[i])

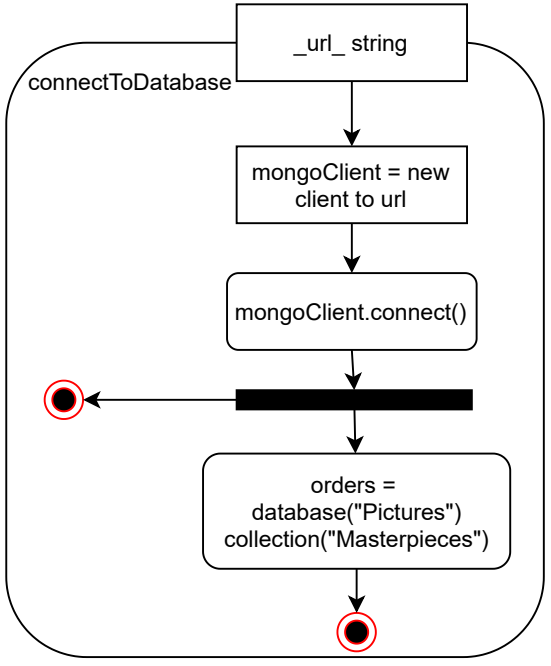
AD Server

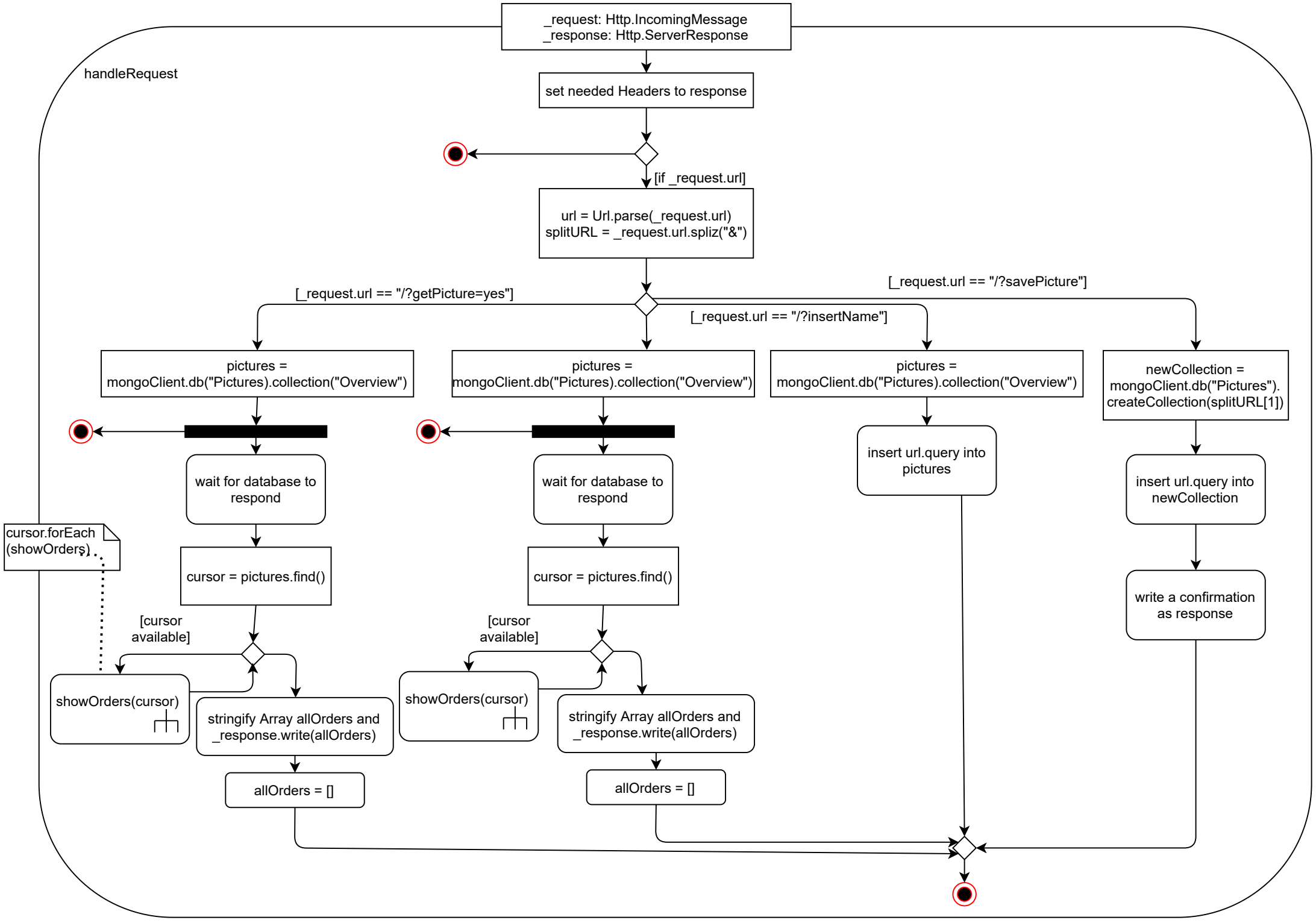

_port: number | string
startServer
create new server
Http.createServer
add request listener to button


Request
handleRequest


Orders: Collection
allOrders: string[] = []
startServer(port to use)
connectToDatabase and commit url to use


showOrders
_item: object
let key in item
[key available]
allOrders.push(key)


connectToDatabase
url string
mongoClient = new client to url
mongoClient.connect()
orders = database("Pictures") collection("Masterpieces")


handleRequest
_request: Http.IncomingMessage
_response: Http.ServerResponse
set needed Headers to response
[if _request.url]
url = Url.parse(_request.url)
splitURL = _request.url.spliz("&")
[_request.url == "/?getPicture=yes"]
[_request.url == "/?insertName"]
[_request.url == "/?savePicture"]
pictures = mongoClient.db("Pictures").collection("Overview")
pictures = mongoClient.db("Pictures").collection("Overview")
pictures = mongoClient.db("Pictures").collection("Overview")
newCollection = mongoClient.db("Pictures").createCollection(splitURL[1])
wait for database to respond
wait for database to respond
insert url.query into pictures
insert url.query into newCollection
cursor.forEach (showOrders)
cursor = pictures.find()
cursor = pictures.find()
write a confirmation as response
[cursor available]
[cursor available]
showOrders(cursor)
showOrders(cursor)
stringify Array allOrders and _response.write(allOrders)
stringify Array allOrders and _response.write(allOrders)
allOrders = []
allOrders = []