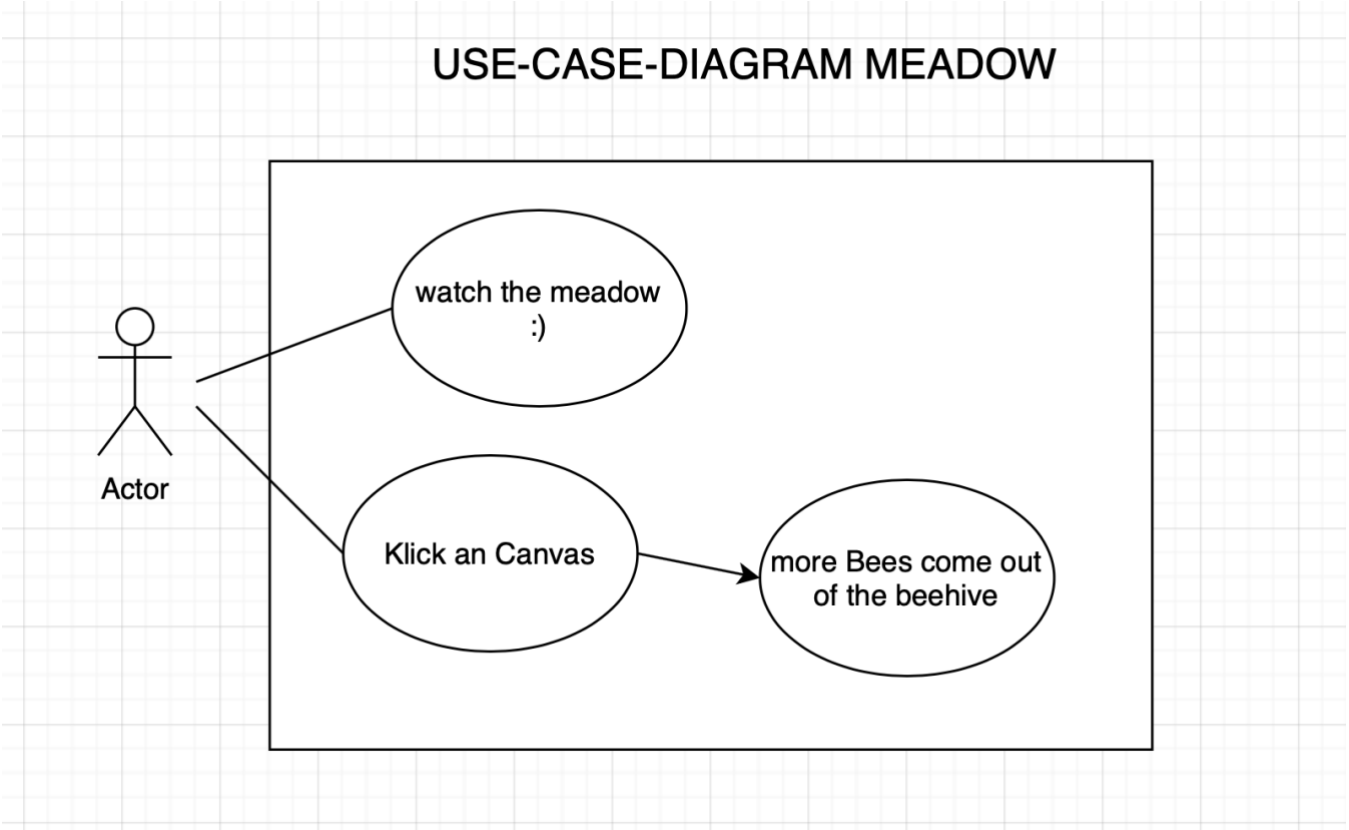
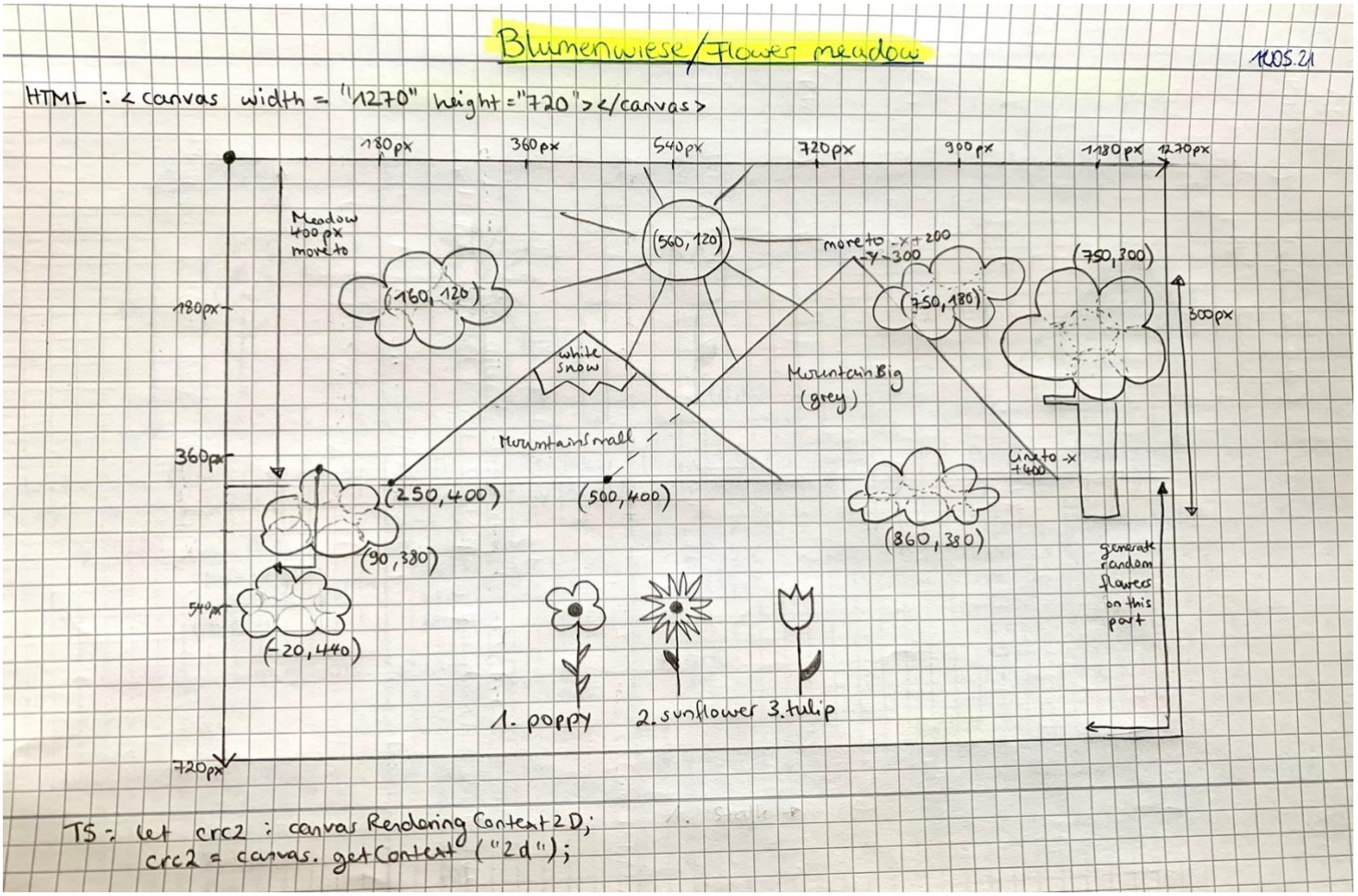
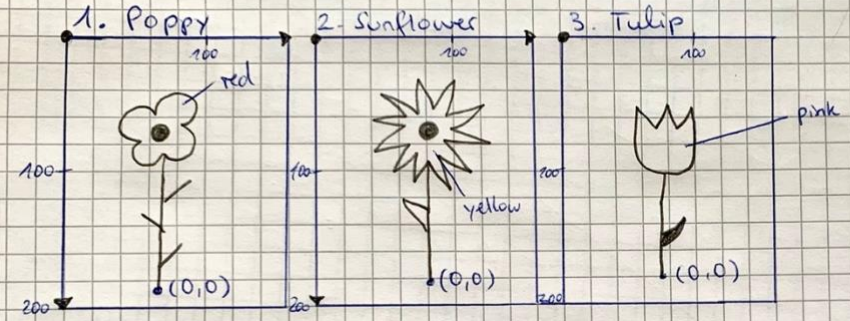
KONZEPT MEADOW
MONA STINGL





11.05.2

Funktion Blumen



function drawPoppy (_x: number, _y: number): void {

- Stalk: beginPath, strokeStyle } green, fillStyle } green, fillRect
- leaves: 2x moveTo & lineTo, stroke, fill
- blossoms: beginPath, fillStyle - (red), moveTo, 4x arc, fill
- blossoms-middle: arc

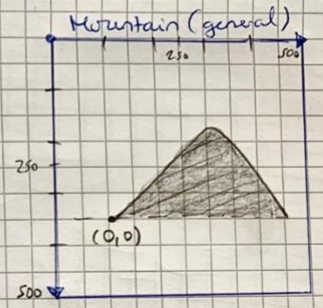
function drawSunflower (_x: number, _y: number) void {

- beginPath, strokeStyle } green, fillStyle, fillRect
- lineTo, lineTo, lineTo } on the stalk
- beginPath, fillStyle - (yellow), moveTo, 11 lineTo, fill
- arc

function drawTulip (_x: number, _y: number): void {

- beginPath, fillStyle - green, fillRect
- arc, fill
- beginPath, fillStyle - (pink), arc, moveTo, 6x lineTo, closePath, fill (no middle)

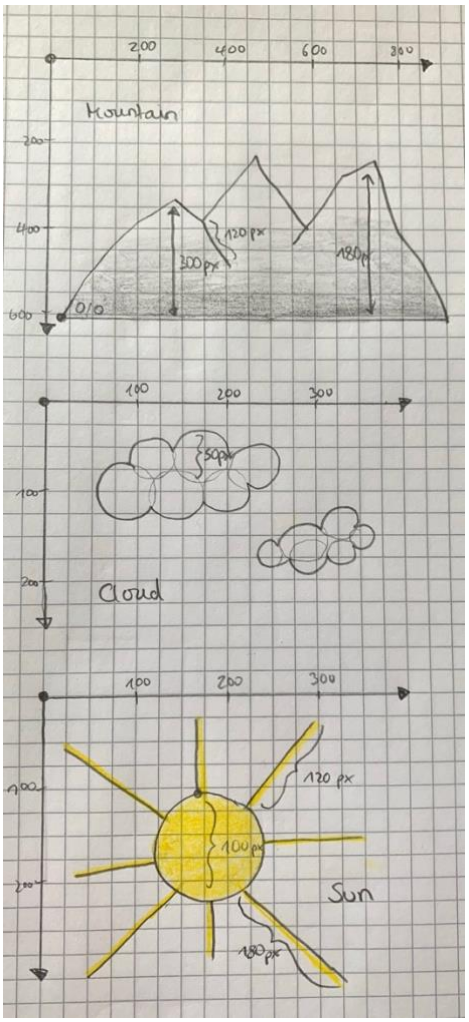
Funktion Berg

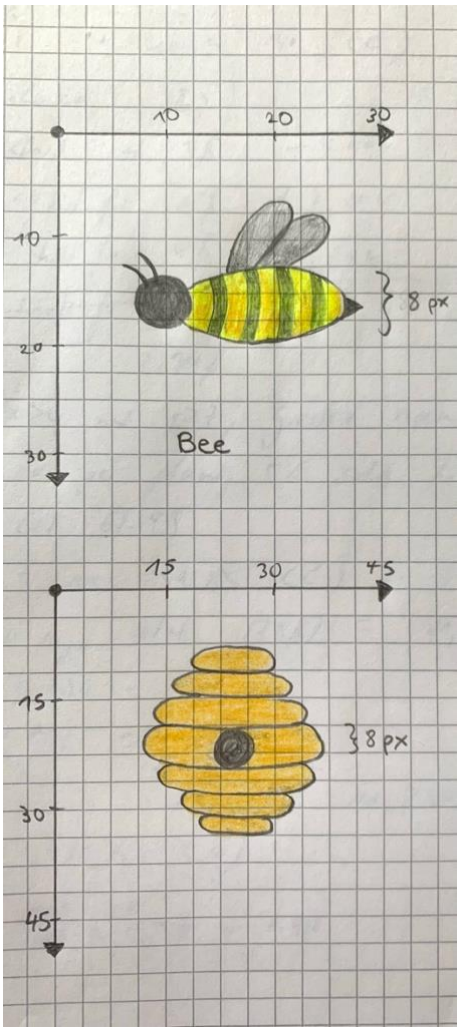


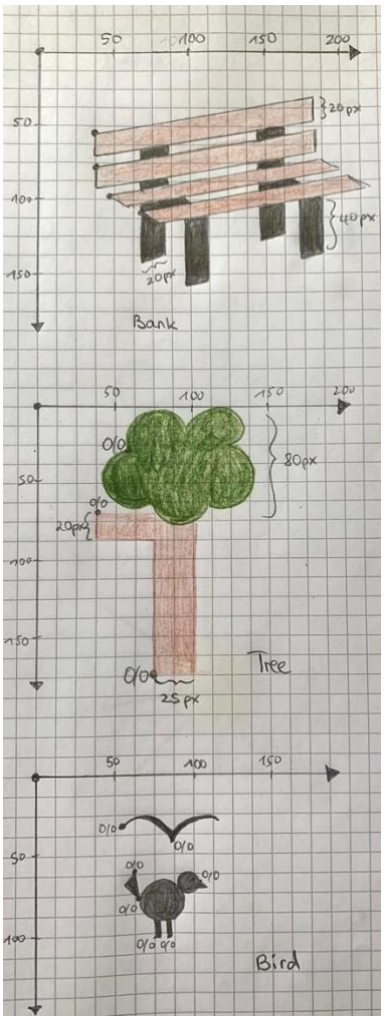
function drawMountain (_x: number, _y: number, ...)

```
crc2.beginPath();
crc2.fillStyle = ...
crc2.moveTo(...)
crc2.lineTo(...)
crc2.lineTo(...)
crc2.closePath();
crc2.fill()
```

Others
cloud → arcTo & curveTo
sun → arcTo / curveTo (?) lineTo
tree → arcTo, curveTo, lineTo
bush → arcTo, curveTo

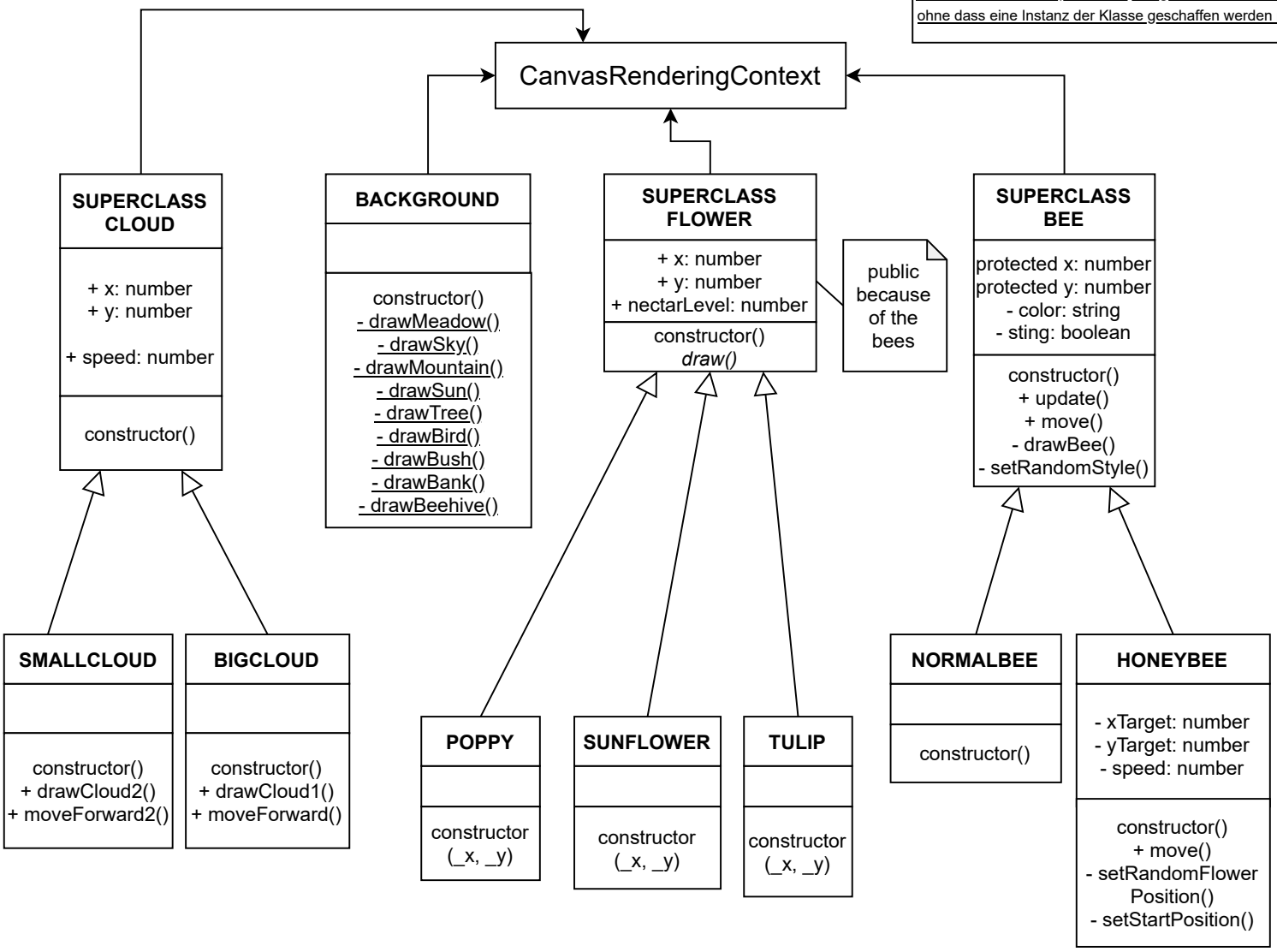




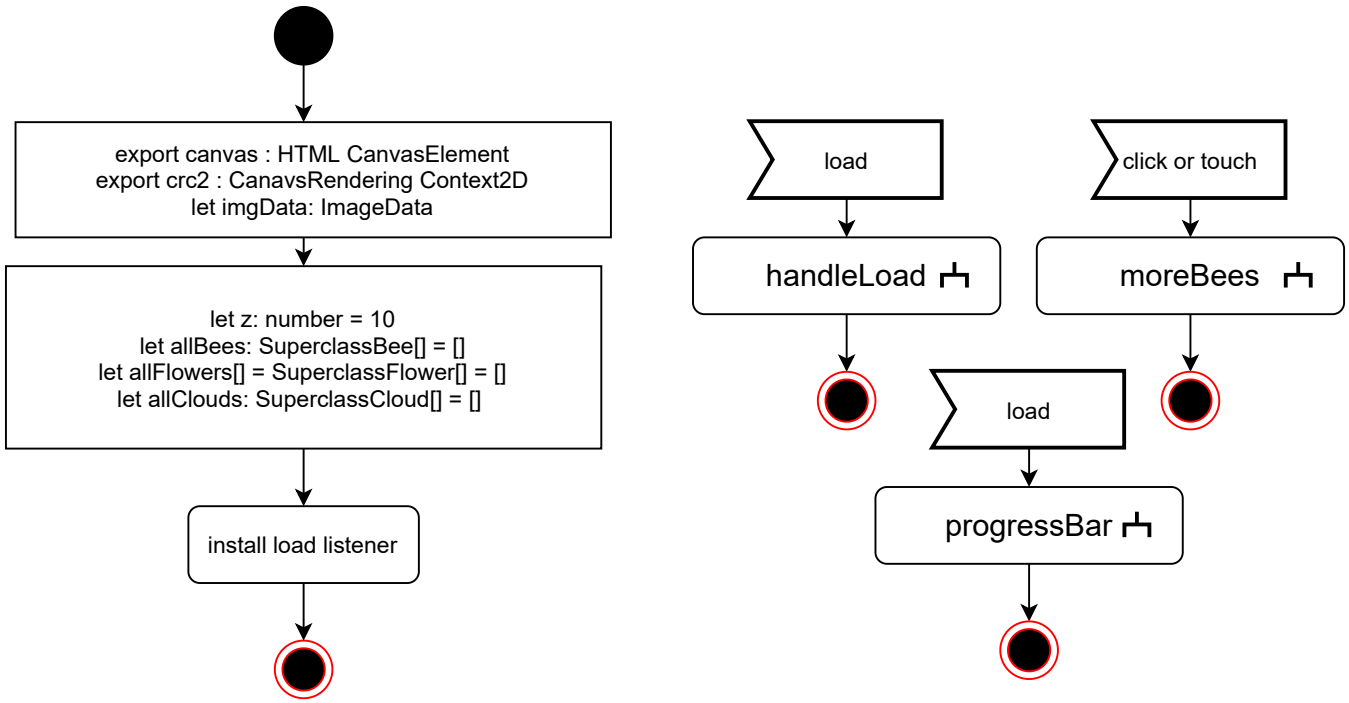


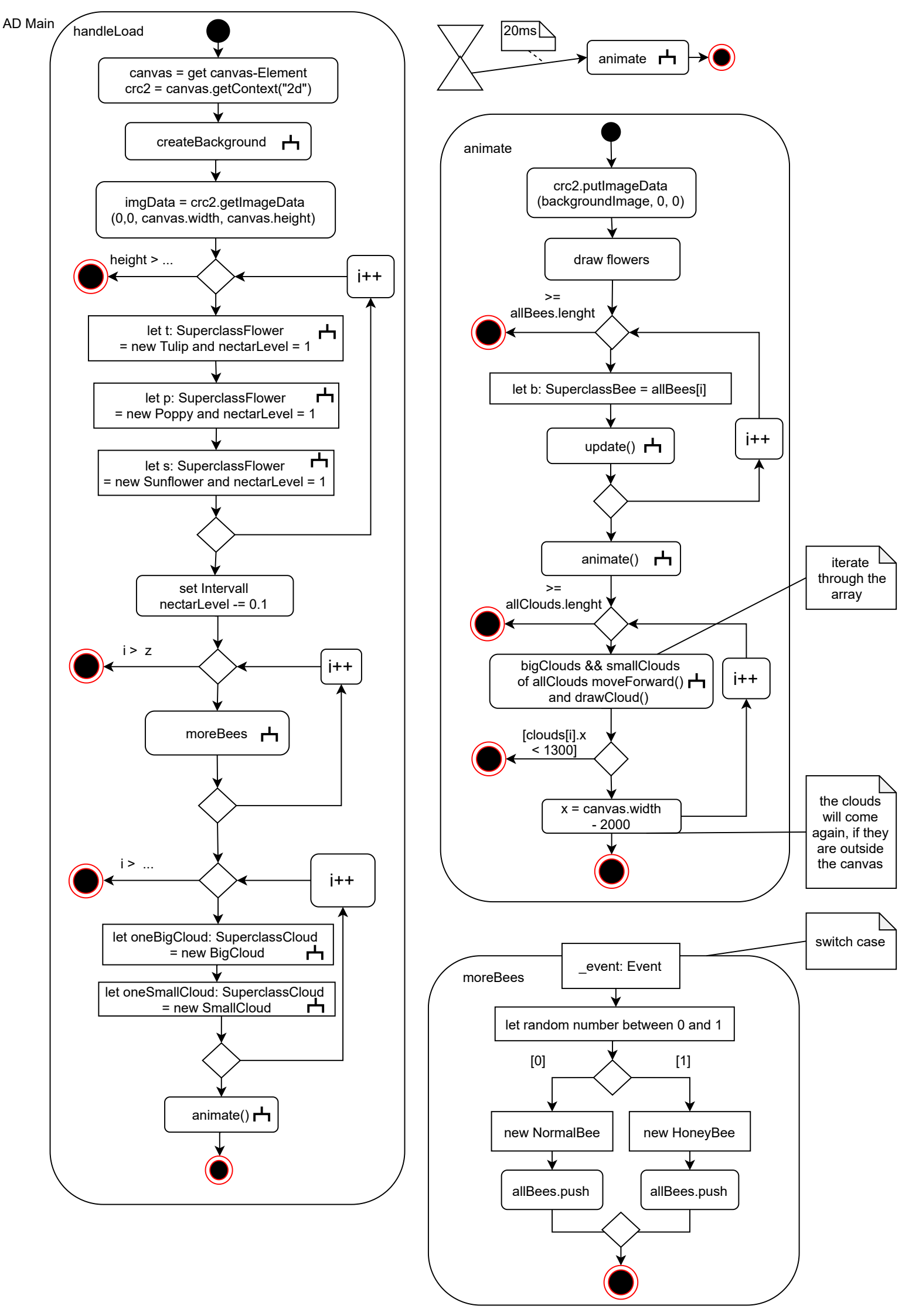
CLASS-DIAGARM MEADOW

kursiv = Abstrakt (Wenn Superklassen Methoden aufweisen, die nur die Subklassen brauchen)
+ = public (wenn alle darauf Zugriff brauchen)
- = private (Zugriff nur in der Klasse)
protected = Zugriff gleichen Klassen und deren Subklassen
unterstrichen = static (Methoden, die genutzt werden können, ohne dass eine Instanz der Klasse geschaffen werden muss)

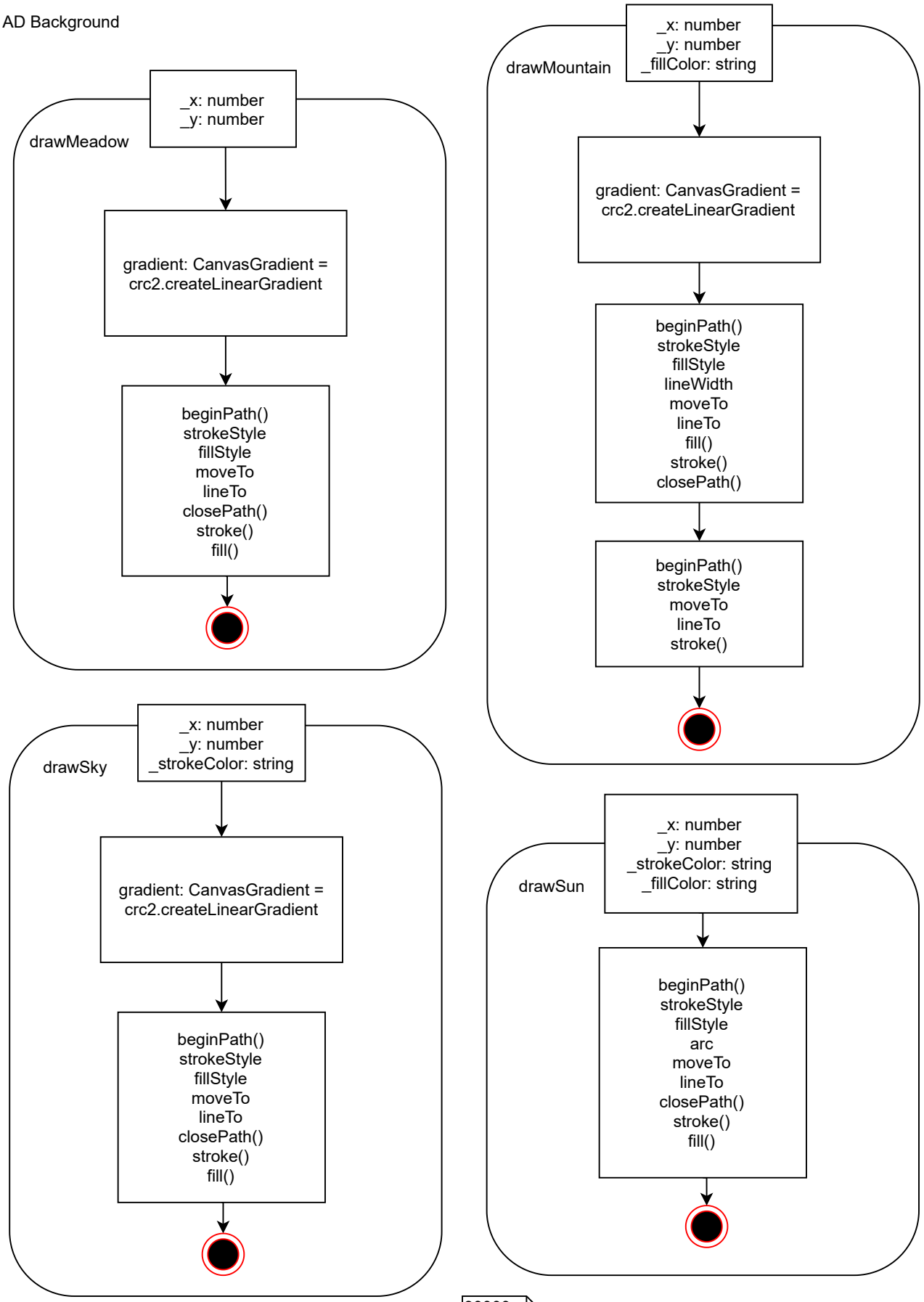


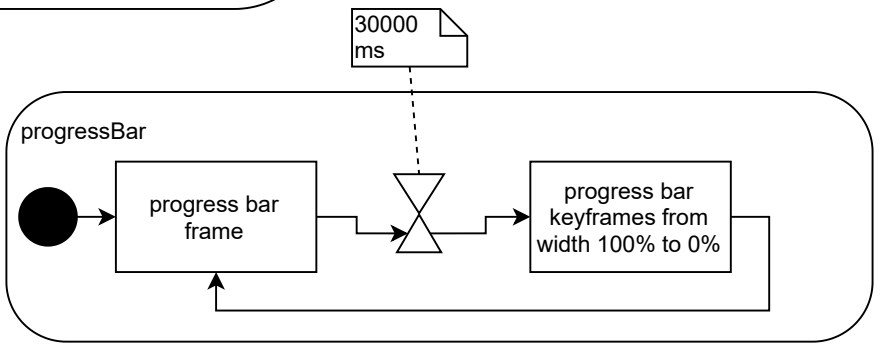
ACTIVITY-DIAGRAM MEADOW



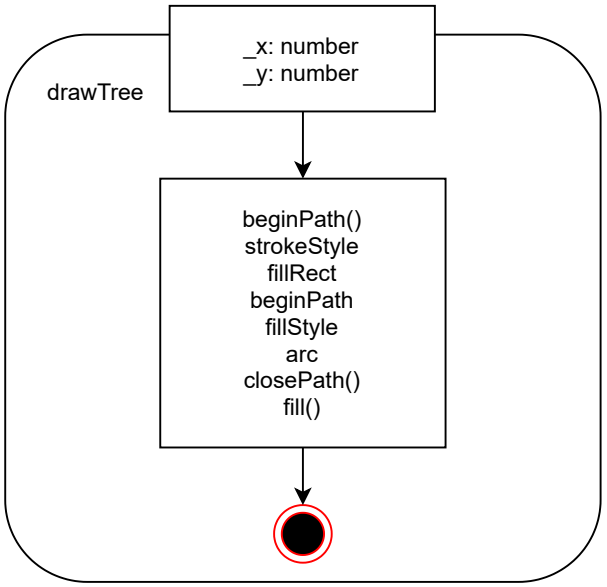

AD Main
handleLoad
canvas = get canvas-Element
crc2 = canvas.getContext("2d")
createBackground
imgData = crc2.getImageData
(0,0, canvas.width, canvas.height)
height > ...
i++
let t: SuperclassFlower
= new Tulip and nectarLevel = 1
let p: SuperclassFlower
= new Poppy and nectarLevel = 1
let s: SuperclassFlower
= new Sunflower and nectarLevel = 1
set Intervall
nectarLevel -= 0.1
i > z
i++
moreBees
i > ...
i++
let oneBigCloud: SuperclassCloud
= new BigCloud
let oneSmallCloud: SuperclassCloud
= new SmallCloud
animate()
20ms
animate
animate
crc2.putImageData
(backgroundImage, 0, 0)
draw flowers
>=
allBees.lenght
let b: SuperclassBee = allBees[i]
update()
i++
animate()
>=
allClouds.lenght
bigClouds && smallClouds
of allClouds moveForward()
and drawCloud()
i++
[clouds[i].x
< 1300]
x = canvas.width
- 2000
iterate
through the
array
the clouds
will come
again, if they
are outside
the canvas
moreBees
_event: Event
switch case
let random number between 0 and 1
[0]
[1]
new NormalBee
new HoneyBee
allBees.push
allBees.push

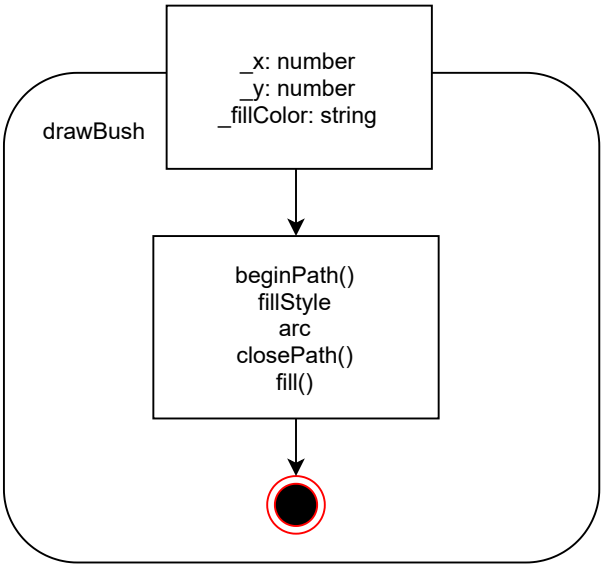
AD Background

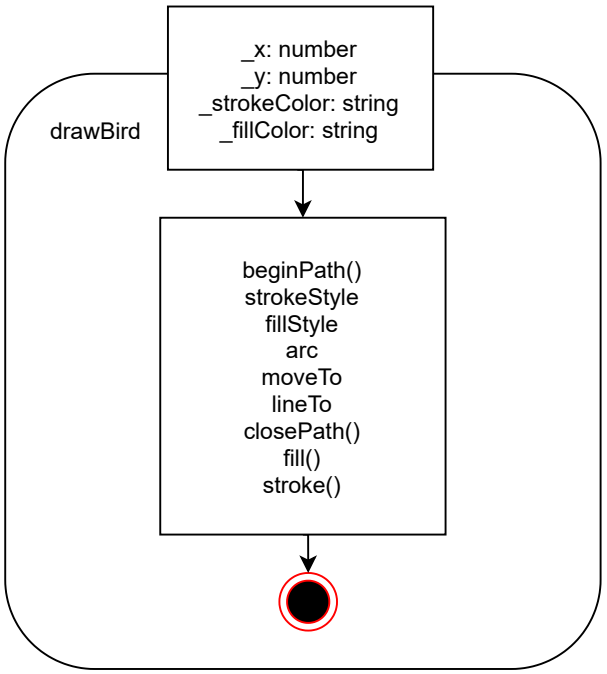


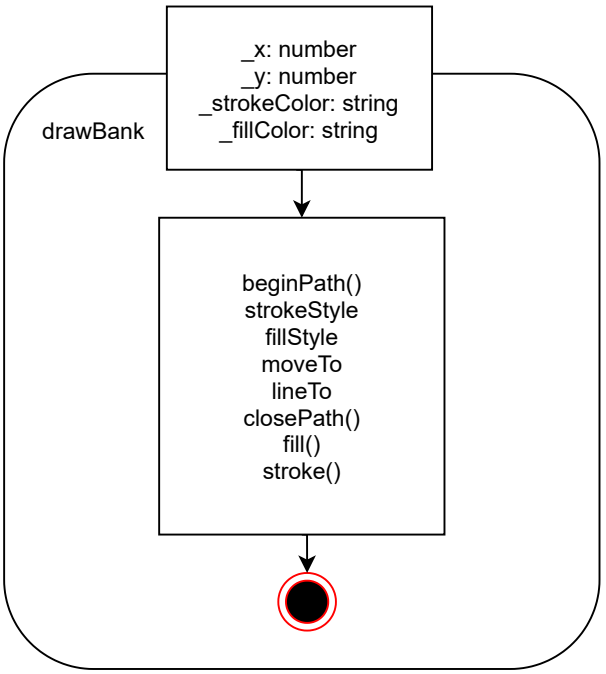


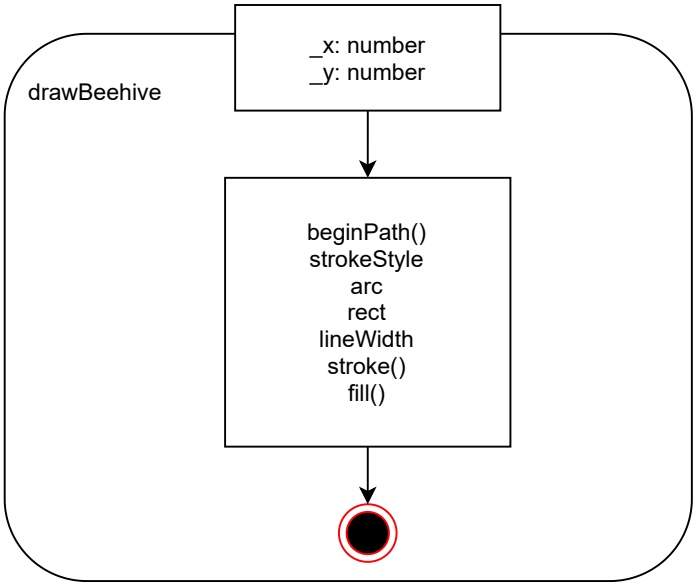
AD Background



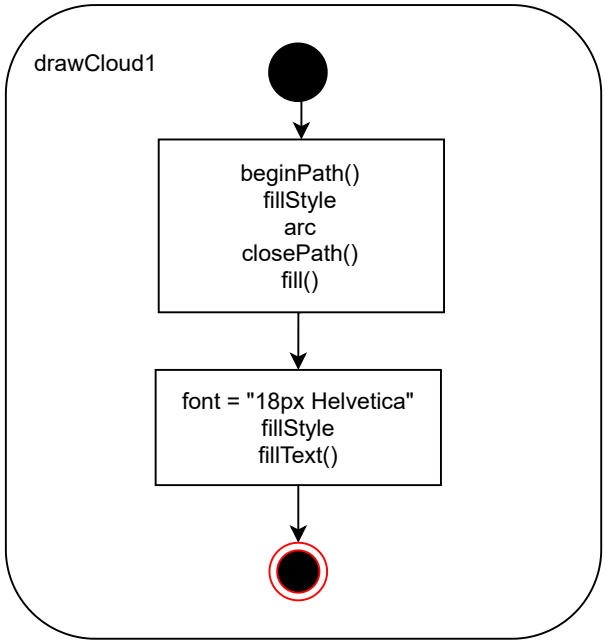


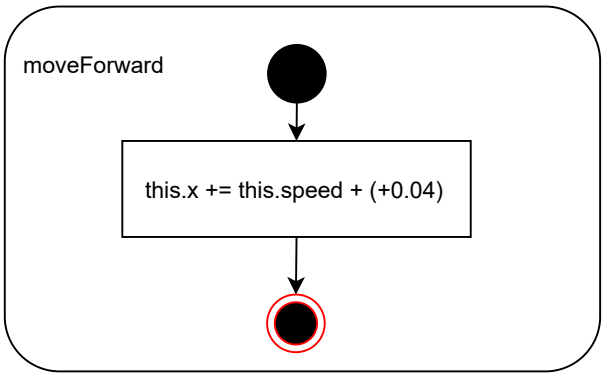




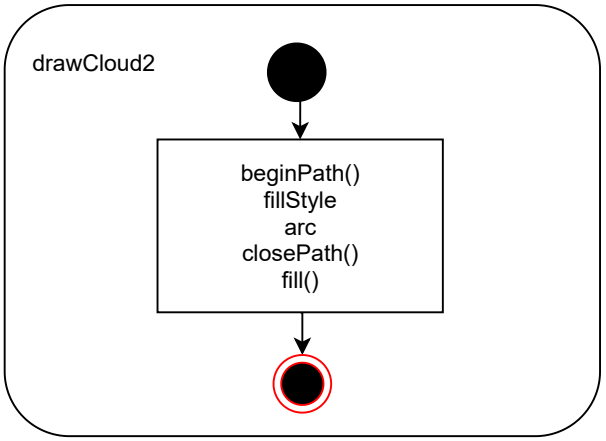


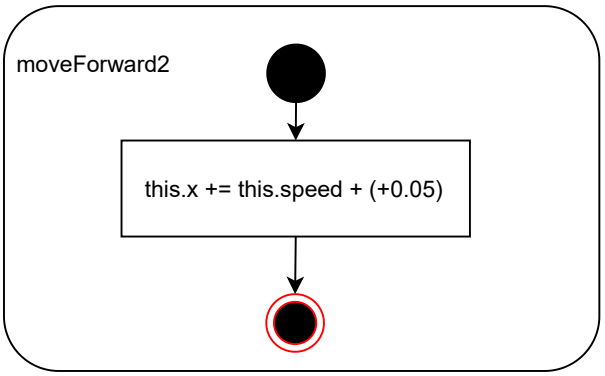
AD Subclass BigCloud



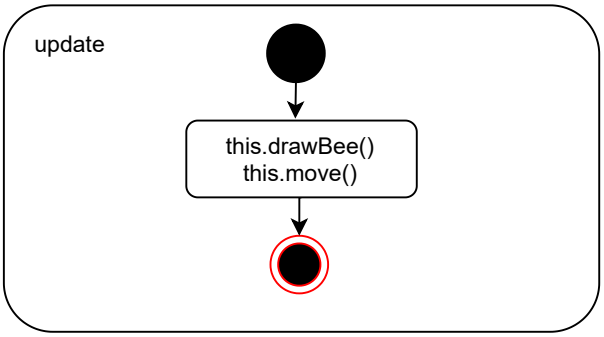


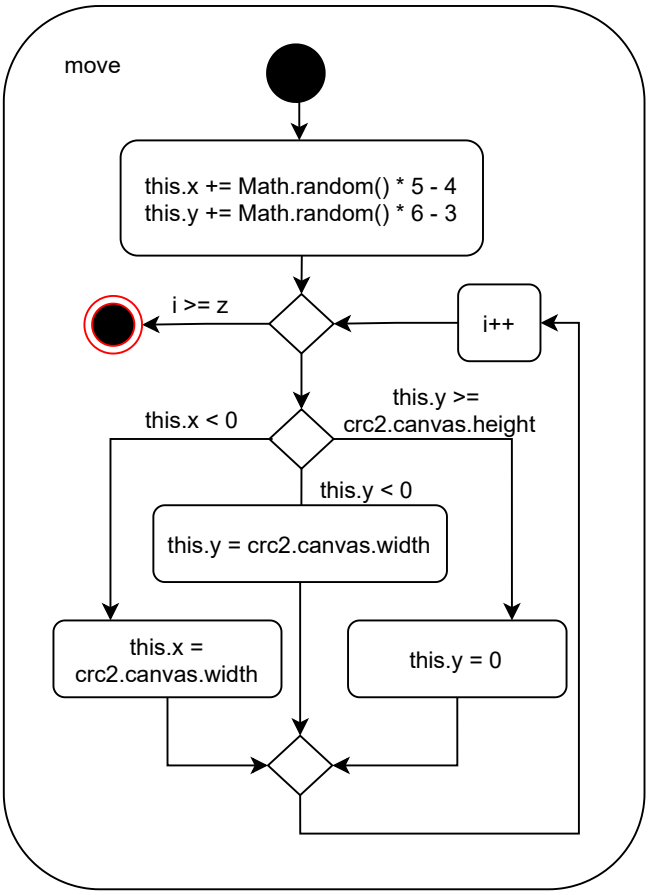
AD Subclass SmallCloud

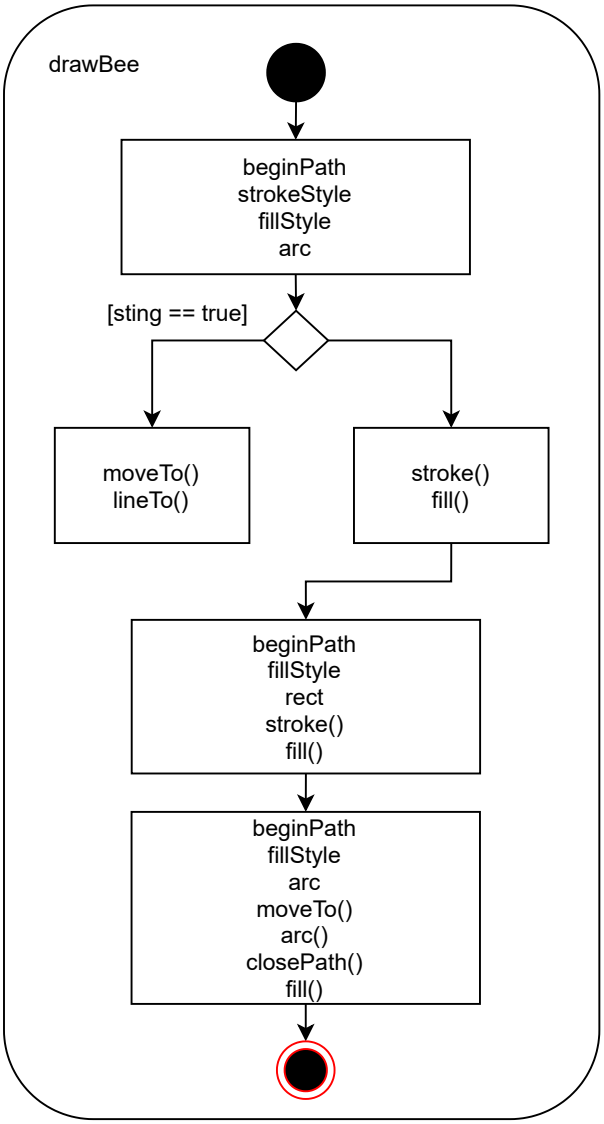


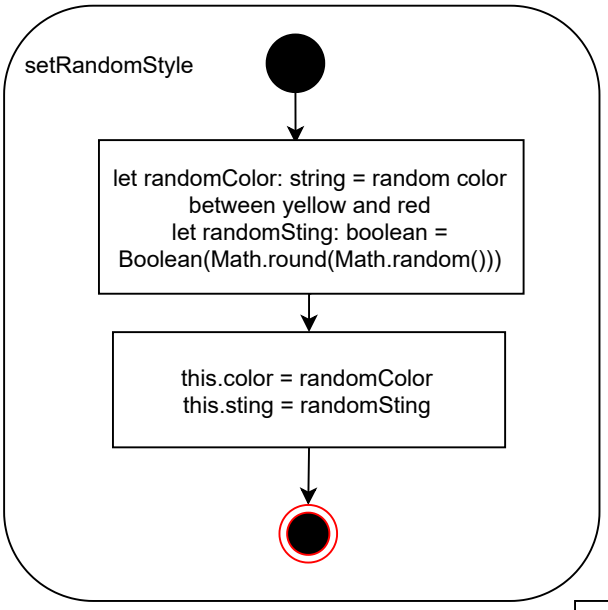


AD SuperclassBee

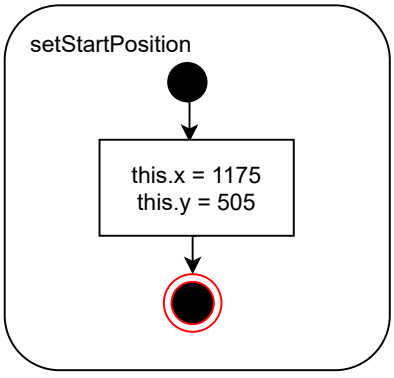


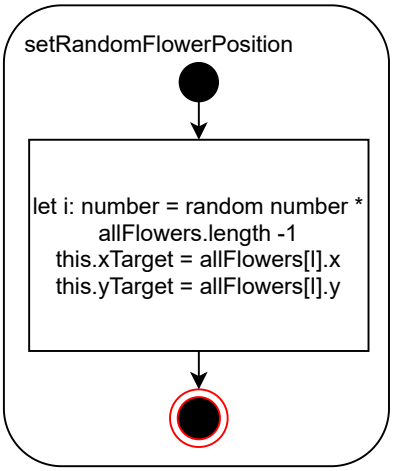


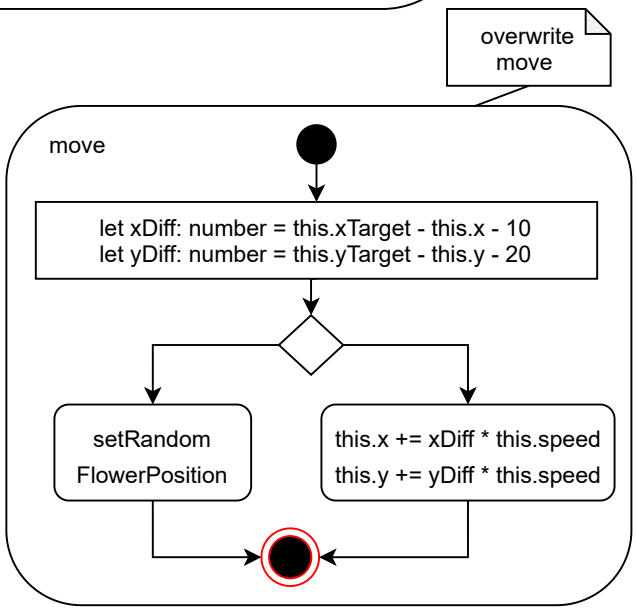




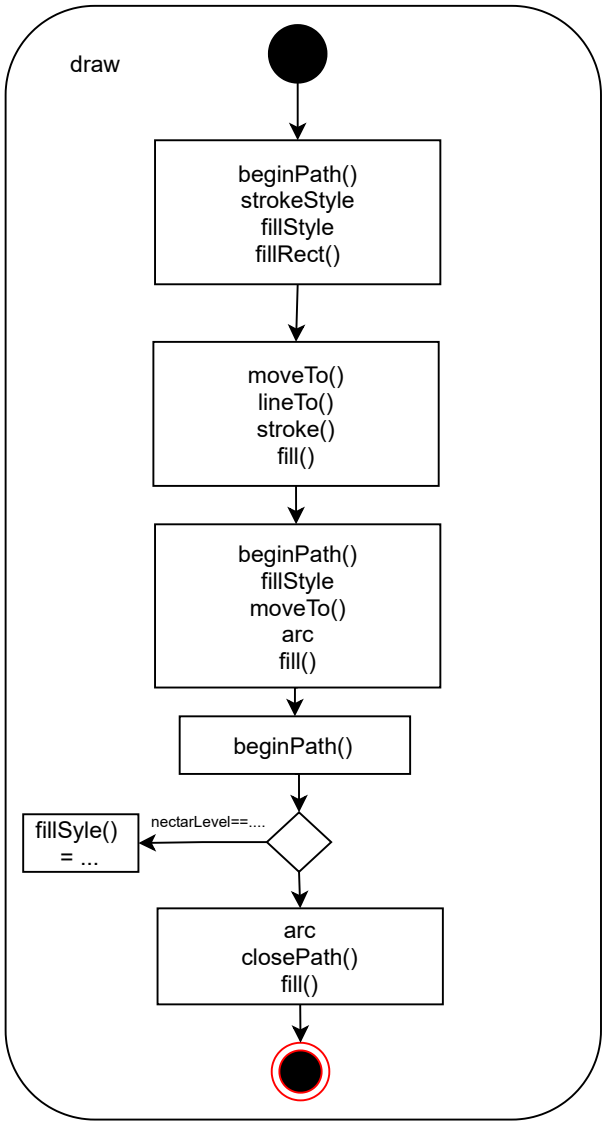
AD Subclass HoneyBee



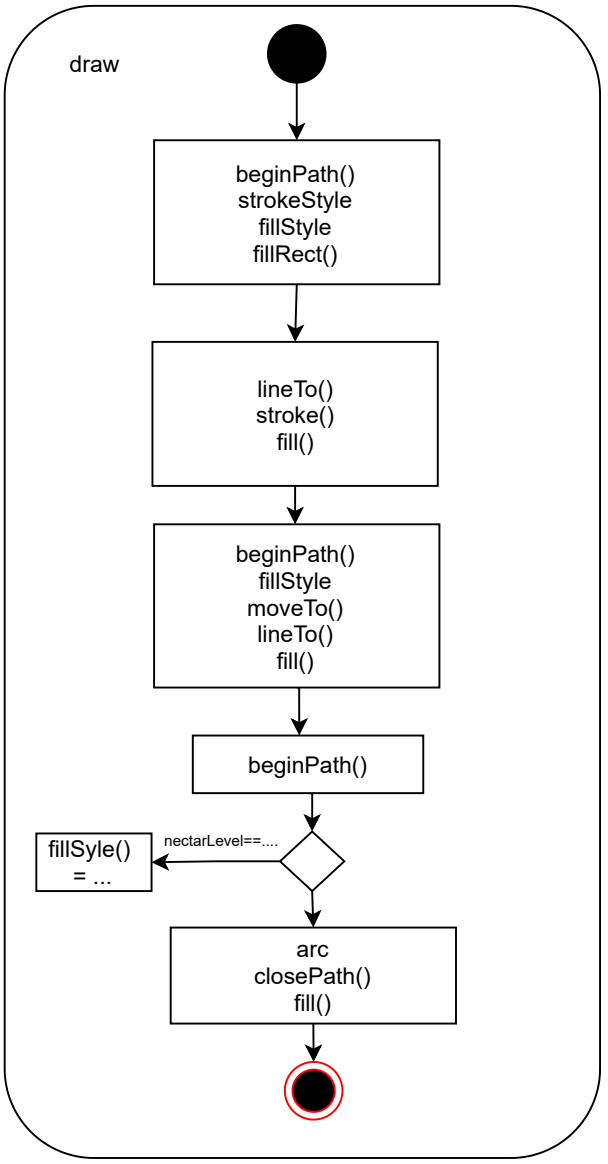
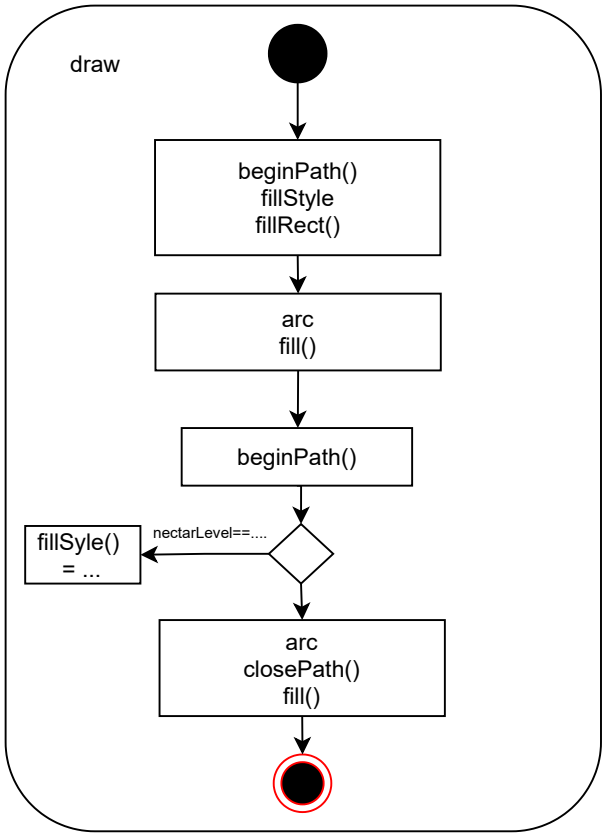




AD Subclass Poppy



AD Subclass Sunflower



AD Subclass Tulip