Creative Coding

WiSe 2019/20 – Grundlagen

Präsentationen

Präsentationen

Imperative Programmierung

Imperative Programmierung

Imperative Programmierung

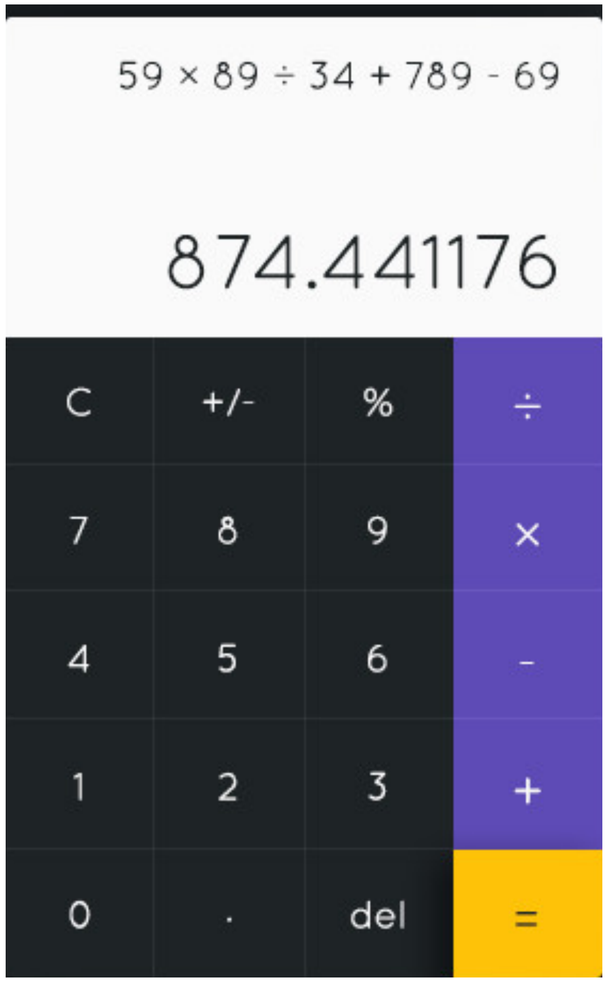
Das WIE

Imperative Programmierung

Das WAS

Deklarative Programmierung

Datentypen



Datentypen

Datentypen

Bei charmanten Bären sind Ihre
leckeren Früchte dahin

Datentypen

Bei charmanten Bären sind Ihre
leckeren Früchte dahin

Datentypen

Bei charmanten Bären sind Ihre
leckeren Früchte dahin

Datentypen

byte
char
bool
short
int
long
float
double

Bei charmanten Bären sind Ihre leckeren Früchte dahin

Datentypen

byte
char
bool
short
int (2,147,483,647 – -2,147,483,648)
long
float
double

Bei charmanten Bären sind Ihre leckeren Früchte dahin

Datentypen

byte
char
bool
short
int (2,147,483,647 – -2,147,483,648)
long
float (1.0/3.0 = 0.33333334) → 0.33333334*3.0 = 1.0 (1.0)
double

Bei charmanten Bären sind Ihre leckeren Früchte dahin

Datentypen

Bei charmanten Bären sind Ihre
leckeren Früchte dahin

Datentypen

"B", "e", "i", " ", "c","h","a","r","m","a","n","t","e","n","
","B","ä","r","e","n"," ","s","i","n","d"," ","I","h","r","e","
","l","e","c","k","e","r","e","n"," ","F","r","ü","c","h","t","e","
","d","a","h","i","n","

Datentypen

```
final float constant = 12.84753;
println(constant);  // Gibt 12.84753 aus
```

Konstanten

```
println(1 + 1);  // Gibt 2 aus
```

Variablen

```
int zahl1 = 1;
println(zahl1 + 1);  // Gibt 2 aus
```

Variablen

```
int zahl1 = 1;
println(zahl1 + 1);  // Gibt 2 aus
zahl1 = 10;
println(zahl1 + 1);  // Gibt 11 aus
```

Variablen

```
int zahl1 = 1;
int zahl2 = 1;
println(zahl1 + zahl2);  // Gibt 2 aus

zahl1 = 10;
println(zahl1 + zahl2);  // Gibt 11 aus

println(zahl1 - zahl2);  // Subtrahieren
println(zahl1 * zahl2);  // Multiplizieren
println(zahl1 / zahl2);  // Dividieren
```

Variablen

```
int zahl1 = 1;
int zahl2 = 1;
println(zahl1 + zahl2);  // Gibt 2 aus
```

Variablen

```
int zahl1 = 1;
int zahl2 = 1;
int ergebnis = zahl1 + zahl2;
println(ergebnis);  // Gibt 2 aus
```

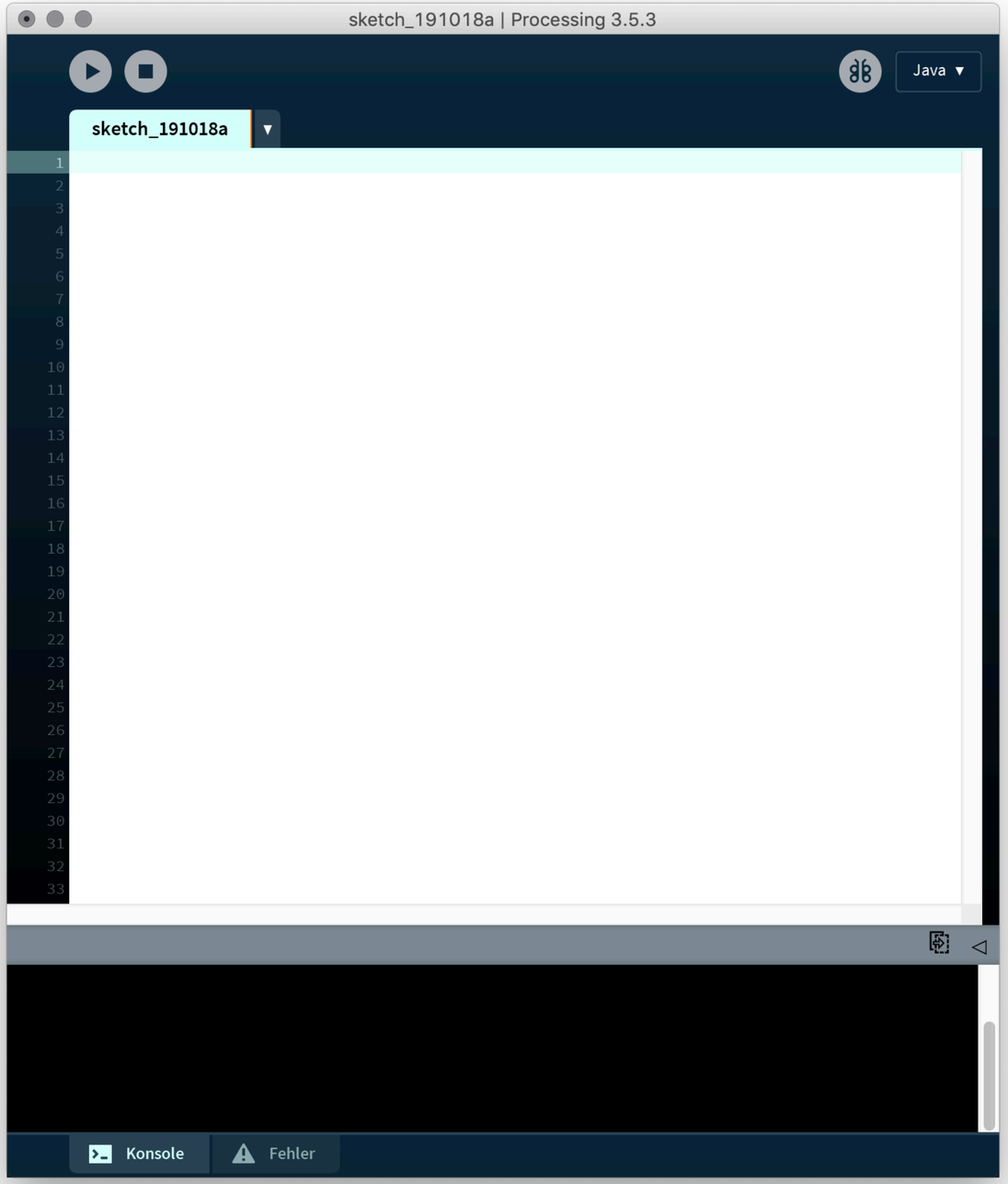
Variablen

```
int zahl1 = 1;
int zahl2 = 1;
int ergebnis = zahl1 + zahl2;
println(ergebnis);  // Gibt 2 aus

ergebnis = ergebnis + zahl1; // ergibt 3
ergebnis += zahl2; // ergibt 4
ergebnis = ergebnis + ergebnis; // 8
```

Variablen

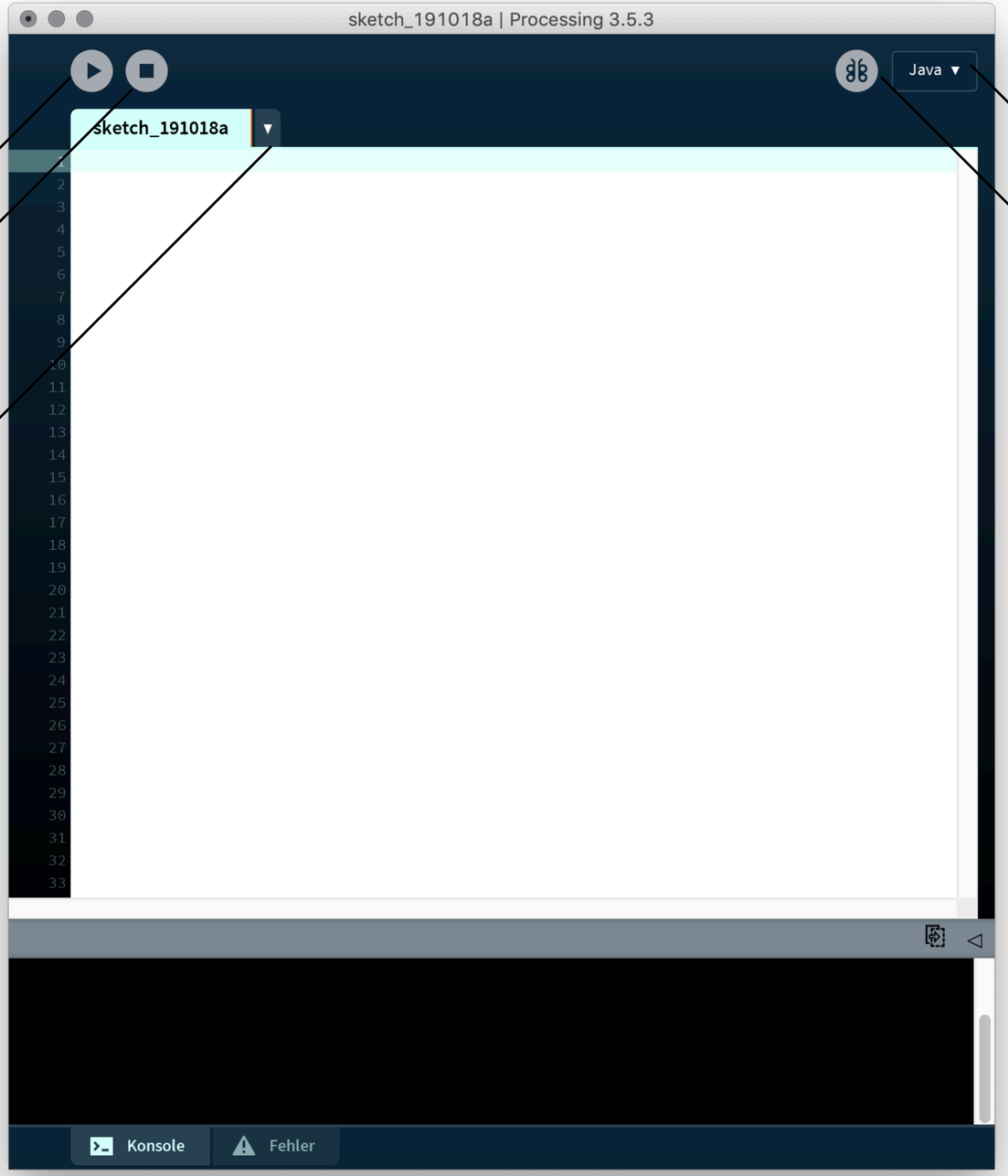
```
int zahl1 = 1;
int zahl2 = 1;
int ergebnis = zahl1 + zahl2;
println(ergebnis);  // Gibt 2 aus

ergebnis = ergebnis + zahl1; // ergibt 3
ergebnis += zahl2; // ergibt 4
ergebnis = ergebnis + ergebnis; // 8


String unserText = "Bei charmanten Bären sind ihre leckeren Früchte
dahin";
```

Variablen



Processing IDE

Processing IDE



Start

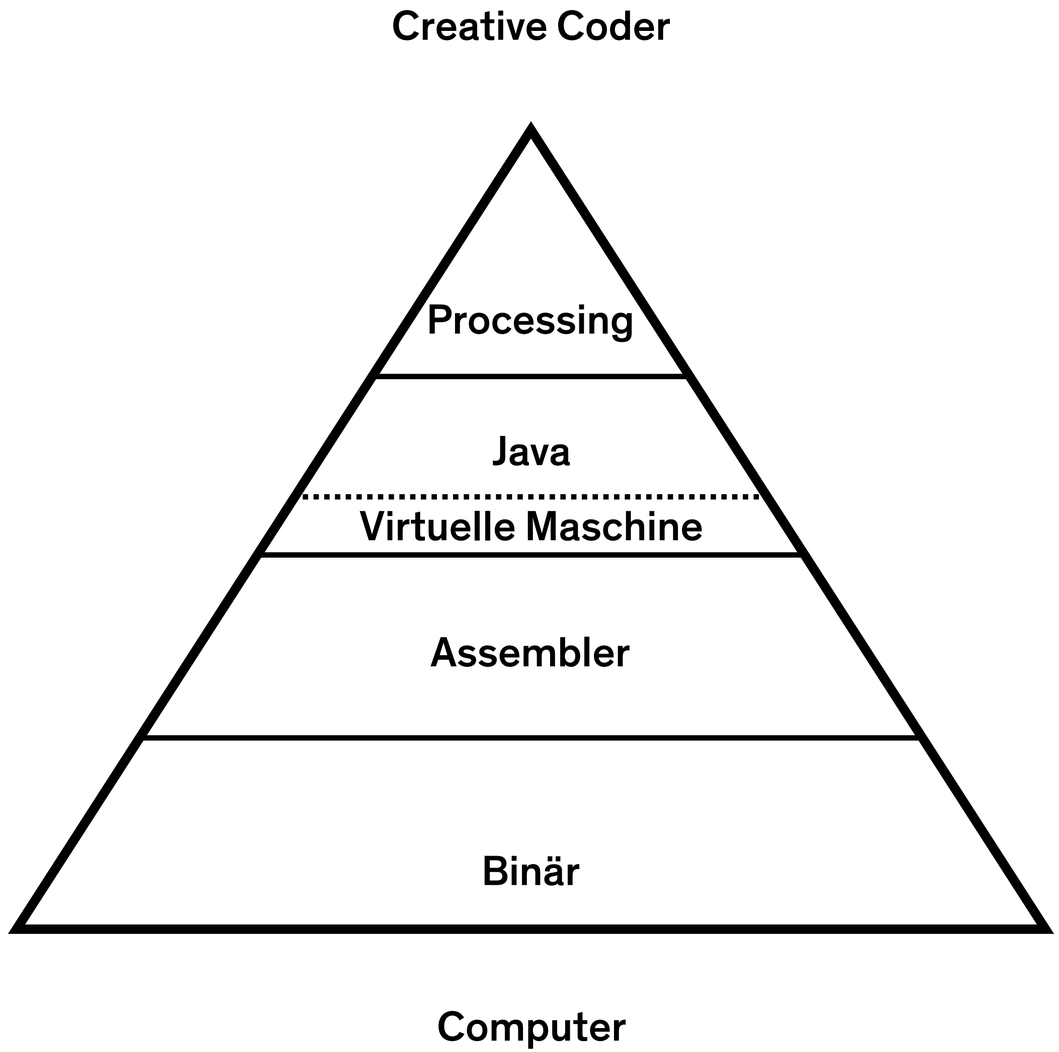
Stopp

Tabs / Reiter

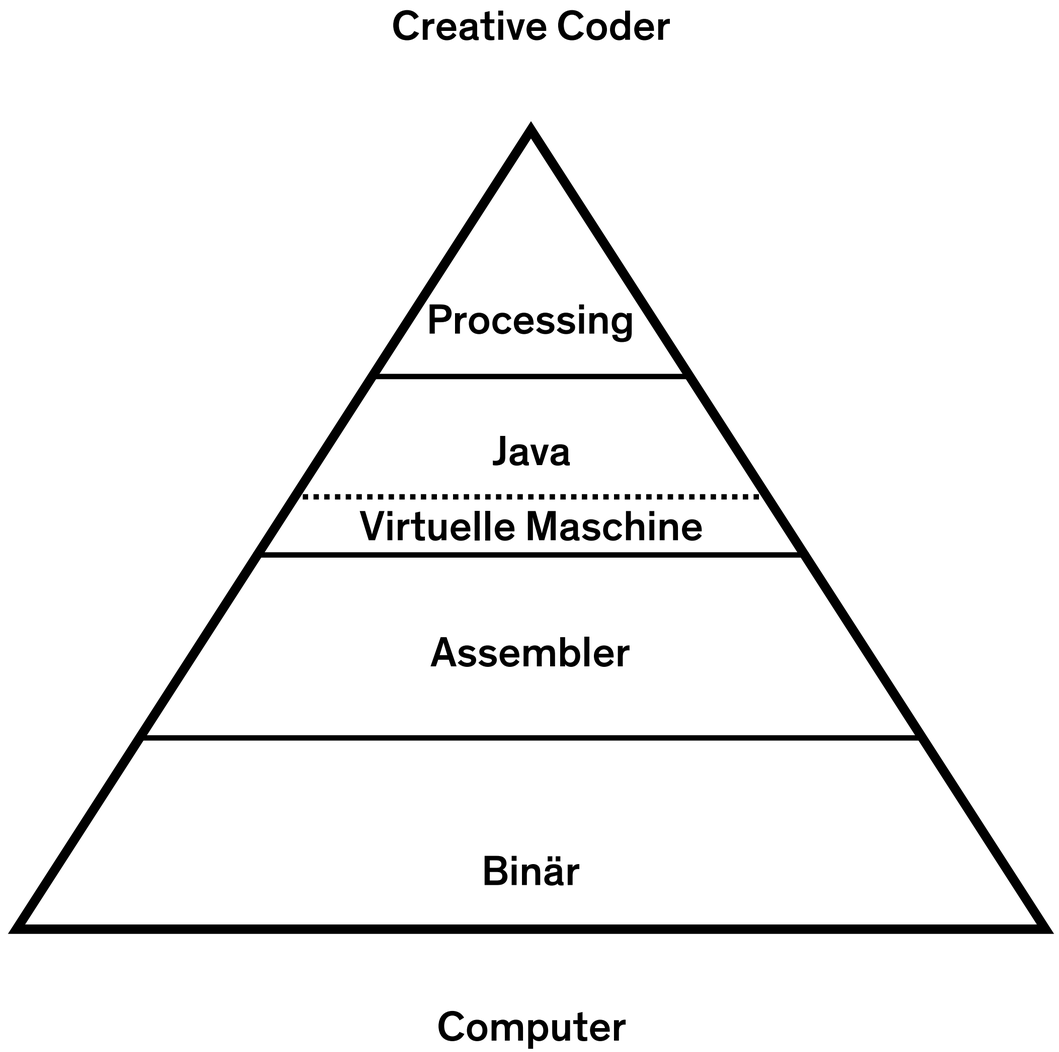
Modus (Java, Python, R…)

Debug (Fehlersuche)

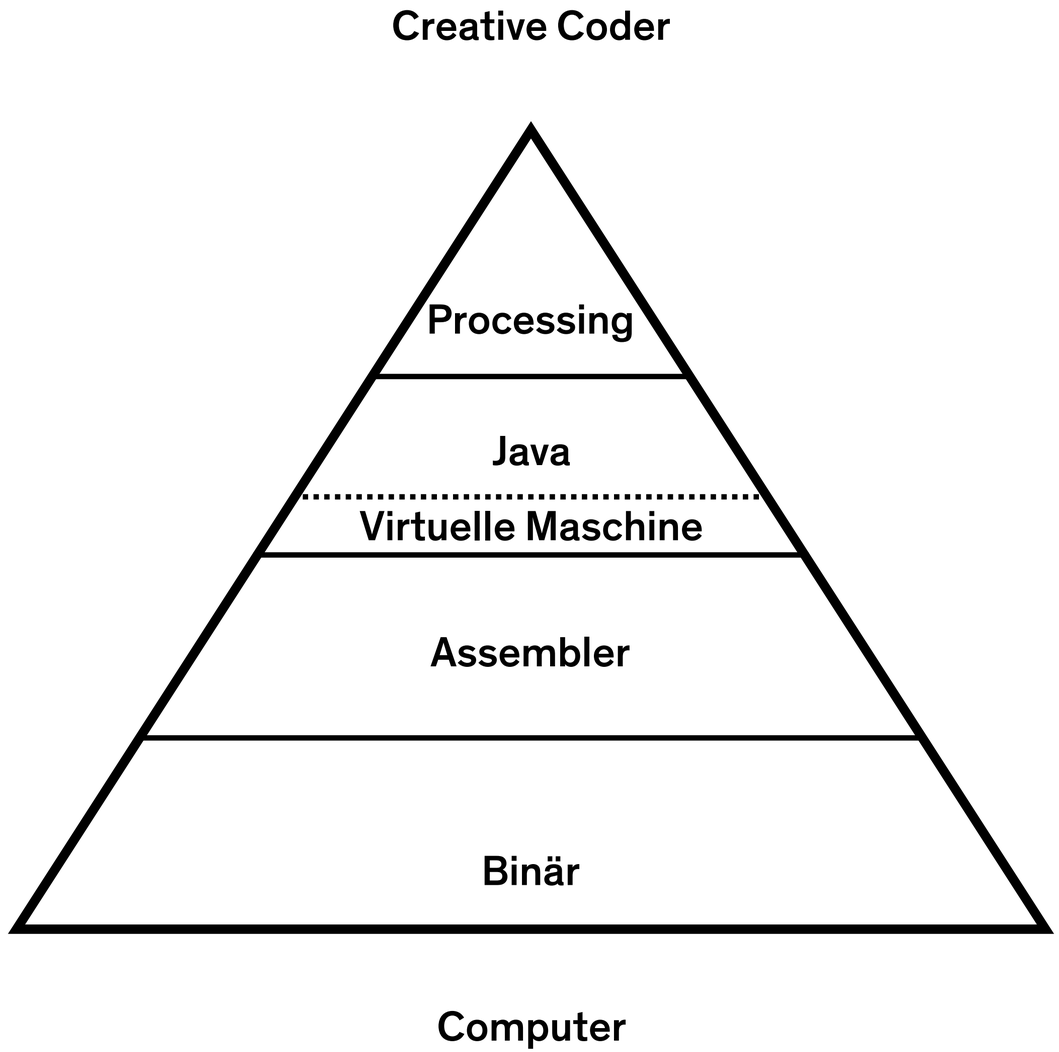
Konsolenausgabe



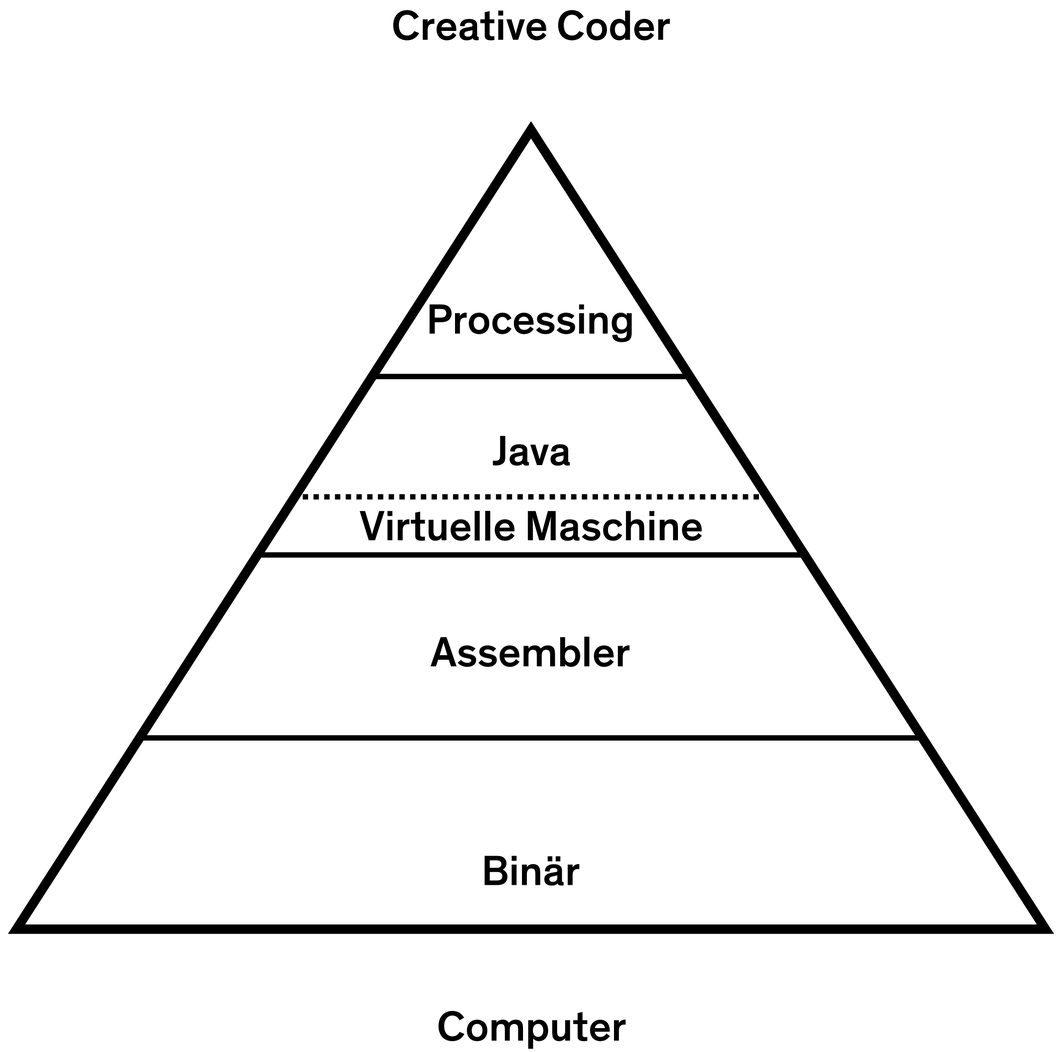
Processing IDE



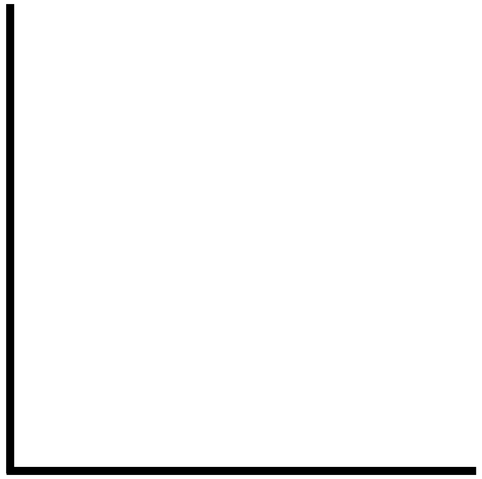
Processing IDE



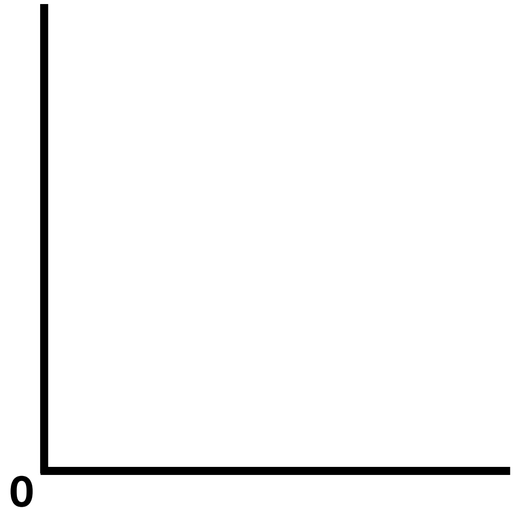
Processing IDE



Processing IDE

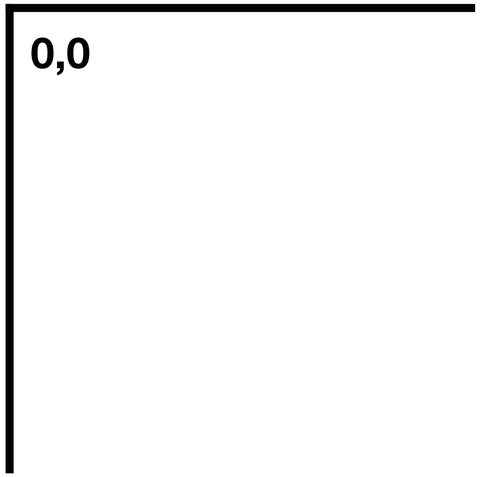


Orientierung

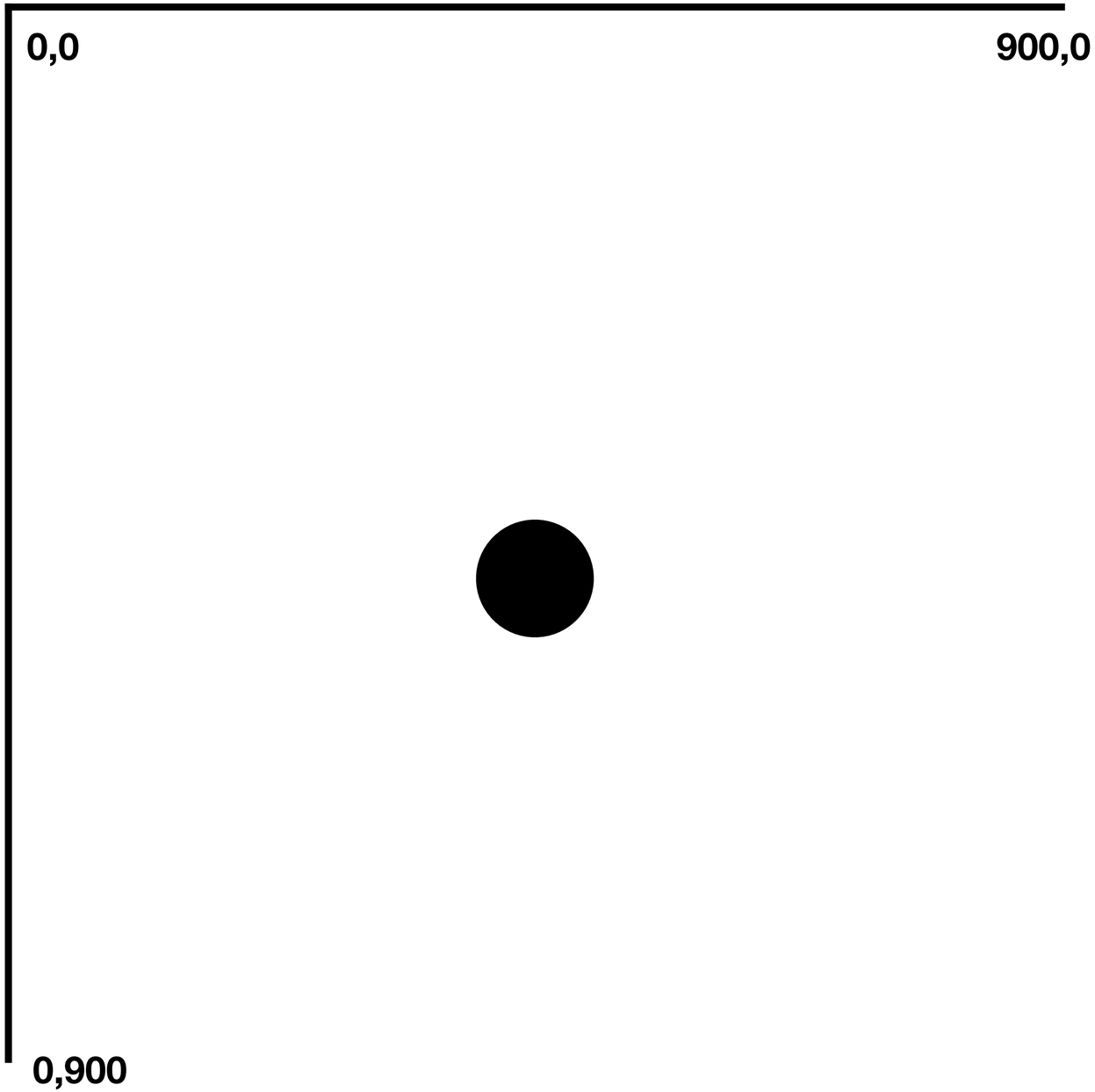


Orientierung

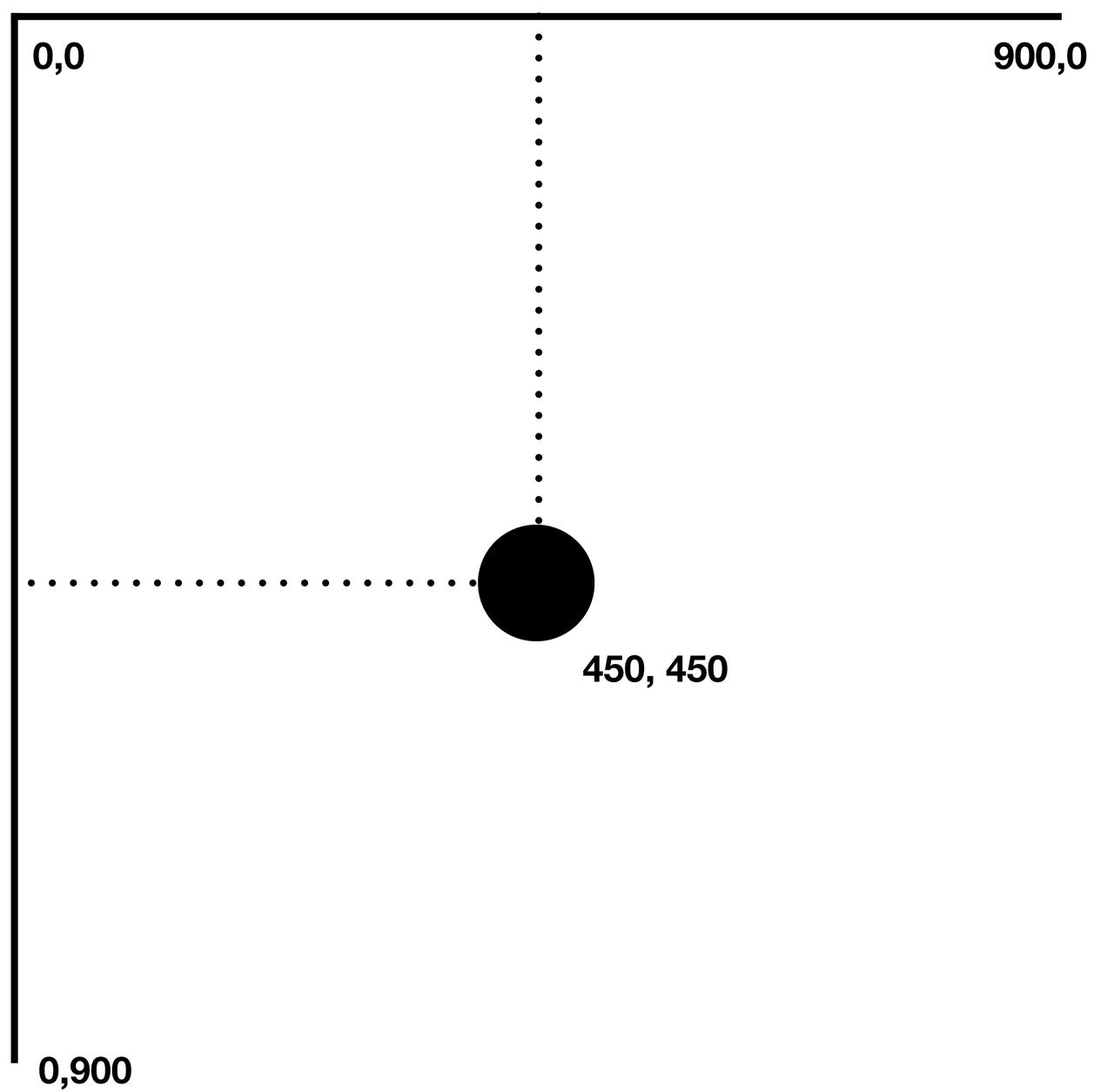
Orientierung



Orientierung



Orientierung



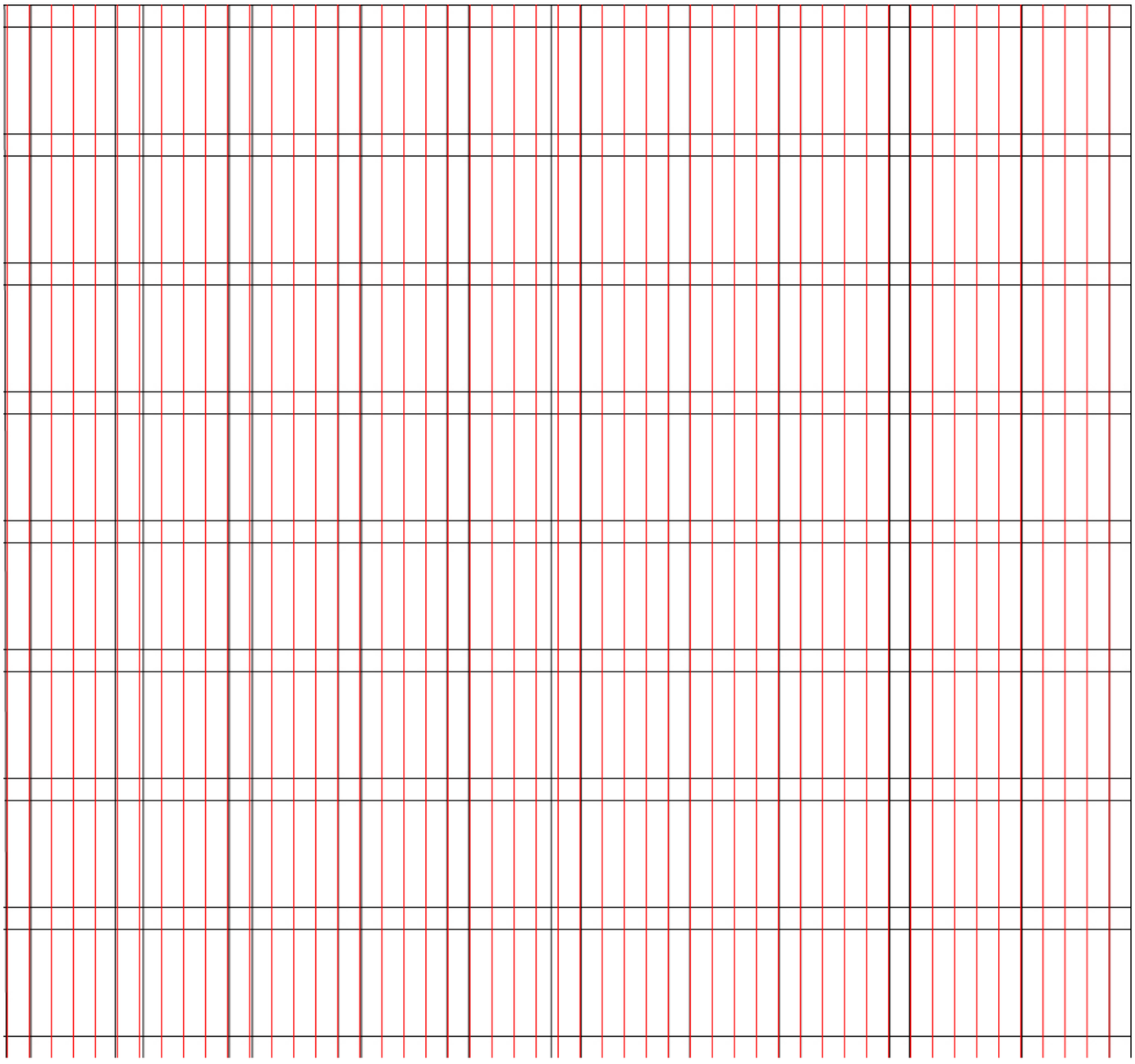
Orientierung

Das Raster

Das Raster

Creative Coding

Das Raster



Dimension

1D

****

1D

2D

****

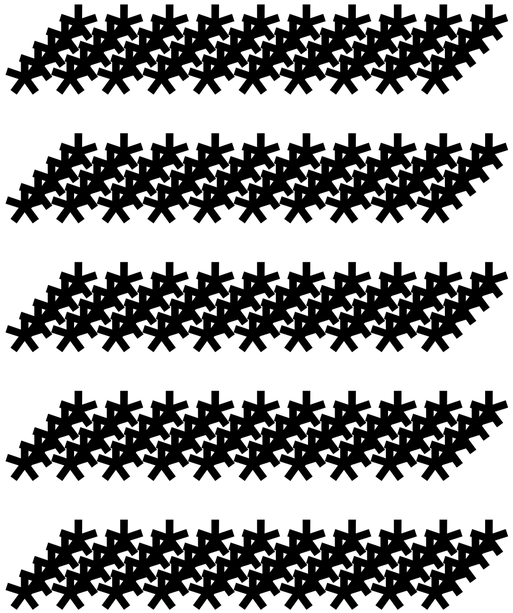
****

****

****

****

3D

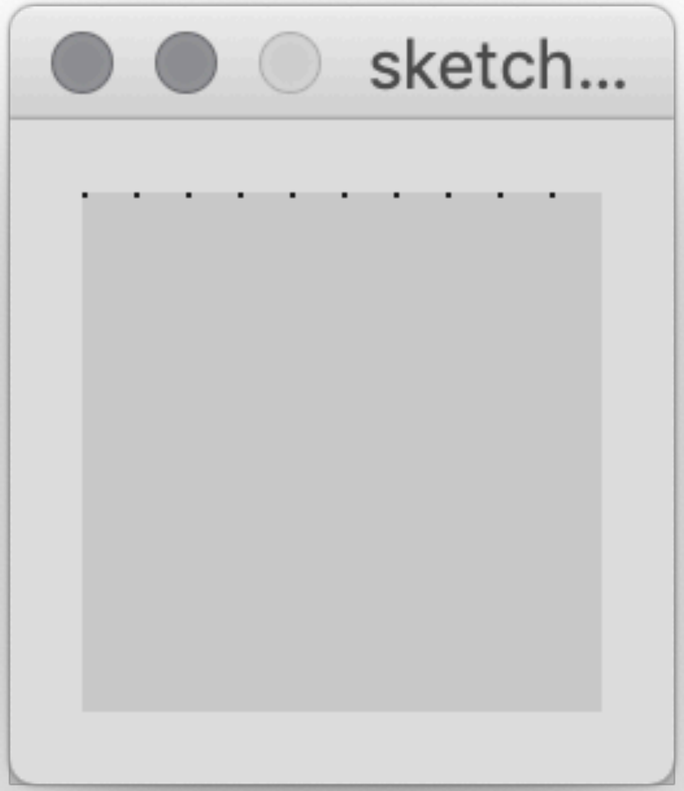


3D

```
for (int x = 0; x < 10; x++) {
    point(x * 10, 0);
}
```

Iteration 1D

```
for (int x = 0; x < 10; x++) {
    point(x * 10, 0);
}
```

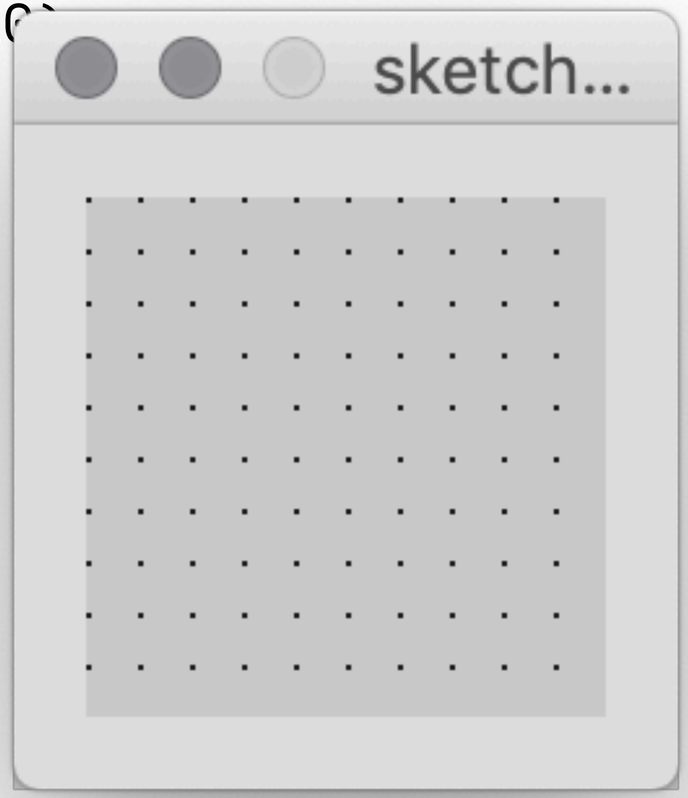


Iteration 1D

```
for (int y = 0; y < 10; y++) {
    for (int x = 0; x < 10; x++) {
        point(x * 10, y * 10);
    }
}
```

Iteration 2D

```
for (int y = 0; y < 10; y++) {
    for (int x = 0; x < 10; x++) {
        point(x * 10, y * 10);
    }
}
```

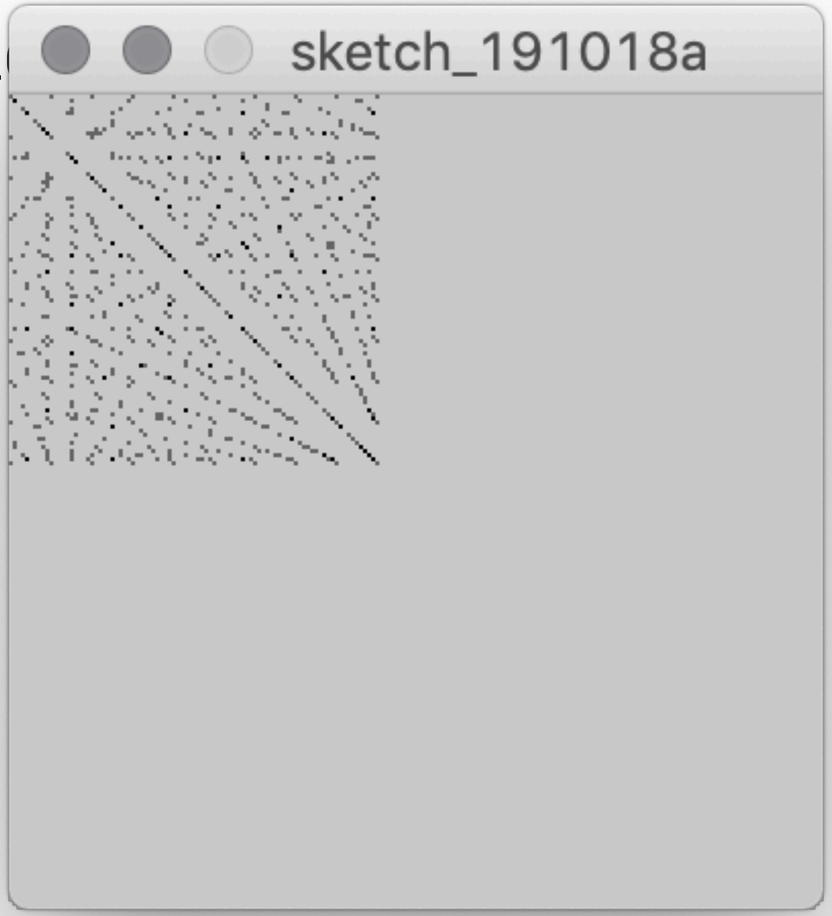


Iteration 2D

```
size(200, 200, P3D);
for (int z = 0; z < 10; z++) {
    for (int y = 0; y < 10; y++) {
        for (int x = 0; x < 10; x++) {
            point(x * 10, y * 10, z * 10);
        }
    }
}
```

Iteration 3D

```
size(200, 200, P3D);
for (int z = 0; z < 10; z++) {
    for (int y = 0; y < 1
        for (int x = 0; x
            point(x * 10,
        }
    }
}
```



Iteration 3D

Entwerft und realisiert ein regelmäßiges Raster, bestehend aus min. 3 × 3 Rasterelementen.
Jedes Rasterelement besteht aus min. 3 geometrischen Formen.
Zeichnet jedes Element einzeln.

Aufgabe 2 – Bis 25.10.2019

Dokumentiert die Ausgabe + Code.
Benutzt auch das Tutorium und die Online-Referenz

https://processing.org/reference/

Aufgabe 2 – Bis 25.10.2019

Folgende Befehle können benutzt werden:

- **arc()**
- **ellipse()**
- **line()**
- **point()**
- **quad()**
- **rect()**
- **triangle()**

Aufgabe 2 – Bis 25.10.2019

Folgende Attribute sollten auch mindestens einmal erprobt werden:

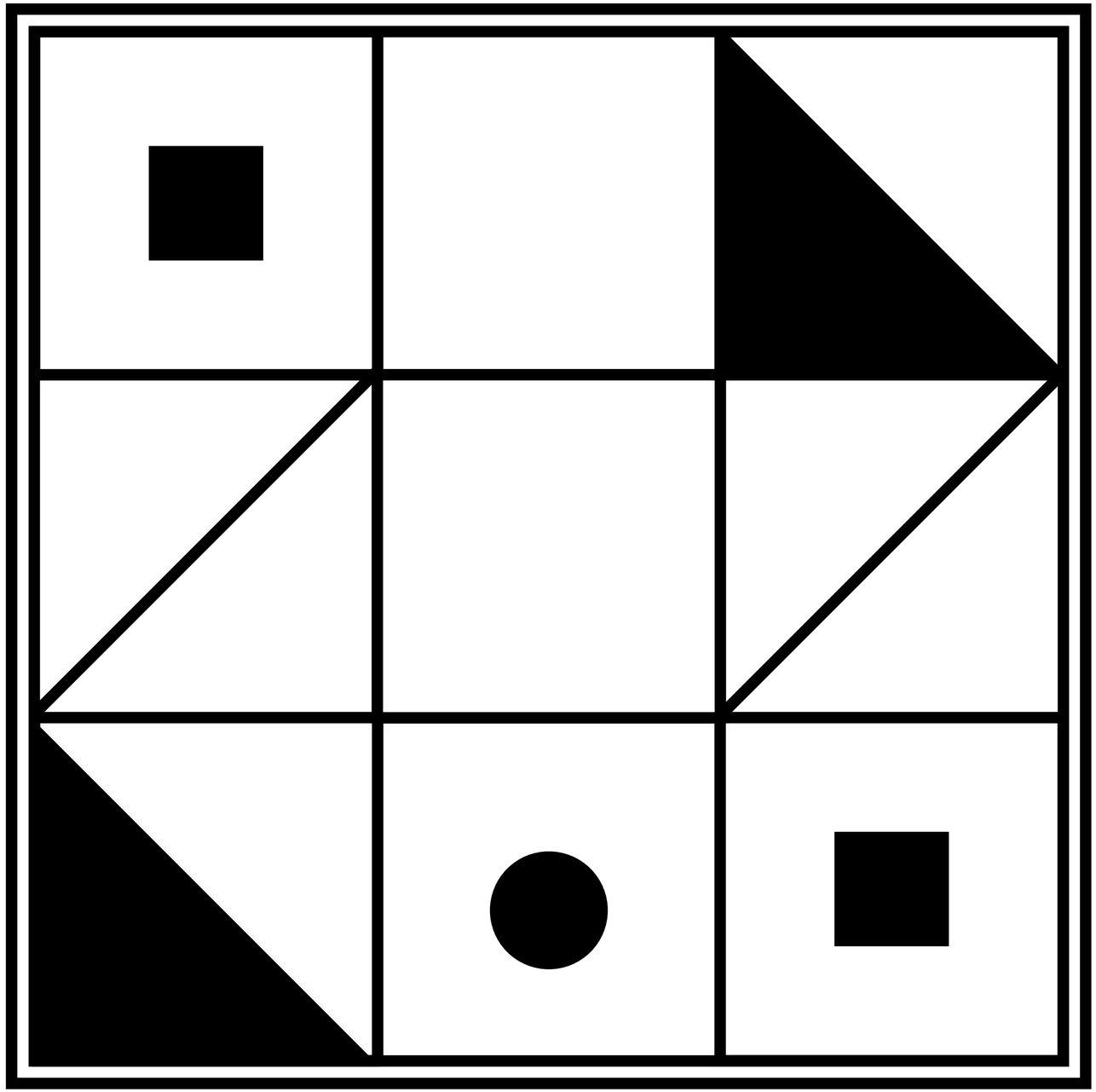
- **ellipseMode()**
- **rectMode()**
- **strokeCap()**
- **strokeJoin()**
- **strokeWeight()**

Aufgabe 2 – Bis 25.10.2019

Folgende Eigenschaften sind im weiteren Verlauf sehr nützlich:

- **background()**
- **clear()**
- **colorMode()**
- **fill()**
- **noFill()**
- **noStroke()**
- **stroke()**

Aufgabe 2 – Bis 25.10.2019



Aufgabe 2 – Bis 25.10.2019