Klasser & Objekter

Programmering-B

Thomas Birk Abildgaard

November 25, 2025

HTX Frederikshavn

Hvad er en klasse og et objekt?

Klasse: En skabelon eller "opskrift" for noget i virkeligheden. F.eks. en `Bold`.

Objekt: En konkret udgave af en klasse. F.eks. en bestemt bold med en farve og position.

En klasse indeholder:

- **Felter/variabler**: Beskriver objektets egenskaber (fx position, farve, fart).
- **Metoder** : Hvad objektet kan gøre (fx tegne sig selv, bevæge sig).
- **Constructor:** En speciel metode, der kører når vi laver et nyt objekt.

Hvorfor Klasser & Objekter

Hvorfor bruger vi klasser og objekter?

- Det gør vores kode mere overskuelig.
- Vi kan genbruge den samme kode til mange objekter.
- Det passer godt med hvordan vi forstår verden: Ting har egenskaber og kan gøre noget.

Eksempel på en simpel klasse i Processing

```
1  class Ball {
2    float x, y;       // position
3    float dx, dy;     // hastighed
4
5    Ball(float startX, float startY) {
6      x = startX;
7      y = startY;
8      dx = 2;
9      dy = 2;
10   }
11
12   void move() {
13     x += dx;
14     y += dy;
15   }
16
17   void display() {
18     circle(x, y, 20);
```

Hvad er en instans ?

Hvad betyder en instans?

En **instans** er et konkret **objekt**, som er lavet ud fra en klasse.

Sammenhæng:

- **Klassen** er en opskrift eller skabelon.
- **Objektet** er det færdige produkt.
- En **instans** er bare et andet ord for et objekt, som er skabt (instansieret) ud fra klassen.

Eksempel:

- Klassen hedder `Ball`.
- Når vi laver `Ball b = new Ball(100, 100);`, laver vi en **instans** af klassen `Ball`.
- Vi kan lave mange instanser (objekter) fra samme klasse.

Hvordan bruges klassen?

```
1  Ball b;  // deklarerer en variabel af typen Ball
2
3  void setup() {
4    size(400, 400);
5    b = new Ball(100, 100);  // opretter et objekt
6  }
7
8  void draw() {
9    background(255);
10   b.move();      // flyt bolden
11   b.display();   // tegn bolden
12 }
```

What is 'this' ?

Hvad betyder this?

this refererer til det **objekt**, som metoden bliver kaldt på.

Hvornår bruger man this?

- Når man vil henvise til objektets egne variabler (felter).
- Især nyttigt, hvis en parameter i en metode eller konstruktor har samme navn som feltet.

Eksempel:

```
1 class Ball {
2   float x, y;
3
4   // Constructor med parameter-navne der matcher
      felterne
5   Ball(float x, float y) {
6     this.x = x;  // this.x = feltet (venstre)
```

Kollision med kanterne af vinduet

Når en bold rammer kanten af vinduet, skal den "bounce" tilbage.

Hvordan gør vi det?

- Tjek om boldens position er uden for vinduets grænser.
- Vend retningen ved at ændre hastighedens fortegn.

Eksempel på metode i `Ball`-klassen:

```
void checkEdges() {
  if (x < 0 || x > width) {
    dx = -dx;
  }
  if (y < 0 || y > height) {
    dy = -dy;
  }
}
```

Hvordan bruges checkEdges()?

```
1 void draw() {
2   background(255);
3   b.move();
4   b.checkEdges();  // tjek for kollision med kanterne
5   b.display();
6 }
```

Resultat: Bolden "rammer" kanten og ændrer retning.

Kollision mellem to cirkler (bolde)

Vi kan også tjekke om to bolde rammer hinanden.

Hvordan?

- Beregn afstanden mellem boldenes centre.
- Hvis afstanden er mindre end summen af deres radier, så rammer de.

Eksempel på metode i `Ball`-klassen:

```
1 boolean checkCollision(Ball other) {
2   float distance = dist(x, y, other.x, other.y);
3   return distance < 20;  // 20 er diameteren af bolden
4 }
```

Hvordan kan vi reagere på kollision?

```
1  // I draw() eller en anden metode:
2  for (int i = 0; i < balls.length; i++) {
3    for (int j = i+1; j < balls.length; j++) {
4      if (balls[i].checkCollision(balls[j])) {
5        // Vend retning for begge bolde
6        balls[i].dx = -balls[i].dx;
7        balls[i].dy = -balls[i].dy;
8        balls[j].dx = -balls[j].dx;
9        balls[j].dy = -balls[j].dy;
10     }
11   }
12 }
```

Note: Dette er en simpel model for kollision uden fysik.