# Code and Objects

# Object-oriented Programming

Vi skaber en ”skabelon” for et objekt

- F.eks. Biler

Objekter kan have forskellige kendetegn

- F.eks. Biler har en farve, har 4 hjul, har forlygter…

Objekter kan have forskellige egenskaber

- F.eks. Biler kan køre, kan standse, kan blinke…

# En klasse: lav en skabelon

```
class Bil {
    constructor (color, speed, xpos, ypos, size) {
        this.color=color;
        this.speed=speed;
        … osv.
    }

  drive () {
    this.xpos=this.xpos + this.speed;
    …
  }
}
```

→ Klassen gives navn.

→ Argumenterne i objektet ”matches” med variabler i funktioner udført på objekterne.

→ Således defineres funktionerne.

# Definer dit Objekt

Hvis du vil have flere af et objekt (flere biler, flere bolde, el. lign.) giver det mening at lave et array:

```
var bil= [ ];

function setup()  {
    bil [0]=new Bil(color(255,0,0),10,10,100,20);
    bil [1]=new Bil(color(0,255,0), 5, 10,400,20);
    bil [2]=new Bil(___...

}
```

# Kald på dit objekt

Hvis du har en lang array af objekter er det nemt at udføre funktioner på dem med et for-loop:

```
function draw () {
   for (var i = 0; i < bil.length; i++) {
     bil[i].drive();
   }
}
```

# Winnie’s Code

https://rawgit.com/AUAP/AP2018/master/class05/sketch05/index.html

```
let car = [];
let button;
let bg;

function preload() {
  bg = loadImage("data/road.jpg");
}

function setup() {
 createCanvas(windowWidth, windowHeight);
 button = createButton('add+1');
 button.style('color','#555555');
 button.mousePressed(add);

 car[0] = new Car(color(255,0,0), 10, 10, 100, 20); //create/construct a new object instance
 car[1] = new Car(color(0,0,255), 15, 20, 300, 10);

}
```

```
22  function draw() {
23   background(bg);
24   button.position(width/2,5);
25
26   for (let i = 0; i <car.length; i++) {
27    car[i].drive();
28    car[i].display();
29   }
30  }
31
32  function add() {
33    append(car,  new Car(color(random(155,255)), random(2,10), random(10,20), random(15,500), random(30)));
34  }
```

```
36  //create a class: template/blueprint of objects
37  class Car {
38    constructor(getcolor, speed, xpos, ypos, size) { //initalize the objects
39      this.getcolor = getcolor;
40      this.speed = speed;
41      this.pos = new createVector(xpos, ypos);  //check this feature: https://p5js.org/reference/#/p5/createVector
42      this.size = size;
43    }
44
45    drive() {
46      this.pos.x+=this.speed;  //this.pos.x = this.pos.x + this.speed;
47      if (this.pos.x > width) {
48         this.pos.x = 0;
49      }
50    }
51
52   display() {
53     noStroke();
54     fill(this.getcolor);
55     rect(this.pos.x,this.pos.y,this.size,this.size);
56   }
57
58  }
```

# Discussion - Instructortimer

Hvad vil I gerne bruge instruktortimerne om onsdagen til?

På fredag er sidste shut-up-and-code med instruktor til stede.

- Vil I gerne gå igennem den nye syntax?
- Vil I hellere bruge tiden til at diskutere jeres miniEx'es fra den foregående uge med hinanden? (Der skal stadig gives peer-feedback på github)
- Vil I hellere diskutere den konceptuelle del af den kommende miniEx?
- Vil I hellere bare lave jeres miniEx opgaver?

# MiniEx4 - Discussion

Sæt jer sammen i grupper, der ikke er dem I ellers er i gruppe med.

Præsenter jeres miniEx fra sidste uge for hinanden.

- Forklar din vision.
- Vær så præcis som mulig med hvorfor du bruger lige præcis den syntax det sted igennem koden.
- Giv gerne feedback og forslag til hinanden.

# Feedback

Peer-feedback er en del af den aktive, regelmæssige og tilfredsstillende deltagelse i undervisningen, derfor SKAL i give feedback HVER uge til 2 andre for at blive indstillet til eksamen!

# MiniEx5

- Make sure you have read the text by Roger Lee
- Think of a simple **game** that you want to design and implement, what are the **objects** required? What are their **properties and behaviors**? At the most basic level, you need to **use class-based object oriented approach** to design your game components that can exhibit certain behaviors. If you can master objects and classes, you ***may*** further work on a **mini game** (with basic rules).
- Upload your game or game object sketch to your own Github account under a folder called **mini_ex5**.
- Create a readme file (README.md) and upload to the same mini_ex5 directory
- Provide peer-feedback to 2 of your classmates on their works by creating "issues" on his/her github corresponding repository. Write with the issue title "Feedback on mini_ex(?) by (YOUR FULL NAME)"

# readMe

The readme file should contain the followings:

A screenshot/animated gif/video of your program

A URL link to your program and run on a browser.

Describe **how does your game/game object works**?

Describe **how you program the objects** and their related attributes and methods in your game.

Based on Shiftman's videos, Lee's text and in-class lecture**, what are the characteristics** of **object-oriented programming**?

Extend/connect your game project to **wider digital culture context,** can you think of a digital example and describe how complex details and operations are being **abstracted and encapsulated?**

# Peer-feedback

- First you describe what is the work, what are the elements in the work? like what you have seen, what you have experienced and what syntax he/she has used.
- What is the emphasis? What does the work express? What does this work say or mean to you? How would you interpret the work?
- Do you like this program, and Why? and which aspect do you like the most?
- Provide suggestion for improvement or expansion of the program/thoughts

# Go Code!