



Objects

Datamatiker-b

2020-09-15 Lyngby



Agenda

- Opfølgning på i går (Functions)
- Eksamens info
- Chapter 08: Objects
- Exercises
- Runde af og dagen I morgen
- Slutter kl 16.00 I dag



Recap

- Ofte laver man en løsning man tror virker. Og når man så skal bruge den finder man ud af den ikke kunne alt det som den skulle alligevel.
- Det har vi to effektive værktøjer til: refactoring og tests.
- Refactoring er meget relevant i forhold til gårsdagens lektion.
- Tests tager vi en anden god gang 😊



Recap

Refactoring er noget vi kan gøre ved gammelt software, som ikke har 'fulgt med' den resterende løsning.

Det er også noget vi kan bruge ved en sprit ny metode vi lige har skrevet.

Første udkast af en metode jeg skriver er netop blot dét – et udkast. Når jeg så har funktionaliteten på plads, så går jeg det igennem igen og kigger på faldgruber, performance, læsbarhed, etc.

Noget af det sværeste som 'ny' programmør er at få noget skrevet på et blankt stykke 'papir'.



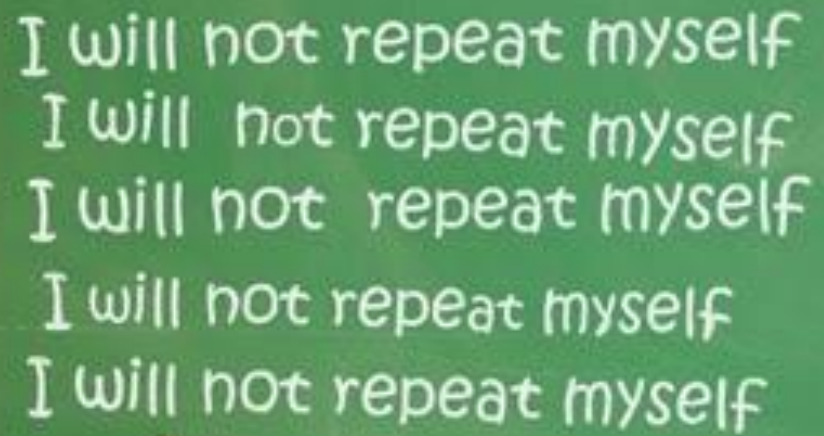
Recap

DRY



Recap

DRY



I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself



Recap: Functions

- Parameters passing

```
drawCar(80,175,40,color(100,0,100));
```

*passing
parameters*

```
void drawCar(int x, int y, int thesize, color C) {  
    // CODE BODY  
}
```



Recap: Functions

- Returning a value

```
int x = sum(5,6,8);  
int y = sum(8,9,10) * 2;  
int z = sum(x,y,40);  
line(100,100,110,sum(x,y,z));
```

```
int answer = sum(5,10,32);
```

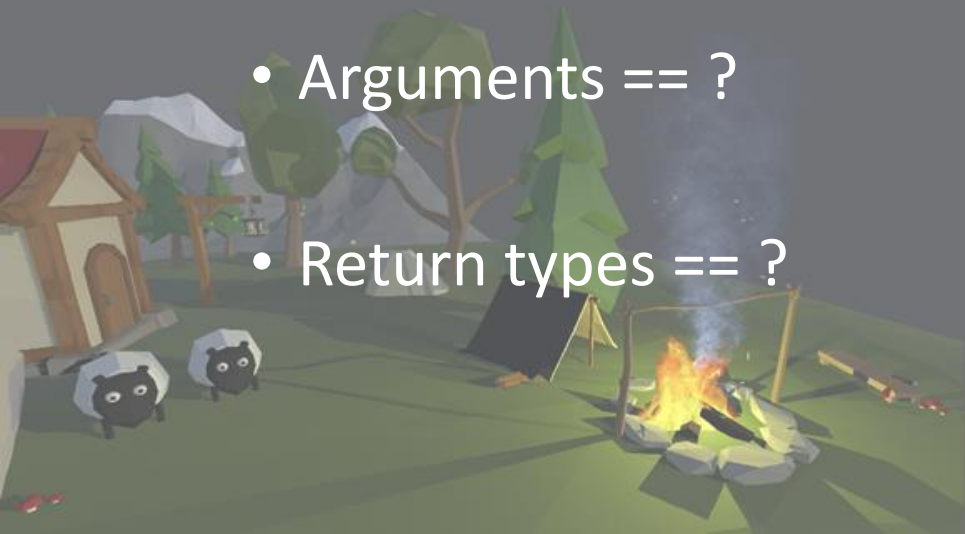
```
int sum(int a, int b, int c) {  
    int total = a + b + c;  
    return total;  
}
```

total is returned



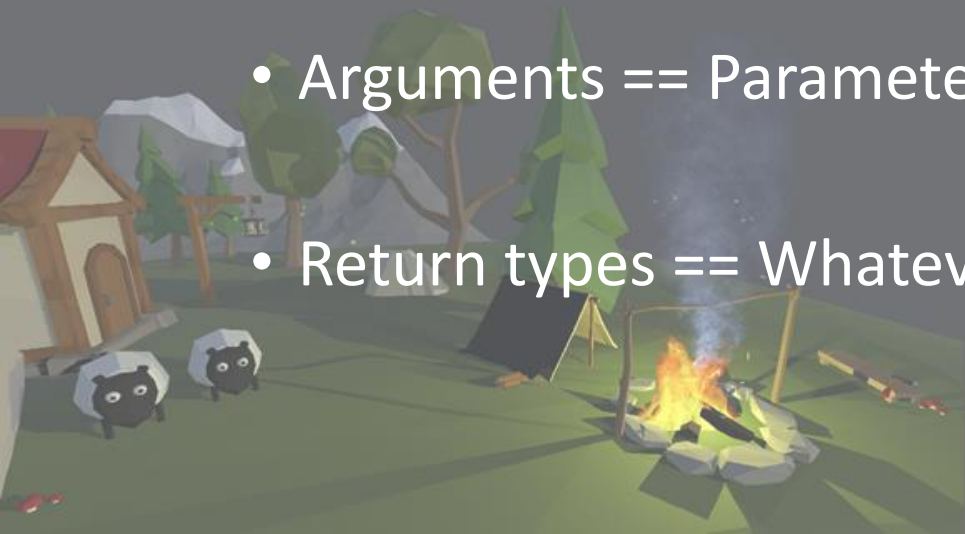
Recap: Functions

- Methods == ?
- Functions == ?
- Parameter == ?
- Arguments == ?
- Return types == ?



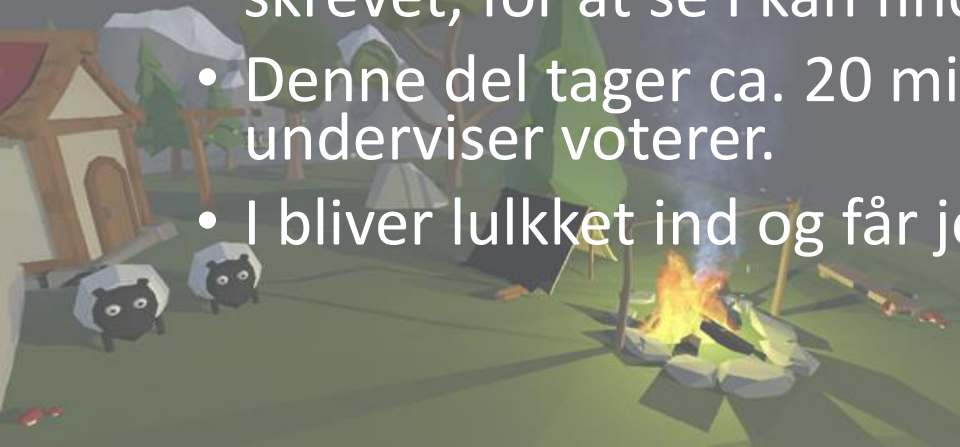
Recap: Functions

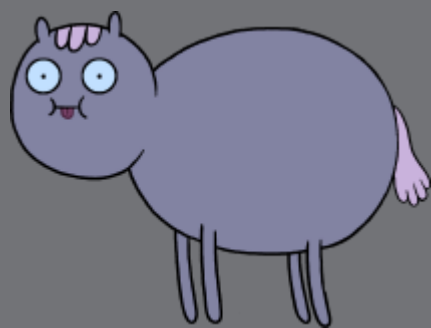
- Methods == Functions
- Functions == Methods
- Parameter == Argument
- Arguments == Parameters
- Return types == Whatever data type the function returns.



Eksamens info

- Man 'trækker' en simple opgave (RNG)
- Opgaverne er indenfor de emner som er vist på listen – **MEN** kun grundlæggende- og OOP basics (ingen databaser eller komplicerede design opgaver).
- Ca. 30 minutter til at løse opgaven.
- Så bliver I lukket ind igen og deler skærm og fortæller om hvad I lige har lavet.
- Så bliver der stillet nogle yderligere spørgsmål til den kode I lige har skrevet, for at se I kan finde ud af at live-kode.
- Denne del tager ca. 20 minutter, hvorefter I lukkes ud. Censor og underviser voterer.
- I bliver lukket ind og får jeres karakter.





Pause

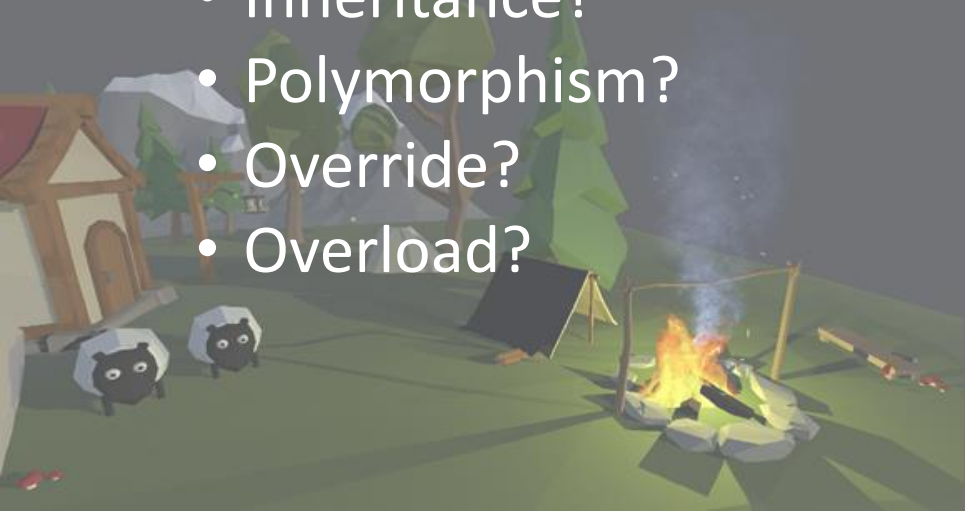


Objects

- Data and functionality, together at last.
- What is an object?
- What is a class?
- Writing your own classes.
- Creating your own objects .
- Processing “ tabs. ”
- Constructor arguments
- Objects are data types too!
- OOP Zoog

Objects

- Inheritance?
- Main?
- Static?
- Information hiding?
- Aggregation? Composition?
- Interfaces?
- Inheritance?
- Polymorphism?
- Override?
- Overload?



Data and functionality, together at last.

Human data

- Height.
- Weight.
- Gender.
- Eye color.
- Hair color.

Human functions

- Sleep.
- Wake up.
- Eat.
- Ride some form of transportation.



CREATE HOOMAN IN UNITY AND SHOW TWO
DIFFERENT ONES WITH DIFFERENT DATA



What is an object?

What is a class?

Think of a cookie cutter. A cookie cutter makes cookies, but it is not a cookie itself. The cookie cutter is the *class*, the cookies are the *objects*.



Using an Object

```
Car myCar;
```

An object in *Processing*.

```
void setup() {  
    myCar = new Car();  
}
```

```
void draw() {  
    background(0);  
    myCar.move();  
    myCar.display();  
}
```



Defining an object



```
class Car {
```

Define a class below the rest of the program.

```
    color c;  
    float xpos;  
    float ypos;  
    float xspeed;
```

Variables.

```
    Car() {  
        c = color(255);  
        xpos = width/2;  
        ypos = height/2;  
        xspeed = 1;  
    }
```

A constructor.

```
    void display() {  
        // The car is just a square  
        rectMode(CENTER);  
        fill(c);  
        rect(xpos, ypos, 20, 10);  
    }
```

Function.

```
    void move() {  
        xpos = xpos + xspeed;  
        if (xpos > width) {  
            xpos = 0;  
        }  
    }  
}
```

Function.

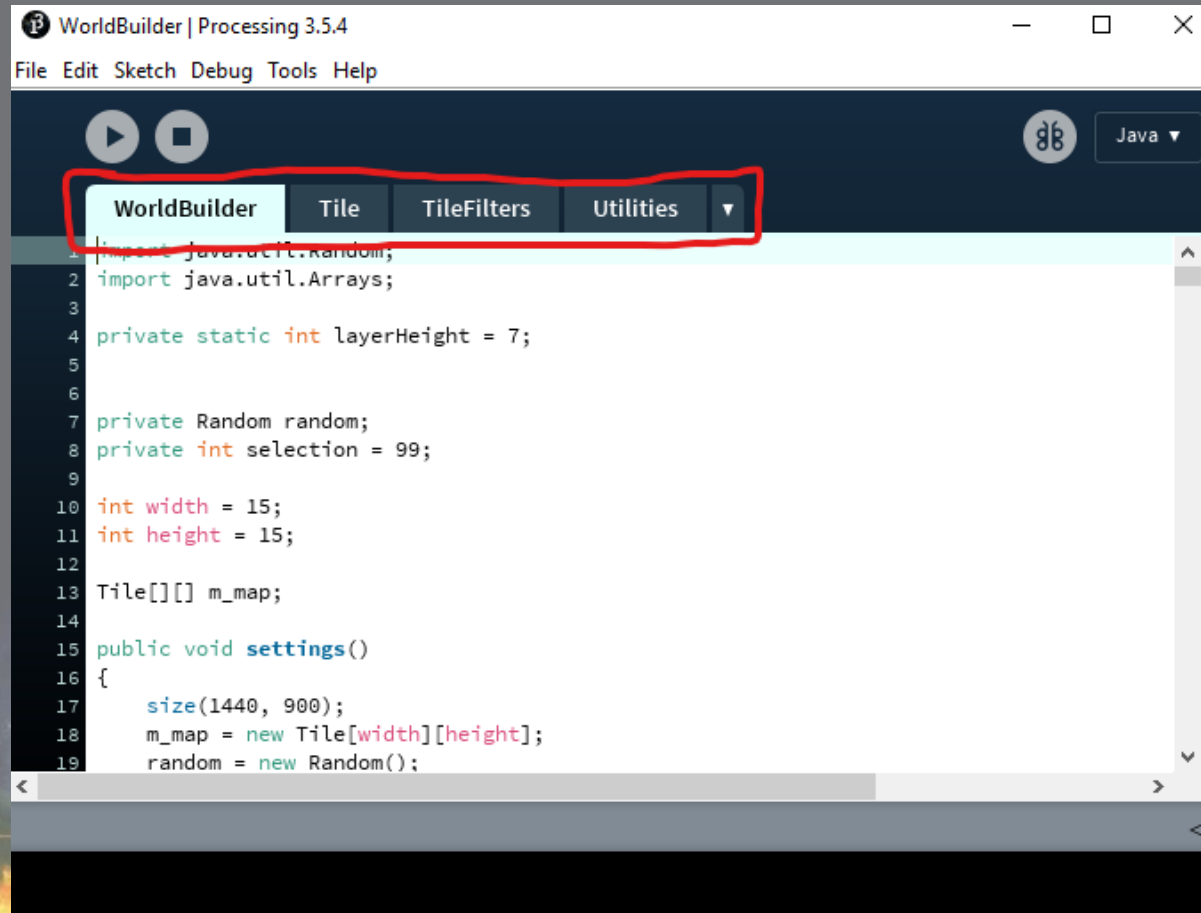
“New tab” vs Classes vs Nested Classes

A Class Is a New Block of Code!

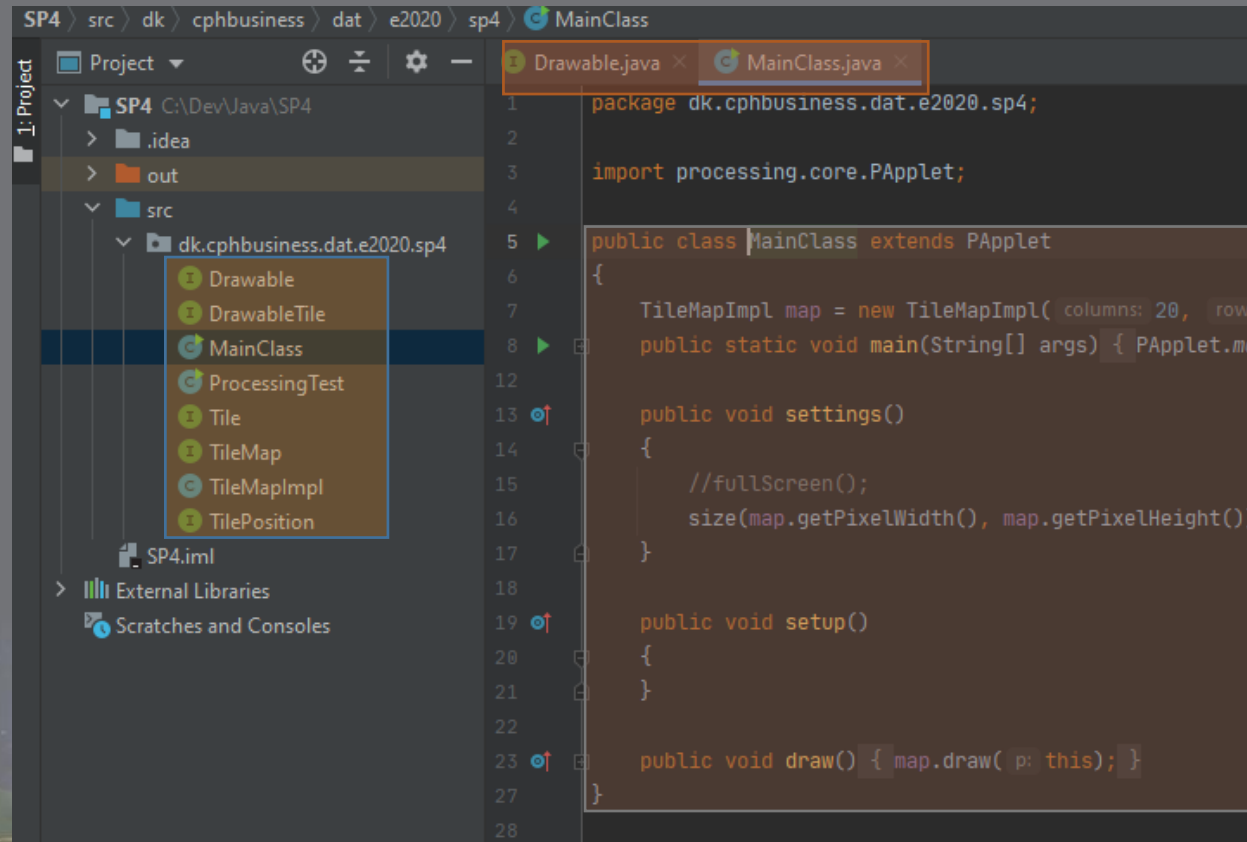
```
void setup() {  
  
}  
void draw() {  
  
}  
class Car {  
  
}
```



“New tab” vs Classes vs Nested Classes



“New tab” vs Classes vs Nested Classes



Instance of an object??

- Instance ?
- Object ?
- Reference ?
- New keyword ?




Instance of an object??

- Instance == Object

- Object == Instance

- Reference



```
Computer c= new Computer()
```

- New keyword

- Hver gang vi bruger keyword'et "new" laver vi en ny instance/object ud fra en type/class



Constructor arguments

Reminder: at kalde en function med parametre:

```
drawCar(80, 175, 40, color(100, 0, 100));
```

*passing
parameters*

```
void drawCar(int x, int y, int thesize, color C) {  
    // CODE BODY  
}
```



Constructor arguments

```
// Creating two car objects  
Car myCar1 = new Car();  
Car myCar2 = new Car();
```

Hvad hvis den ene bil skal være rød og den anden sort?

Diesel || Benzin || El ?

Topfart på 160 km/t vs 220 km/t ??



Constructor arguments

```
Car(color tempC, float tempXpos, float tempYpos, float tempXspeed) {  
    c = tempC;  
    xpos = tempXpos;  
    ypos = tempYpos;  
    xspeed = tempXspeed;  
}
```

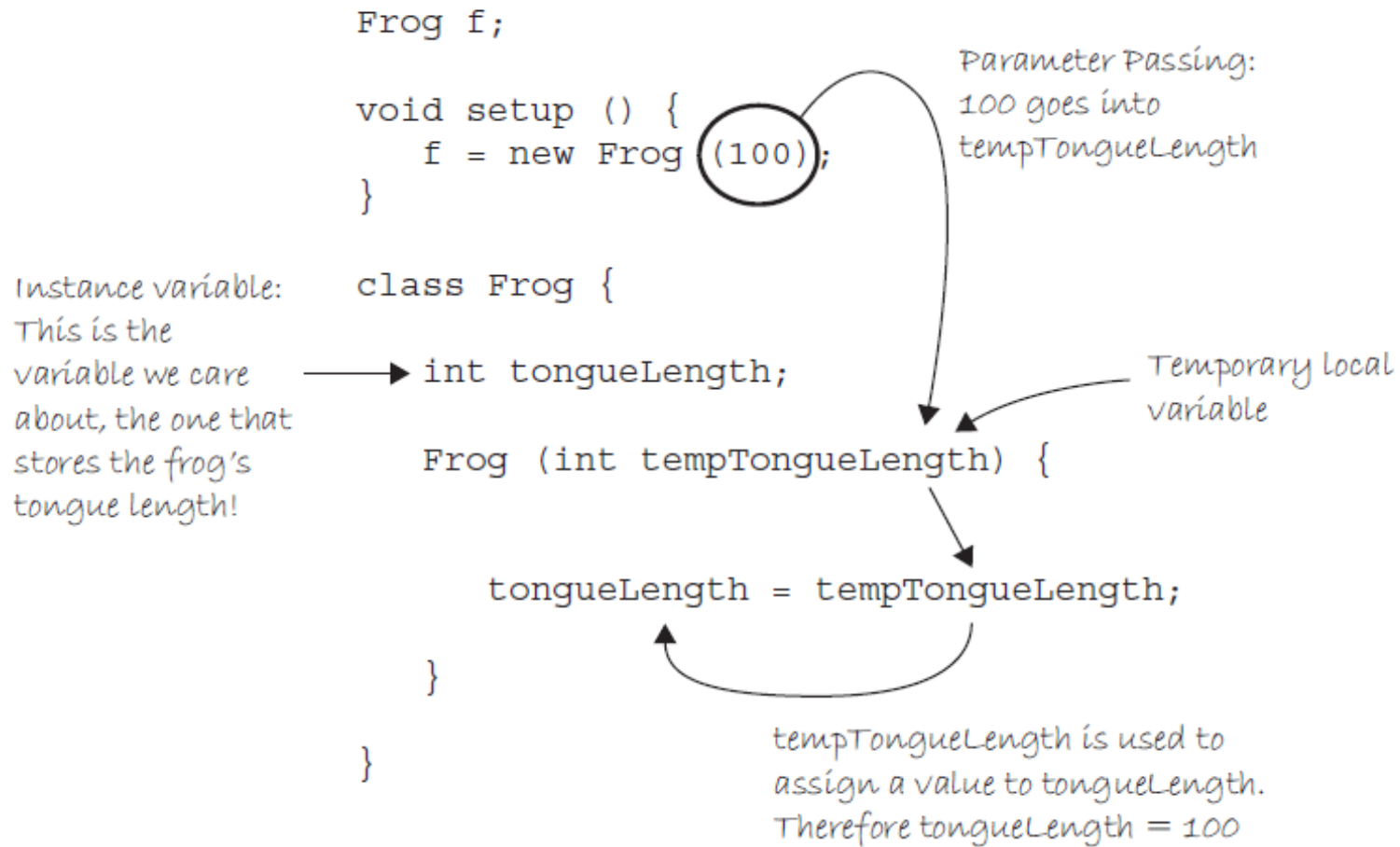


Constructor arguments

- Why the temp variables in the constructor?



Constructor arguments



Translation: Make a new frog with a tongue length of 100.

fig. 8.5

Constructor arguments

Example 8-2: Two Car objects

```
Car myCar1;  
Car myCar2;  
  
void setup() {  
  size(200,200);  
  
  myCar1 = new Car(color(255,0,0),0,100,2);  
  myCar2 = new Car(color(0,0,255),0,10,1);  
}  
  
void draw() {  
  background(255);
```

Two objects!

Parameters go inside the parentheses when the object is constructed.

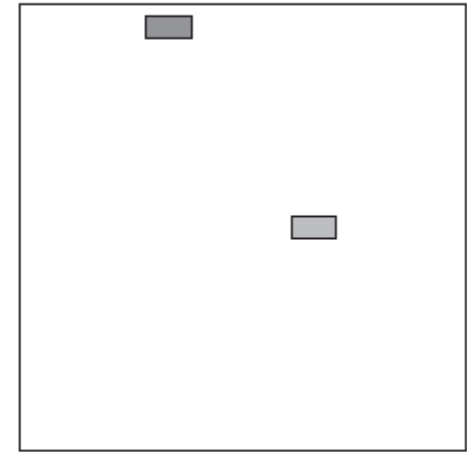
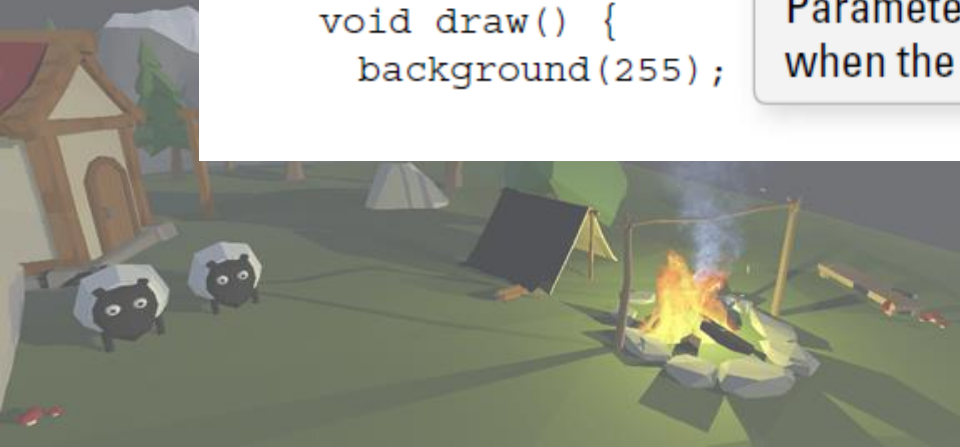


fig. 8.6



Objects are data types too!

```
class PlaceSetting {  
  
    Fork fork;  
    Spoon spoon;  
  
    PlaceSetting() {  
        fork = new Fork();  
        spoon = new Spoon();  
    }  
}
```

A class can include other objects among its variables.



Static vs non static

- Vi kan ikke lave nye instances/objects af ting der er static.
- Mere om dette den 5. Oktober, hvor det indgår som et emne i undervisningen.



Pause



Exercise 8-1

- Consider a car as an object.
- Car data:
 - - ?
- Car Methods:
 - - ?



Example

- Have a look at the CarDealership example.



Gravity Example 8-5

- We're to have a look at example 8-5 together.
- Question:
 - How would you go about adding a 100 balls?
 - How would you display and update the 100 balls?



Exercises



Exercises

- Create a Human class (as a tab) and let it have the data and functions from the start of the chapter.

- In the functions, simply have it print the name of the human.
e.g.:

```
void sleep() {  
    • println("Sleep");  
}
```

- Create 2 humans from the main tab and print some information about each human.

Human data

- Height.
- Weight.
- Gender.
- Eye color.
- Hair color.

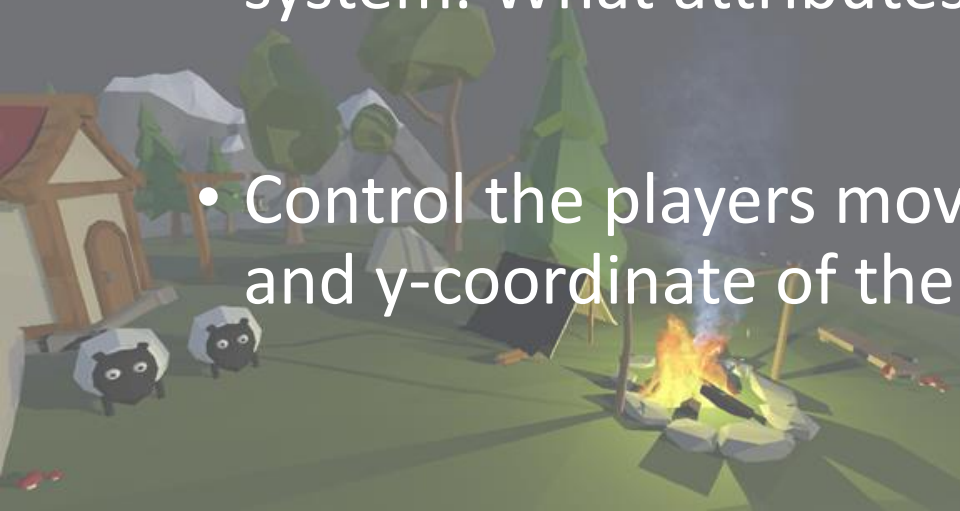
Human functions

- Sleep.
- Wake up.
- Eat.
- Ride some form of transportation.



Gridbased game exercise

- Create a new sketch where you draw a 2D grid as we have done before. (double for loop)
- Create a new class in a new tab, called “Player”
- The goal is to get the player to move around in our grid x/y coordinate system. What attributes and methods should the player have?
- Control the players movement in the main sketch by setting the x- and y-coordinate of the player object and drawing it from draw()



More exercises

- <https://codingBat.com/java>
 - No real object exercises

<https://www.hackerrank.com/domains/java>

Object exercises er niveauet over hvad vi har lært i dag, da de bruger koncepter som inheritance, abstraction, method overriding, etc. – Disse ting kommer vi til.



Runde af

- I morgen: zoom kl 09.00
- <https://cphbusiness.zoom.us/j/8718786820>
- Læs: **Chapter 09 Arrays**

