













0001















Into the Void

Going into the abyss of OOP



Power of OOP

Think hard, Code once

De vier
OOP basis principes

#1

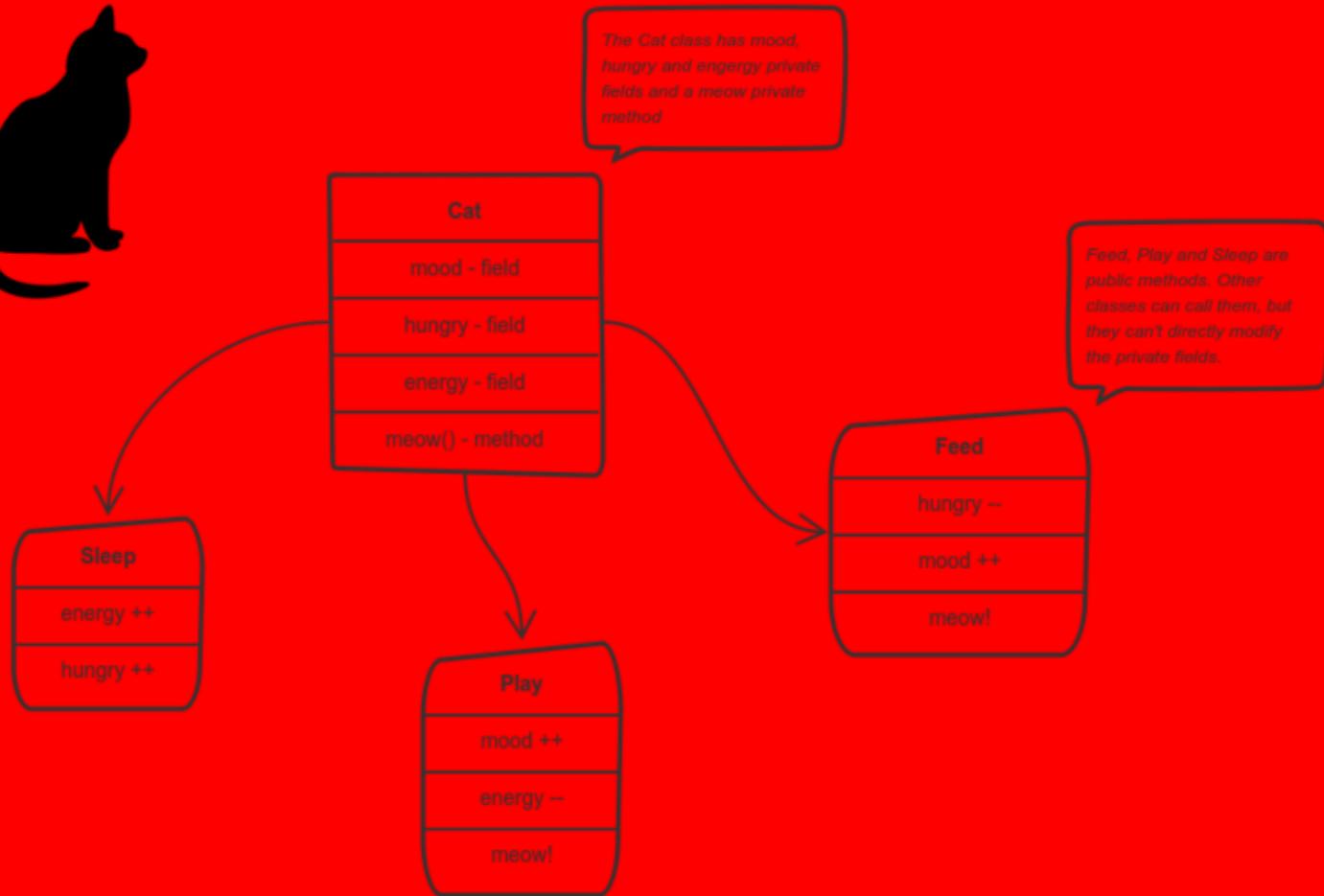
#2

#3

#4

Encapsulation

- Een object publiceert alleen die methodes die gebruikt kunnen worden.
- De rest schermt hij af.
- Data members zijn verborgen
- Toegang is gereguleerd
- Er zijn geen concessies nodig



Abstraction

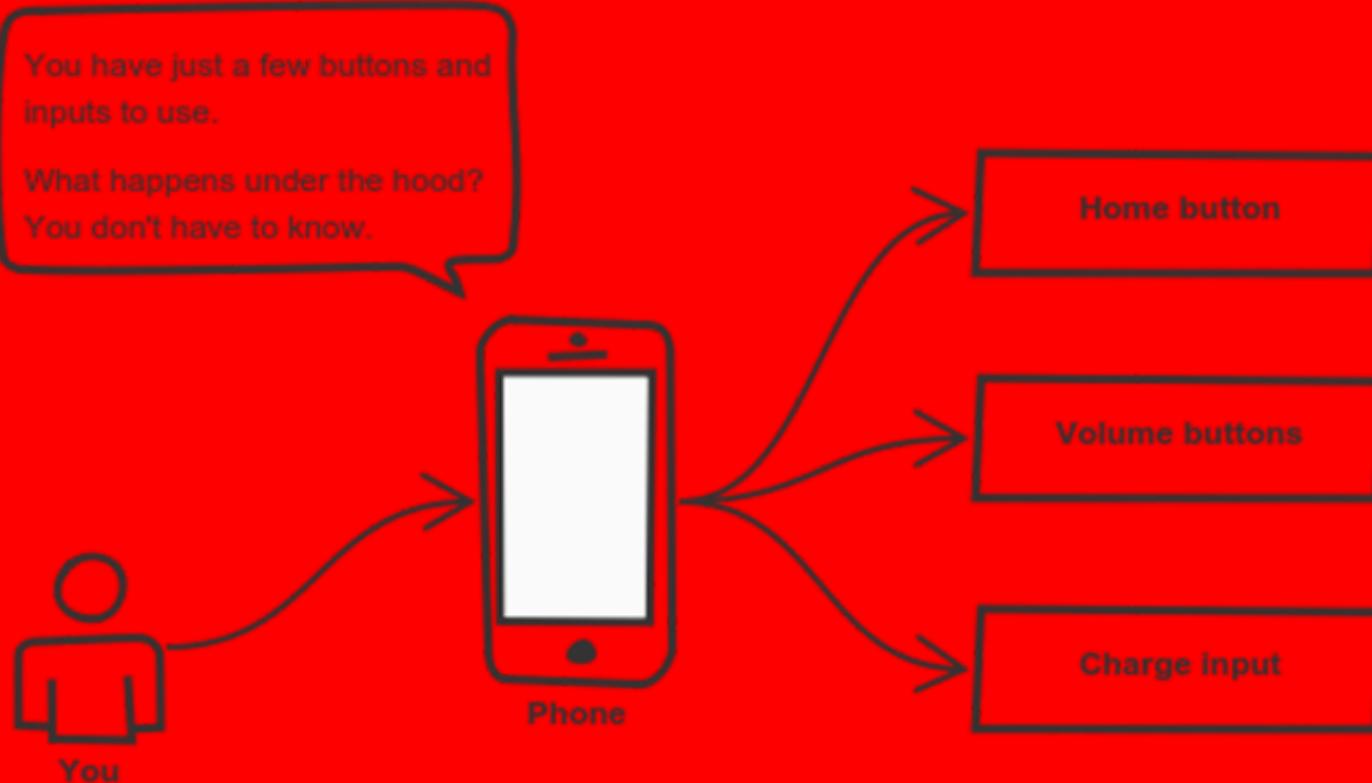
Het afschermen van alle zaken die voor de buiten wereld niet belangrijk zijn

Alle methodes die beschikbaar zijn

- Het hoororgaan
- De ogen
- De Mond

Achter de schermen gebeurt de magie en effect is indirect.

Je kan niet een plaatje rechtstreeks in de hersens krijgen zonder de ogen te gebruiken. (publieke interface)



You have just a few buttons and inputs to use.

What happens under the hood?
You don't have to know.



Home button

Volume buttons

Charge input

Inheritance

Als meerdere objecten gemeenschappelijke eigenschappen bezitten.

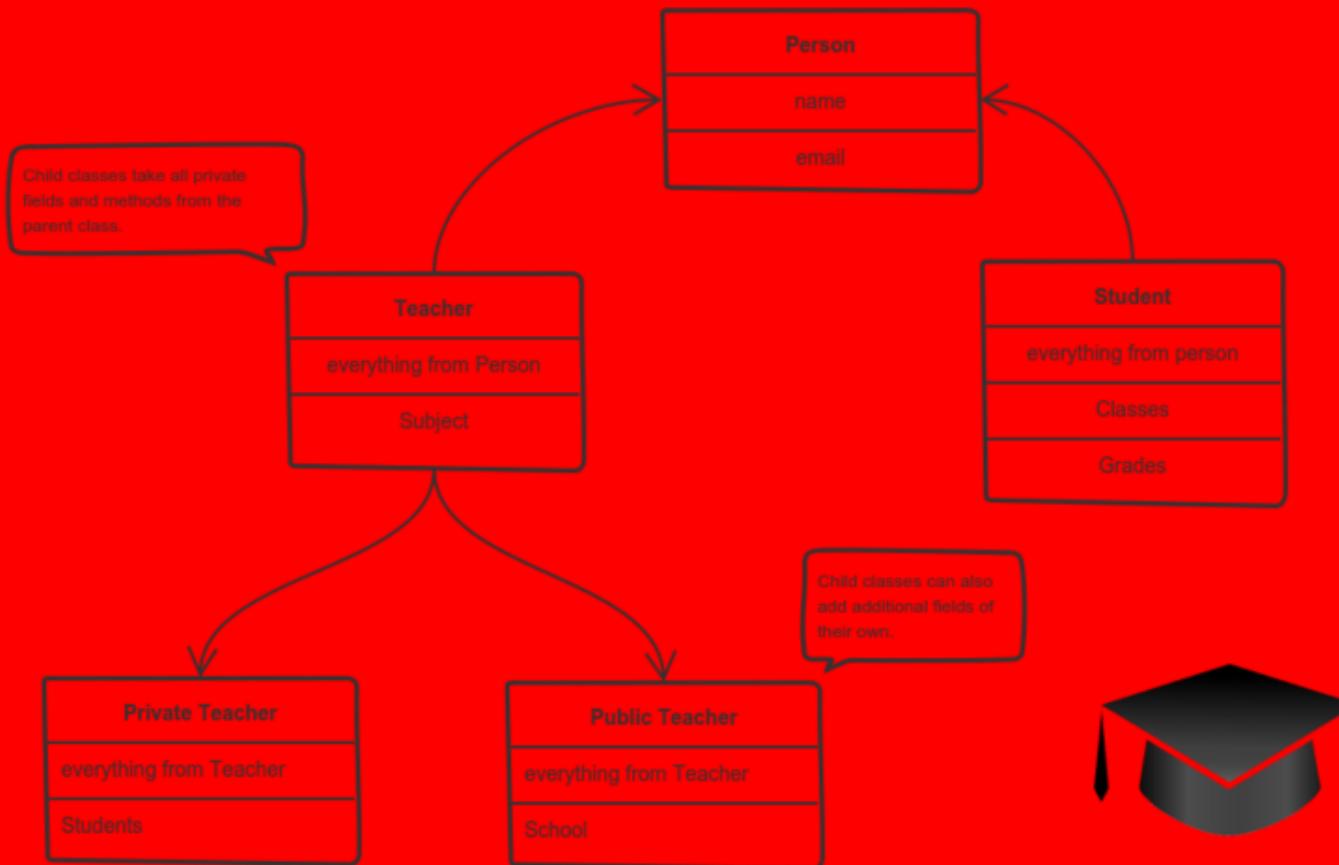
Maar alleen als er een "is een" relatie bestaat

- Een man is een Mens
- Een vrouw is een Mens
- Een programmeur is een Mens (ja echt :)

Is er een "heeft een" of "kan" relatie dan geldt dit niet

- Een hond kan lopen (net als een mens maar is geen mens)
- Een Kat kan ademen (net als een vis maar is geen vis)

Mogelijk is er een abstractie te bedenken waar ze wel samen komen ?

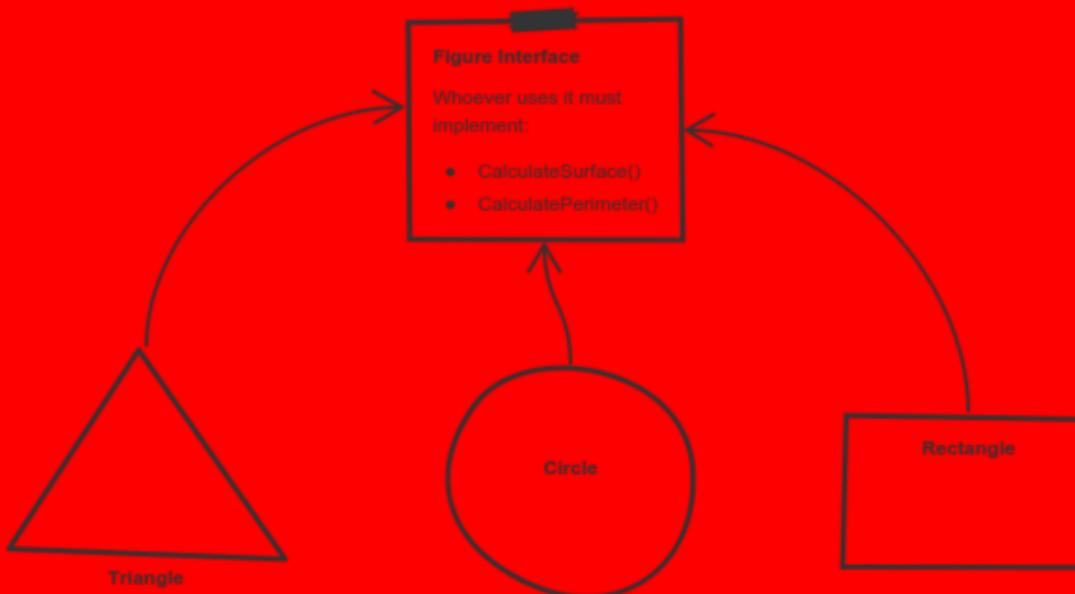


Polymorphism

De mogelijkheid om verschillende objecten gelijk te behandelen op basis van hun gemeenschappelijke eigenschappen.

- Het spreken tegen een man of een vrouw is gelijk.
- Het zien van een landschap door hond, mens, mier.

Echter is de verwerking verschillend per object.



Triangle, Circle and Rectangle inherit the Figure interface or abstract class.

They implement their own versions of CalculateSurface() and CalculatePerimeter().

They can be used in a mixed collection of Figures.



Voorbeeld van de Power of OOP





Structuur

Base
controllers
dataObjects
entities
helpers
views

001

010

011

100

101

110

src\base

Hierin komen alle basis klassen (base classes)

src\controllers

Hierin komen de controllers te staan. De controllers besturen de logica.

src\dataObjects

Hierin komen de object klassen (data containers) welke geen directe invloed hebben op de game.

src\entities

voor dit project herkennen we de speler, de asteroids en alle onderdelen welke interactie in de game kunnen hebben.

We werken naar een State machine implementatie toe. Elke entiteit komt in deze map en per entiteit een submap voor de verschillende states.

bv.

src\entities\player

src\entities\asteroid

src\helpers

in deze map stoppen we speciale klassen
die ons gaan helpen maar niet specifiek
voor deze game zijn.

src\views

Hierin plaatsen we de views. Denk aan de verschillende schermen welke we al hebben:

- Start scherm (MenuView)
- Level scherm (GameView)
- Titel scherm (ScoresView)



Refactor

Aan de slag met de code



Base

Help

View

src\base\viewBase.ts

allereerst maken we een baseclass waarin
we bedenken wat een scherm kan.

Wat zal elk scherm gelijk hebben ?

src\baselineViewBase.ts

allereerst maken we een baseclass waarin we bedenken wat een scherm kan.

Wat zal elk scherm gelijk hebben ?

- een canvas om op te tekenen
- een context van de canvas
- afhandeling van onClick. (zie fout in vb.)
- een methode om te tekenen. (Render)

dit hebben ze allemaal gelijk maar de implementatie zal wel verschillend zijn.

src\helpers\CanvasHelper.ts

vanuit de game.ts >> maakt het ons niet uit welk scherm het is. we weten nu hoe we IEDER scherm kunnen aanspreken

Random

src\helpers\CanvasHelper.ts

vanuit de game.ts >> maakt het ons niet uit welk scherm het is. we weten nu hoe we IEDER scherm kunnen aanspreken

- writeImageToCanvas
- writeTextToCanvas
- writeButtonToCanvas

Random

src\helpers\CanvasHelper.ts

vanuit de game.ts >> maakt het ons niet uit welk scherm het is. we weten nu hoe we IEDER scherm kunnen aanspreken

- writeImageToCanvas
- writeTextToCanvas
- writeButtonToCanvas

- RegisterOnClick
- Clear
- GetCenter
- GetHeight
- GetWidth

Random

src\helpers\MathHelper.ts

```
class MathHelper {  
  
    /**  
     * Renders a random number between min and max  
     * @param {number} min - minimal time  
     * @param {number} max - maximal time  
     */  
    public static randomNumber(min: number, max: number): number {  
        return Math.round(Math.random() * (max - min) + min);  
    }  
}
```

src\views\MenuView.ts

src\views\GameView.ts

src\views\Highscores.ts

```
class MenuView extends ViewBase
{
    public constructor(aCanvas : HTMLCanvasElement,
        aChangeViewCallback : (aNewView : ViewBase) => void ) {
        super(aCanvas,aChangeViewCallback);
    }

    protected HandleClick(X: number, Y: number): void {
    }

    protected RenderScreen(): void {
    }
}
```



Bringing it home



Can you Follow ?



Question Time?

Thank Your Audience!

Goodbye Now!

