

# Object georiënteerd programmeren

```
43 <body <?php body_class()></div>
44 <div id="fb-root"></div>
45 <script>(function(d, s, id) {
46     var js, fjs = d.getElementsByTagName(s)[0];
47     if (d.getElementById(id)) return;
48     js = d.createElement(s); js.id = id;
49     js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.6&appId=200644463628215";
50     fjs.parentNode.insertBefore(js, fjs);
51 }(document, 'script', 'facebook-jssdk'));</script>
52 <div id="page" class="site">
53     <a class="skip-link screen-reader-text" href="#content"><?php esc_html_e( 'Skip to content', 'urduube' );></a>
54     <div id="header" class="site-header">
55         <div class="site-branding">
56             <div class="navBtn pull-left">
57                 <?php if ( is_home() && $xpanel['homepage-style'] == 1 ) { ?>
58                     <a href="#" id="openMenu"><?php echo esc_html( 'Menu' );></a>
59                     <?php if ( ! is_user_logged_in() ) { ?>
60                         <a href="#" id="openMenu2"><?php echo esc_html( 'Log In' );></a>
61                     <?php } ?>
62                 </div>
63                 <div class="logo pull-left">
64                     <a href="#"><?php echo esc_url( home_url() );></a>
65                     
66                 </div>
67             </div>
68             <div class="search-box hidden-xs hidden-sm pull-left ml-10">
69                 <?php get_search_form(); ?>
70             </div>
71             <div class="submit-btn hidden-xs hidden-sm pull-left ml-10">
72                 <a href="#"><?php echo get_page_link( $xpanel['submit-link'] );></a>
73             </div>
74             <div class="user-info pull-right mr-10">
75                 <?php
76                 if ( is_user_logged_in() ) {
```

# Inhoud

1. Objecten, attributen en methoden, klassen
2. Superklassen en overerving

```
43 <body <?php body_class()></div>
44 <div id="fb-root"></div>
45 <script>(function(d, s, id) {
46     var js, fjs = d.getElementsByTagName(s)[0];
47     if (d.getElementById(id)) return;
48     js = d.createElement(s); js.id = id;
49     js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.6&appId=2980444362801";
50     fjs.parentNode.insertBefore(js, fjs);
51 })(document, "script", "facebook-jssdk");</script>
52 <div id="page" class="site">
53     <div id="main-link screen-reader-text" href="#content" class="visually-hidden">
54         <header id="masthead" class="site-header">
55             <div class="site-branding">
56                 <div class="navBtn pull-left">
57                     <?php if(is_home() && $xpanel['homepage-style'] == 1) { ?>
58                         <a href="#" id="openMenu"><i class="fa fa-bars fa-3x"></i></a>
59                     <?php } else { ?>
60                         <a href="#" id="openMenu2"><i class="fa fa-bars fa-3x"></i></a>
61                     <?php } ?>
62                 </div>
63                 <div class="logo pull-left">
64                     <a href="<?php echo esc_url( home_url() ) ?>">
65                         
67                 </div>
68                 <div class="search-box hidden-xs hidden-sm pull-left ml-10">
69                     <?php get_search_form(); ?>
70                 </div>
71                 <div class="submit-btn hidden-xs hidden-sm pull-left ml-10">
72                     <a href="<?php echo get_page_link($xpanel['submit-link']) ?>" class="header-submit-btn"
73                 </div>
74                 <div class="user-info pull-right mr-10">
75                     <?php
76                     if ( is_user_logged_in() ) {
```



# Objecten, klassen attributen en methoden

# Variabelen en datatypen

- Javascript bepaalt tijdens het uitvoeren welk soort informatie in een variabele wordt bewaart. Dit heet het datatype

```
var i = 0;
```

number

```
var gameOver = true;
```

boolean

```
var startTekst = "Welkom bij de game";
```

string

# Variabelen en datatypen

- Dit kàn maar moet je NOOIT doen  
(je krijgt een beerput vol rare fouten)

```
var mijnVar = 0;
```

```
mijnVar = "Welkom bij de game";
```

```
mijnVar = true;
```

number

string

boolean

## Logische eenheden

- In je code heb je vaak stukjes **informatie** en / of **functionaliteit** die bij elkaar horen.
- bijvoorbeeld:
  - de **x- en y- waarde** van een bal, evt. met **horizontale en verticale snelheden**, samen met de code om de bal een stukje te verplaatsen.
  - evenzo van een kogel, auto, speler, enz enz
  - de **positie, titel en grootte** van een 'knop', met daarbij de code die uitgevoerd wordt als je op de knop klikt

# Logische eenheden

- JavaScript kent daarvoor objecten.
- Verzameling van waarden met een label

```
var knopA = { x : 30,  
              y : 100,  
              breedte : 200,  
              hoogte : 50,  
              titel : "Niet klikken"  
            };  
rect(knopA.x, knopA.y, knopA.breedte, knopA.hoogte);  
text(knopA.titel, knopA.x + 10, knopA.y + 10, knopA.breedte-20, knopA.hoogte-20);
```

## Logische eenheden

- Of, in het geval van onze simulator:

```
var mensA = { x: 300,  
              y: 600,  
              speedX: 2,  
              speedY: -3  
            }
```

- En dan verderop:

```
mensA.x = mensA.x + mensA.speedX;  
mensA.y = mensA.y + mensA.speedY;
```



## Logische eenheden

- Of, in het geval van onze simulator:

```
var mensA = { x: 300,  
              y: 600,  
              speedX: 2,  
              speedY: -3  
            }
```

- En dan verderop:

```
mensA.x = mensA.x + mensA.speedX;  
mensA.y = mensA.y + mensA.speedY;
```

- Nog beter: als naamloze objecten in een array

# Logische eenheden

- Nog beter: als naamloze objecten in een array

```
var mensen = [ { x: 300,           for (var i=0; i<mensen.length; i++) {
                  y: 600,           mensen[i].x = mensen[i].x + mensen[i].speedX;
                  speedX: 2,        mensen[i].y = mensen[i].y + mensen[i].speedY;
                  speedY: -3        }
                },
                { x: 800,
                  y: 300,
                  speedX: -4,
                  speedY: 1
                } // etcetera
];
```

## Logische eenheden

- De updatecode hoort eigenlijk ook bij het object. Dat doe je zo:

```
var mensA = { x: 300,  
              y: 600,  
              speedX: 2,  
              speedY: -3,  
              update() {  
                this.x = this.x + this.speedX;  
                this.y = this.y + this.speedY;  
              }  
            }  
mensA.update();
```

- Waarom 'this'?
- -> De code in update kan niet 'weten' dat het object beschikbaar is onder het label 'mensA';

## Wat is wat?

```
var mensA = { x: 300,  
              y: 600,  
              speedX: 2,  
              speedY: -3,  
              update() {  
                this.x = this.x + this.speedX;  
                this.y = this.y + this.speedY;  
              }  
};  
mensA.update();
```

} attributen

} methode

## Probleempje...

```
var mensen = [ { x: 300,  
                 y: 600,  
                 speedX: 2,  
                 speedY: -3,  
                 update() {  
                     this.x = this.x + this.speedX;  
                     this.y = this.y + this.speedY;  
                 }  
               },  
               { x: 800,  
                 y: 300,  
                 speedX: -4,  
                 speedY: 1,  
                 update() {  
                     this.x = this.x + this.speedX;  
                     this.y = this.y + this.speedY;  
                 }  
               } // etcetera  
];
```

```
for (var i=0; i<mensen.length; i++) {  
    mensen[i].update()  
}
```

## Dubbele methoden

- Voor ieder object opnieuw de methodes schrijven is zonde van de tijd en opslagruimte.
- Waarom kunnen we geen objecten maken van eerder gemaakt ontwerp?
- Dat kan met de beschrijving van een klasse:



## Beschrijf de class Mens

```
class Mens {  
    x;  
    y;  
    speedX;  
    speedY;  
  
    constructor(x, y, speedX, speedY) {  
        this.x = x;  
        this.y = y;  
        this.speedX = speedX;  
        this.speedY = speedY;  
    }  
}
```

## Beschrijf de class Mens (nu met deel van update)

```
class Mens {  
    x;  
    y;  
    speedX;  
    speedY;  
  
    constructor(x, y, speedX, speedY) {  
        this.x = x;  
        this.y = y;  
        this.speedX = speedX;  
        this.speedY = speedY;  
    }  
  
    update() {  
        this.x = this.x + this.speedX;  
        this.y = this.y + this.speedY;  
        // hier moet nog veel meer komen  
    }  
}
```

## Beschrijf de class Mens (nu met deel van update)

```
class Mens {  
    x;  
    y;  
    speedX;  
    speedY;
```

```
    constructor(x, y, speedX, speedY) {  
        this.x = x;  
        this.y = y;  
        this.speedX = speedX;  
        this.speedY = speedY;  
    }
```

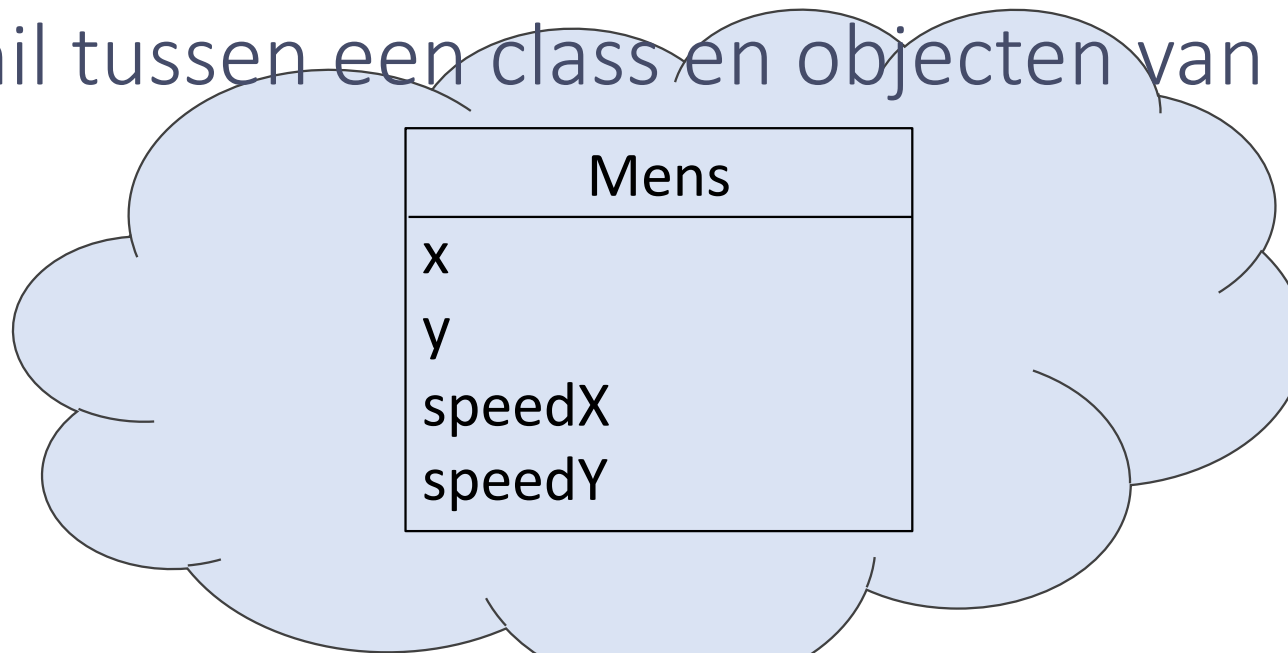
```
    update() {  
        this.x = this.x + this.speedX;  
        this.y = this.y + this.speedY;  
        // hier moet nog veel meer komen  
    }  
}
```

constructor wordt aangeroepen met 'new', zoals:

```
var mensA = new Mens(50, 50, -2, 3);
```



Verskil tussen een class en objecten van die class:



mensA : Mens	
x	= 50
y	= 50
speedX	= -2
speedY	= 3

mensB : Mens	
x	= 74
y	= 24
speedX	= -4
speedY	= 2

mensC : Mens	
x	= 150
y	= 91
speedX	= -5
speedY	= -3

# Hoe definieer ik een class?

```
class <Naam van de class> {  
    attribuut1;  
    attribuut2;  
  
    constructor(parameter1, parameter2) {  
        this.attribuut1 = parameter waarvan je de waarde wilt gebruiken;  
        this.attribuut2 = parameter waarvan je de waarde wilt gebruiken;  
    }  
  
    methodenaam() {  
        // code die uitgevoerd moet worden  
  
        return <waarde>; // alleen als er een waarde teruggegeven moet worden  
    }  
}
```

```
class Bal {  
  constructor(_x, _y, _speedX, _speedY) {  
    this.x = _x;  
    this.y = _y;  
    this.speedX = _speedX;  
    this.speedY = _speedY;  
  }  
  
  show() {  
    fill(255, 100, 255);  
    ellipse(this.x, this.y, 80, 80);  
  }  
  
  update() {  
    this.x = this.x + this.speedX;  
    this.y = this.y + this.speedY;  
  
    // hier moet ook de code voor het kaatsen komen  
    // . . .  
  }  
}
```



# Superclasses en overerving

# Stel...

Mens
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

Kat
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

- Waar zitten de verschillen?

# Stel...

Mens
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

Dokter
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

- Waar zit het verschil?

Een dokter wordt anders  
getekend.  
Wordt geregeld in show().

# show() :

## Mens

```
show() {  
  noStroke();  
  if (this.isBesmet === true) {  
    fill(255, 0, 0);    // rood  
  }  
  else {  
    fill(255, 255, 255); // wit  
  }  
  
  rect(this.x, this.y, this.breedte, this.breedte);  
}
```

## Dokter

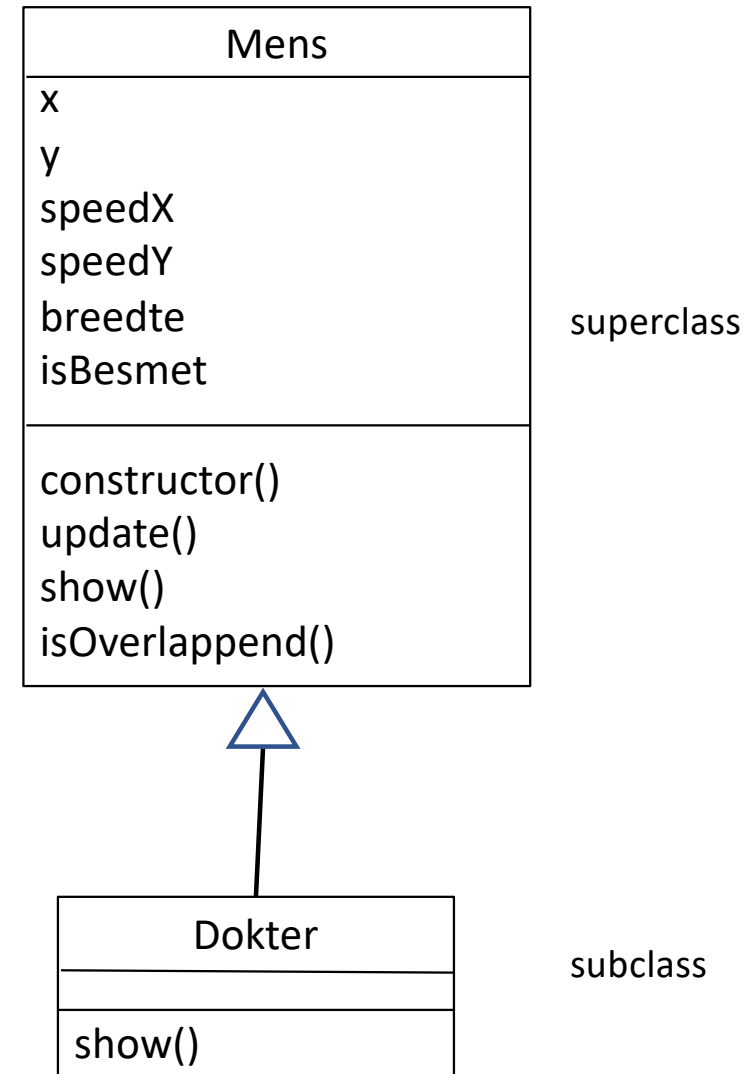
```
show() {  
  // wit vierkant  
  noStroke();  
  fill(255, 255, 255); // wit  
  rect(this.x, this.y, this.breedte, this.breedte);  
  
  // teken kruis  
  strokeWeight(5);  
  stroke(255, 0, 0);    // rood  
  line(this.x + this.breedte / 2, this.y,  
        this.x + this.breedte / 2, this.y + this.breedte);  
  line(this.x, this.y + this.breedte / 2,  
        this.x + this.breedte, this.y + this.breedte / 2);  
}
```

# Dokter & Mens

- De klasse Dokter heeft exact dezelfde attributen en methoden als de klasse Mens, maar de methode show is anders.
- In logisch opzicht is dit ook zo: een dokter is een 'speciaal soort' mens.
- Object georiënteerd programmeren biedt hiervoor *subclassing*

# Subclassing

- De klasse Dokter erft alle attributen en methoden van Mens
- De klasse Dokter heeft een eigen implementatie van de methode show()





# Subclassing in code

```
class Dokter extends Mens {  
  show() {  
    // wit vierkant  
    noStroke();  
    fill(255, 255, 255); // wit  
    rect(this.x, this.y, this.breedte, this.breedte);  
  
    // teken kruis  
    strokeWeight(5);  
    stroke(255, 0, 0); // rood  
    line(this.x + this.breedte / 2, this.y,  
         this.x + this.breedte / 2, this.y + this.breedte);  
    line(this.x, this.y + this.breedte / 2,  
         this.x + this.breedte, this.y + this.breedte / 2);  
  }  
}
```

- Het keyword `extends` geeft aan dat een klasse een subklasse is

# Subclassing in code

```
class Dokter extends Mens {  
  show() {  
    // wit vierkant  
    noStroke();  
    fill(255, 255, 255); // wit  
    rect(this.x, this.y, this.breedte, this.breedte);  
  
    // teken kruis  
    strokeWeight(5);  
    stroke(255, 0, 0); // rood  
    line(this.x + this.breedte / 2, this.y,  
          this.x + this.breedte / 2, this.y + this.breedte);  
    line(this.x, this.y + this.breedte / 2,  
          this.x + this.breedte, this.y + this.breedte / 2);  
  }  
}
```

- Moet Dokter echter ook code hebben om een vierkant te tekenen?
- Een dokter wordt altijd al wit getekend, want die is nooit besmet

# Subclassing in code

```
class Dokter extends Mens {  
  show() {  
    // wit vierkant  
    super.show();  
  
    // teken kruis  
    strokeWeight(5);  
    stroke(255, 0, 0);    // rood  
    line(this.x + this.breedte / 2, this.y,  
          this.x + this.breedte / 2, this.y + this.breedte);  
    line(this.x, this.y + this.breedte / 2,  
          this.x + this.breedte, this.y + this.breedte / 2);  
  }  
}
```

- `super.show()` roept de methode `show()` van de superklasse aan.
- `Mens` blijft zo verantwoordelijk voor het witte vierkantje van een mens
- `Dokter` is verantwoordelijk voor het rode kruis.

## Dit komt vaker voor...

Mens
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

Kat
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

- Waar zitten de verschillen?

# Stel...

Mens
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

Kat
x y speedX speedY breedte isBesmet
constructor() update() show() isOverlappend()

- Waar zitten de verschillen?

De breedte van een kat is anders.  
Wordt geregeld in de constructor.

De kleur van een kat is anders.  
Wordt geregeld in show().

## update() :

### Mens

```
update() {  
    // stuiten tegen linker- of rechterkant  
    if (this.x <= 0 || this.x + this.breedte >= width) {  
        this.speedX = this.speedX * -1;  
    }  
  
    if (this.y <= 0 || this.y + this.breedte >= height) {  
        speedY = this.speedY * -1;  
    }  
  
    // geef nieuwe positie  
    this.x = this.x - this.speedX;  
    this.y = this.y - this.speedY;  
}
```

### Kat

```
update() {  
    // stuiten tegen linker- of rechterkant  
    if (this.x <= 0 || this.x + this.breedte >= width) {  
        this.speedX = this.speedX * -1;  
    }  
  
    if (this.y <= 0 || this.y + this.breedte >= height) {  
        speedY = this.speedY * -1;  
    }  
  
    // geef nieuwe positie  
    this.x = this.x - this.speedX;  
    this.y = this.y - this.speedY;  
}
```



# show() :

## Mens

```
show() {  
  noStroke();  
  if (this.isBesmet === true) {  
    fill(255, 0, 0);    // rood  
  }  
  else {  
    fill(255, 255, 255); // wit  
  }  
  
  rect(this.x, this.y, this.breedte, this.breedte);  
}
```

## Kat

```
show() {  
  noStroke();  
  if (this.isBesmet === true) {  
    fill(255, 140, 0);  // oranje  
  }  
  else {  
    fill(0, 0, 255);    // blauw  
  }  
  
  rect(this.x, this.y, this.breedte, this.breedte);  
}
```

# constructor() :

## Mens

```
constructor(newX, newY, newSpeedX, newSpeedY) {  
    this.x = newX;  
    this.y = newY;  
    this.speedX = newSpeedX;  
    this.speedY = newSpeedY;  
    this.breedte = 20;  
  
    this.isBesmet = false;  
}
```

## Kat

```
constructor(newX, newY, newSpeedX, newSpeedY) {  
    this.x = newX;  
    this.y = newY;  
    this.speedX = newSpeedX;  
    this.speedY = newSpeedY;  
    this.breedte = 10;  
  
    this.isBesmet = false;  
}
```

# Mens en Kat

- `update()` is exact hetzelfde
- `show()` is anders voor de kleuren. De opbouw van de code is hetzelfde
- constructor is erg hetzelfde, behalve de waarde voor `this.breedte`

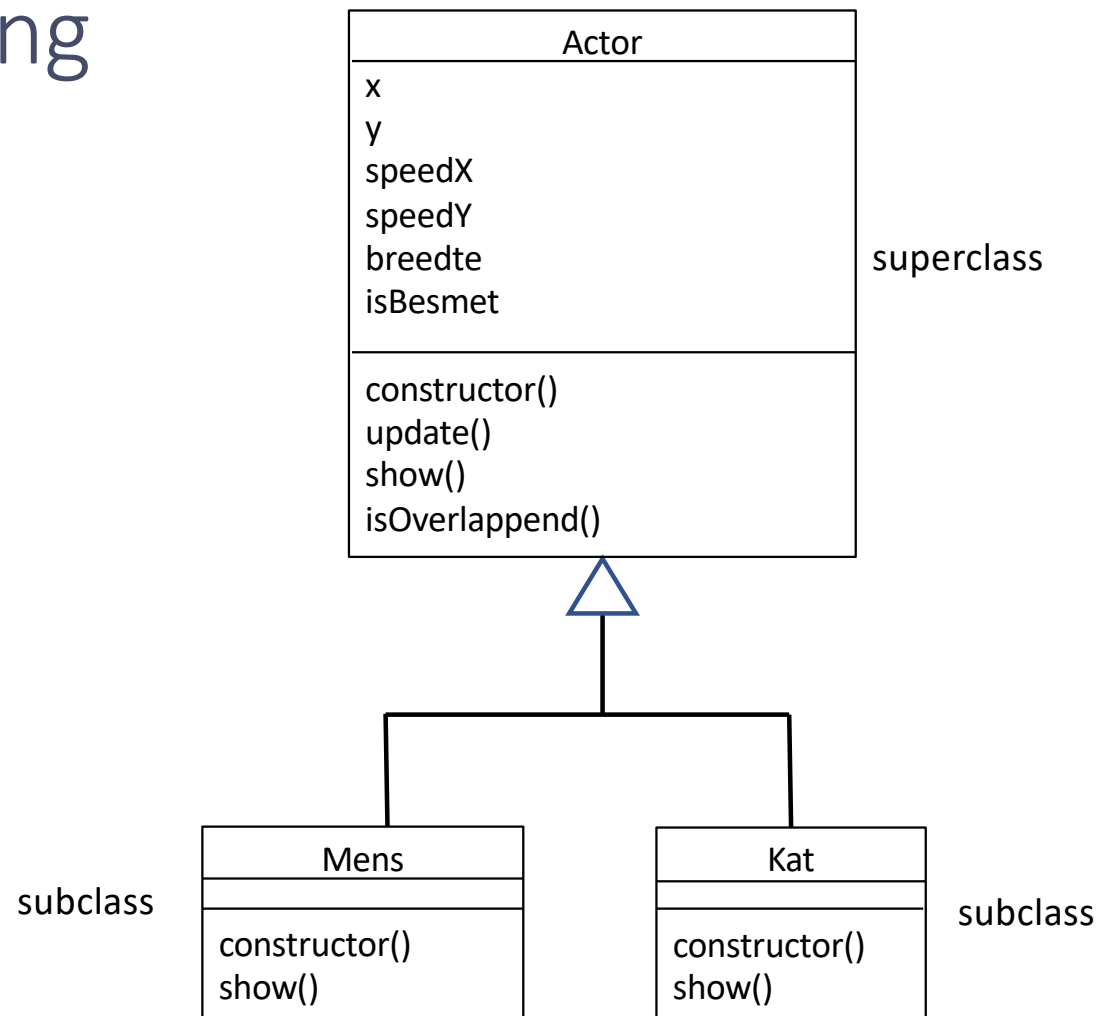
# Mens en Kat

- `update()` is exact hetzelfde
- `show()` is anders voor de kleuren. De opbouw van de code is hetzelfde
- constructor is erg hetzelfde, behalve de waarde voor `this.breedte`
- Maar... een Kat is niet een 'specifiek soort' mens of andersom
- Hoe lossen we dit op?

# Mens en Kat

- `update()` is exact hetzelfde
- `show()` is anders voor de kleuren. De opbouw van de code is hetzelfde
- constructor is erg hetzelfde, behalve de waarde voor `this.breedte`
- Maar... een Kat is niet een 'specifiek soort' mens of andersom
- Hoe lossen we dit op?
- Destilleer de gemeenschappelijk code eruit en maak er een superclass van.

# Subclassing



# Actor

- Vraag: gaan we in onze simulatie ooit instanties van Actor opnemen, of alleen van subklassen?
- Ja: Actor moet alle functionaliteit hebben
- Nee: Actor hoeft zichzelf misschien niet te kunnen tekenen

# Actor

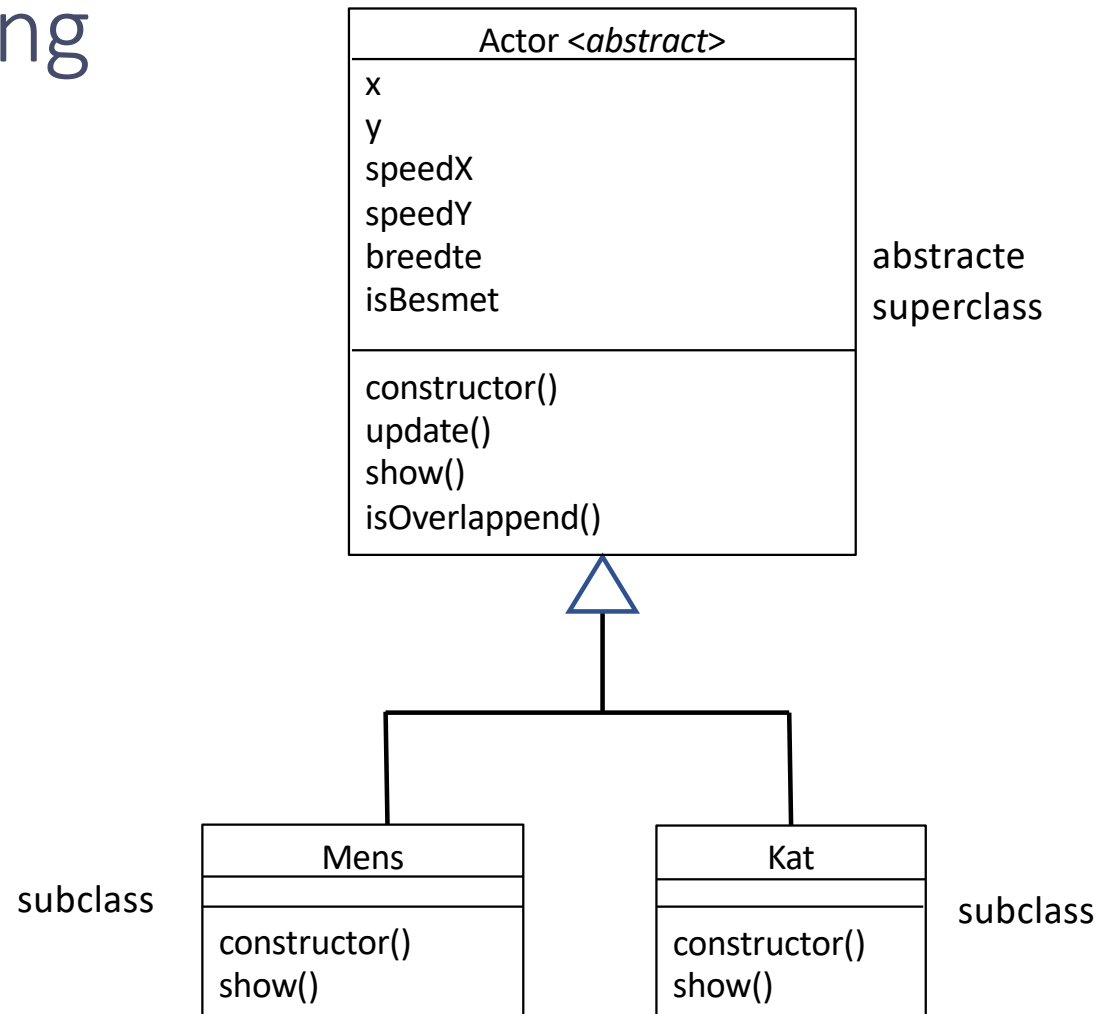
- Vraag: gaan we in onze simulatie ooit instanties van Actor opnemen, of alleen van subklassen?
- Ja: Actor moet alle functionaliteit hebben
- Nee: Actor hoeft zichzelf misschien niet te kunnen tekenen  
Maar we willen wel specificeren dat **iedere** object dat van Actor afstamt, een methode show() heeft:

show() { }

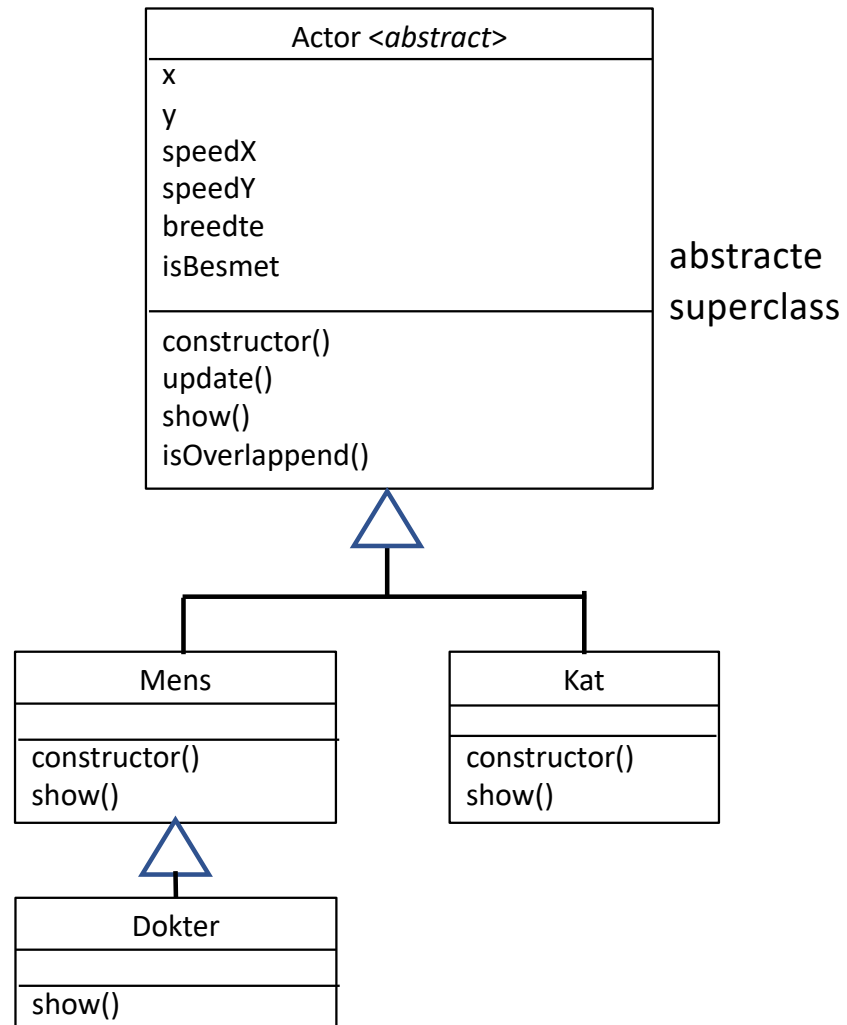
- Show is een lege methode die door een subklasse wordt 'overschreven' -> overriding
- De klasse Actor heet dan **abstract**



# Subclassing



# Hele plaatje



## Ander voorbeeld

Leerling
voornaam achternaam adres telefoonnummer email klas

Leraar
voornaam achternaam afkorting email mentorklas

# Twée classes met informatie

Leerling
voornaam achternaam adres telefoonnummer email klas

Leraar
voornaam achternaam afkorting email mentorklas

leerlingA : Leerling
voornaam: Joop achternaam: Vriezen adres: Plataanstraat 14 telefoonnummer: 0678256521 email: jopie@gmail.com klas: 4Ga

leerlingB : Leerling
voornaam: Marja achternaam: Klinklaar adres: Brink 5 telefoonnummer: 0685312314 email: jopie@gmail.com klas: 4Vb

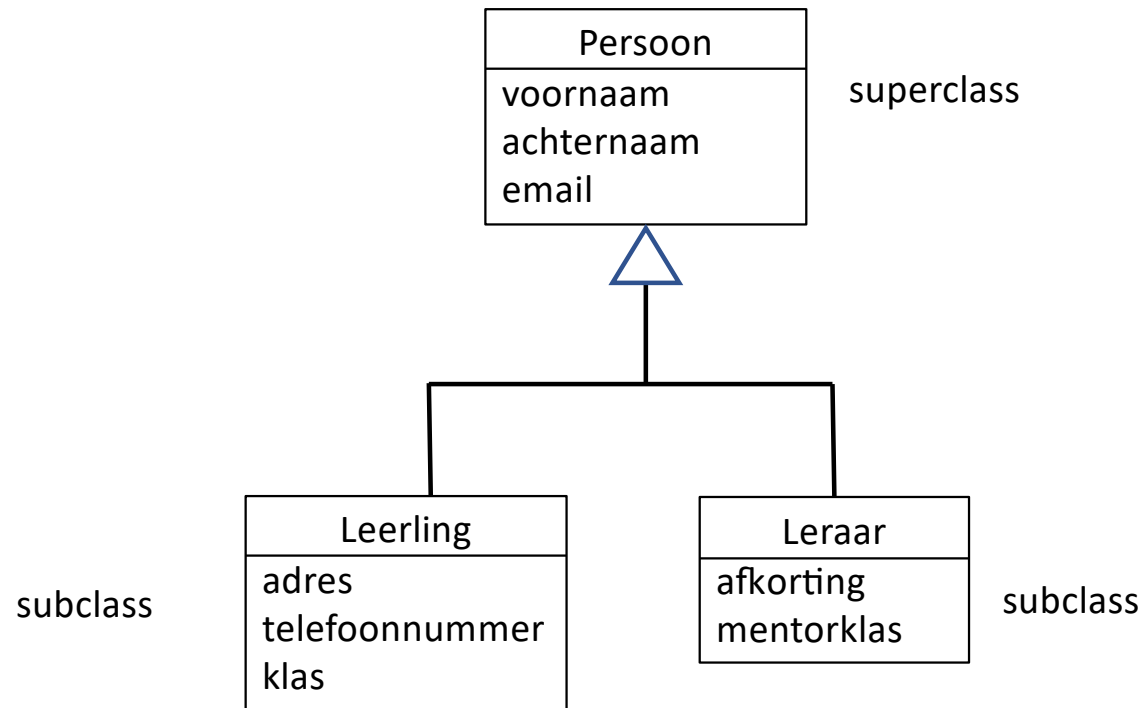
leraar1 : Leraar
voornaam: Charlie achternaam: Stipjes afkorting: STI email: STI@emmauscollege.nl mentorklas

Een deel van je klasse-ontwerp is dubbel 😞

Leerling
voornaam
achternaam
adres
telefoonnummer
email
klas

Leraar
voornaam
achternaam
afkorting
email
mentorklas

# Twée classes met informatie



# Code voor superclass / subclass

```
class Persoon {  
    voornaam;  
    achternaam;  
    email;  
  
    constructor(_voornaam, _achternaam, _email) {  
        this.voornaam = _voornaam;  
        this.achternaam = _achternaam;  
        this.email = _email;  
    }  
  
    // evt. andere methodes  
}
```

```
class Leerling extends Persoon {  
    adres;  
    telefoonnummer;  
    klas;  
  
    constructor(_voornaam, _achternaam, _email,  
                _adres, _telefoonnummer, _klas) {  
        super(_voornaam, _achternaam, _email)  
        this.adres = _adres;  
        this.telefoonnummer = _telefoonnummer;  
        this.klas = _klas;  
    }  
  
    // evt. andere methodes die niet in Persoon  
    // zitten  
}
```

# Code voor superclass / subclass

erft alle attributen en  
methodes van Persoon

```
class Persoon {  
    voornaam;  
    achternaam;  
    email;  
  
    constructor(_voornaam, _achternaam, _email) {  
        this.voornaam = _voornaam;  
        this.achternaam = _achternaam;  
        this.email = _email;  
    }  
  
    // evt. andere methodes  
}
```

```
class Leerling extends Persoon {  
    adres;  
    telefoonnummer;  
    klas;  
  
    constructor(_voornaam, _achternaam, _email,  
                _adres, _telefoonnummer, _klas) {  
        super(_voornaam, _achternaam, _email);  
        this.adres = _adres;  
        this.telefoonnummer = _telefoonnummer;  
        this.klas = _klas;  
    }  
  
    // evt. andere methodes die niet in Persoon  
    // zitten  
}
```



# Code voor superclass / subclass

erft alle attributen en  
methodes van Persoon

```
class Persoon {  
    voornaam;  
    achternaam;  
    email;  
  
    constructor(_voornaam, _achternaam, _email) {  
        this.voornaam = _voornaam;  
        this.achternaam = _achternaam;  
        this.email = _email;  
    }  
  
    // evt. andere methodes  
}
```

```
class Leerling extends Persoon {  
    adres;  
    telefoonnummer;  
    klas;  
    constructor(_voornaam, _achternaam, _email,  
                _adres, _telefoonnummer, _klas) {  
        super(_voornaam, _achternaam, _email)  
        this.adres = _adres;  
        this.telefoonnummer = _telefoonnummer;  
        this.klas = _klas;  
    }  
  
    // evt. andere methodes die niet in Persoon  
    // zitten  
}
```

# Code voor superclass / subclass

erft alle attributen en  
methodes van Persoon

```
class Persoon {
    voornaam;
    achternaam;
    email;

    constructor(_voornaam, _achternaam, _email) {
        this.voornaam = _voornaam;
        this.achternaam = _achternaam;
        this.email = _email;
    }

    // evt. andere methodes
}
```

```
class Leerling extends Persoon {
    adres;
    telefoonnummer;
    klas;

    constructor(_voornaam, _achternaam, _email,
                _adres, _telefoonnummer, _klas) {
        super(_voornaam, _achternaam, _email)
        this.adres = _adres;
        this.telefoonnummer = _telefoonnummer;
        this.klas = _klas;
    }

    // evt. andere methodes die niet in Persoon
    // zitten
}
```

constructor heeft ook de info voor  
constructor van de superclass nodig

eerste regel in constructor is  
aanroepen van constructor  
van de superclass