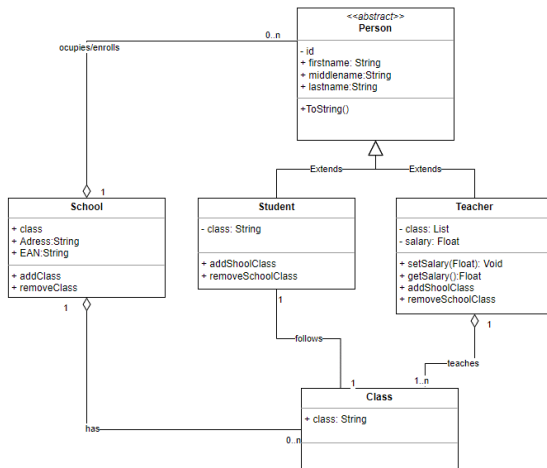


# OOP med kode

Vi forsøger at koble OOP begreber med praktisk kodning

## Fra design til kode

Navnlig kan Klasse diagrammet kædes sammen med kode. Klassediagrammet er et design dokument, der af udviklere opfattes som et blueprint af egenskaber og funktioner som objekter af klassen skal kunne.



```
01 <?php
02 class Person
03 {
04     public $name;
05
06     public function getName()
07     {
08         return $this->name;
09     }
10 }
11
12 $person = new Person();
13 $person->name = 'Bob Smith';
14 echo $person->getName(); // prints 'Bob Smith'
15 ?>
```

I både kode og design Class diagrammet kan man aflæse attributter (øverst) og metoder (nederst)

Man kan sagtens tidligt i et forløb arbejde med modellen i praksis, dvs. at skrive funktionel kode til de komponenter diagrammet indeholder. På model niveau kan man implementere interfaces, og med simple testscripts kan funktionaliteten afprøves.

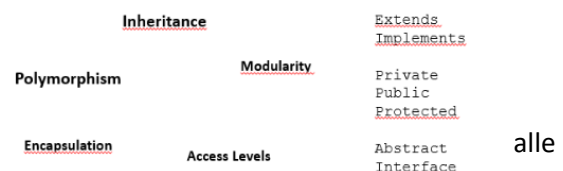
Dette beskæftiger vi os med i Demo videoerne, hvor vi undervejs afdækker OOP begreber vist ovenfor til højre. Der er 2 videoer. Se side 2 og 3 for mere om dem

1. [Programmering af en klasse \(27 minutter\)](#)
2. [OOP Polymorfi med PHP \(23 minutter\)](#)

## Object-Oriented PHP With Classes and Objects

Det kunne være søgeordene anvendt på google, og der findes mange informationer på webben der prøver at forklare om klasser og sætte det i relation til PHP.

I linket finder du forklaring på hvad en php klasse er, og begreberne vist i oversigten er afdækket, med flere eksempler.



<https://code.tutsplus.com/tutorials/basics-of-object-oriented-programming-in-php--cms-31910>

# OOP med kode

Vi forsøger at koble OOP begreber med praktisk kodning

## Demo video: [Programmering af en klasse](#)

27 minutter med en bottom up approach, at eksperimentere med en klasse, og taler om OOP begreber, praksis, og business. Selvom man måske ikke har kodet så meget, er der her en chance for afprøve kodeeksemplerne. Brug bare at echoe variabler i konsollen, hvis det er det du er vant til.

Vi gennemgår med PHP hvordan vi opbygger en klasse der kan instantieres, hvor vi starter fra bunden og ender med en standardklasse af en Teacher (en underviser). I klassen er der 3 navne attributter, og en optionel salary attribut, og en enkelt ToString() metode.

Fordi vi anvender typedefinitioner arbejder vi lidt med håndteringen af det, under en forklaring om PHP og dynamiske datatyper.

Undervejs taler vi om forskellige forretningsaspekter ift. OOP begreberne, feks. taler vi om hvorfor en attribut som \$salary sikkert er indkapslet af noget security osv.

	<p>Videoen ligger på en på Skoletube kanal: <a href="https://www.studietube.dk/group/Instruktionsvideoerogdemoer">https://www.studietube.dk/group/Instruktionsvideoerogdemoer</a> Brug denne kode: 22F378C9 for adgang</p> <p>Screenecast: <a href="#">Programmering af en klasse</a></p> <p>Kode anvendt i videoen kan downloades fra Git <a href="https://github.com/kimo1ucl/PHP-OOP-Basic-Class">https://github.com/kimo1ucl/PHP-OOP-Basic-Class</a></p>
Indhold	<p>00:00 – 05:30: En tom klasse, __destruct(), garbagecollector og hvad kan vi bruge den til?</p> <p>05:30 – 14:00: Attributter=Egenskaber tilstand, Metoder=Noget objektet kan. Objekter og attribut initialisering</p> <p>14:00 – 21:00: Access modifiers - Hvorfor skal de være private?</p> <p>21:00 – 27:42: En klasse med alle gettere og settere, __construct(), med og uden datatype erklæring. Numeriske værdier &lt;&gt; null</p>

# OOP med kode

Vi forsøger at koble OOP begreber med praktisk kodning

Demo video: [OOP Polymorfi med PHP](#)

“**Polymorphism** is a Greek word that literally means many forms. In object-oriented programming, polymorphism is closely related to inheritance.

Polymorphism allows **objects of different classes to respond differently based on the same message.**”

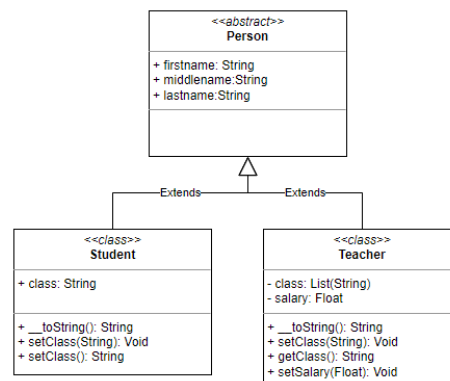
From <https://www.phptutorial.net/php-oop/php-polymorphism/>

## Scenarier

### Inheritance-Abstract Scenarie

Scenariet er en Person – Student – Teacher konstruktion, hvor en Student og en Teacher har identiske kendetegn, men de er også hver især forskellige.

Feks. har en Student ikke en Salary attribut. En Teacher kan have flere klasser, mens en Student kun kan være assignet til en bestemt klasse (eller hold om man vil)

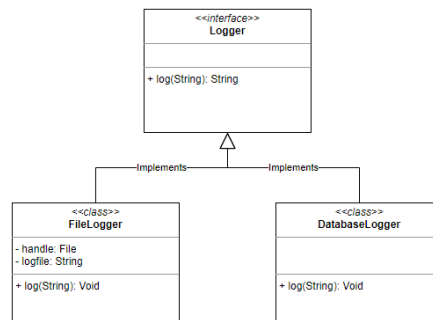


### Interface Scenarie

Interface er en anden almindelig form for polymorfisme hvor en parent klasse definerer de krævede metoder, som skal implementeres i nedarvende klasser.

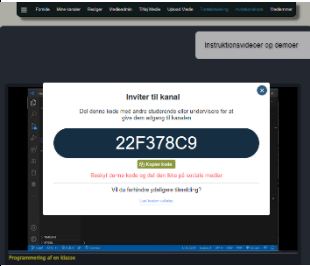
I dette scenarie har vi et logger interface, der bare forlanger en metode implementeret, nemlig function log(message) som vel må være mindstekravet i en loggermekanisme. I dette setup har vi en fil logger og en database logger, der begge nedarver fra interfacet

<https://www.phptutorial.net/php-oop/php-interface/>



# OOP med kode

Vi forsøger at koble OOP begreber med praktisk kodning

	<p>Videoen ligger på en på Skoletube kanal: <a href="https://www.studietube.dk/group/Instruktionsvideoerogdemoer">https://www.studietube.dk/group/Instruktionsvideoerogdemoer</a> Brug denne kode: 22F378C9 for adgang</p> <p>Screencast: <a href="#">OOP Polymorfi med PHP</a></p> <p>Kode anvendt i videoen kan downloades fra Git <a href="https://github.com/kimo1ucl/PHP-OOP-Polymorphism">https://github.com/kimo1ucl/PHP-OOP-Polymorphism</a></p>
Indhold	<p>00:00 – 08:55: Inheritance 08:55 – 13:23: Abstraction 13:23 – 21:25: Interface</p>