OOP

Object Oriented Programming

En walkthrough med kode og VS Code (inklusive Xdebugger) og afprøver forskellige koncepter af OOP implementeret i PHP

Udgangspunktet er en **Person – Student – Teacher** konstruktion, hvor en Studen 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 bestem klasse (eller hold om man vil)

I den sidste del arbejder vi med et ultra simplet **logger framework**, der skriver til en lokal fil. Modellen implementerer en **fil- og database logger**, database loggeren er dog ren facade.

Perspektiv

OOA

- use cases and object models
- User-interface mockups or prototypes
- *What* is to be built

OOD

- *how* the system is to be built, implementation constraints to the conceptual model
- concrete technologies
- hardware and software platforms, persistent storage
- usability of the system
- architectural patterns and design patterns (eks. MVC, Singleton, Observer (publish/subscribe))

OOP

- **Encapsulation,**
- **Composition**
- **Inheritance, Polymorphism**
- Classes and objects as instances of classes
- Encapsulating Attribute/properties and methods
- *Building* it

OOP Begreber og kode

Inheritance

Polymorphism

Modularity

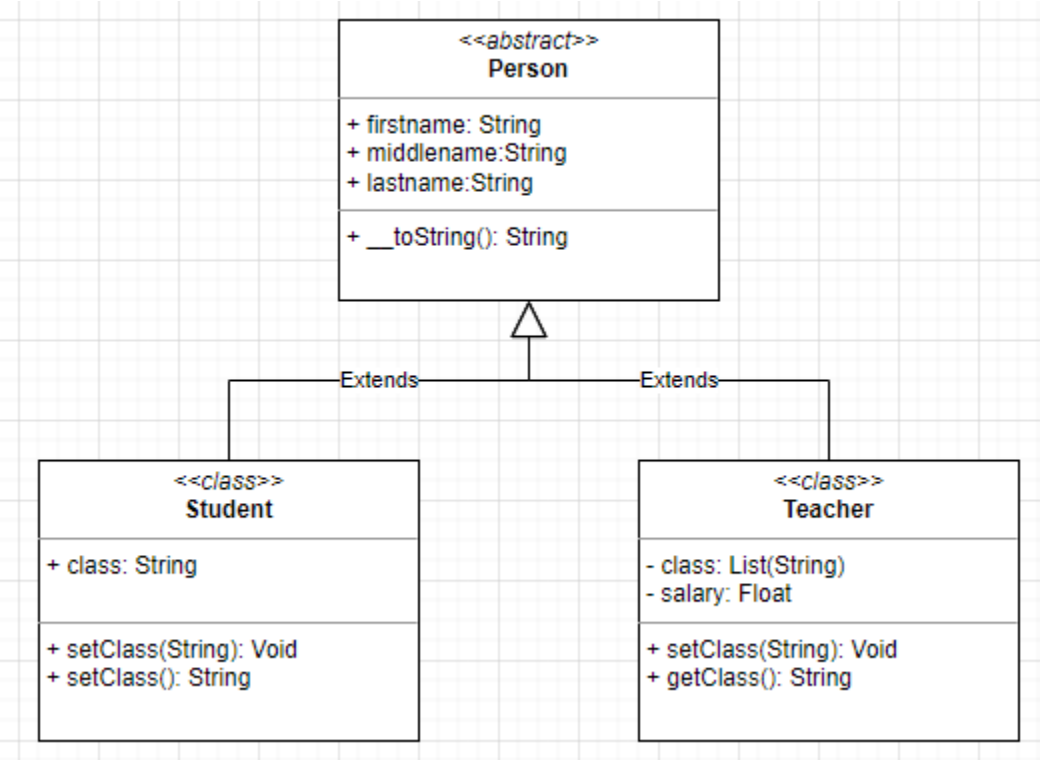
Encapsulation

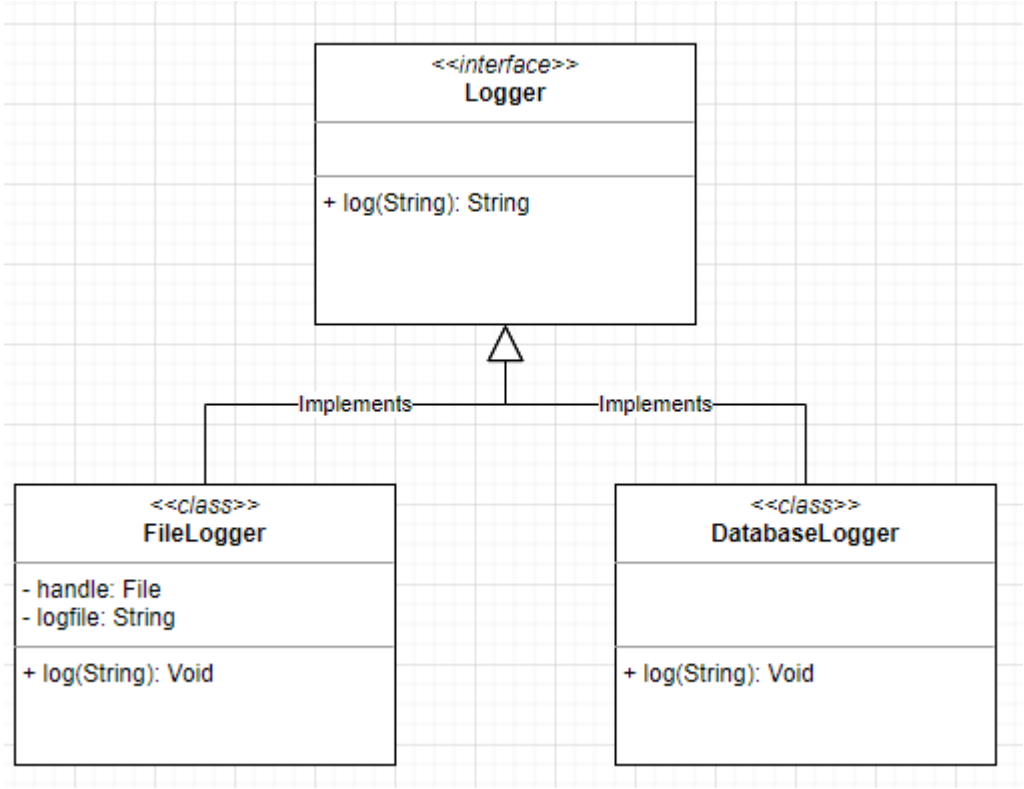
```
Extends
Implements

Private
Public
Protected

Abstract
Interface
```

Klassediagrammer





Modularity

Modularity

A **class** (file) is a blueprint representation of a domain **objects** with their own **fields** and **functionality**. **State** and **behavour**. **Properties** and **Methods**

By keeping the code in separate files, we are better able to work with the code, we can track the changes and it is easier to maintain. More contributers can work simultaneously

1.Intro\2. StateAndBehaviour.php

Vi instantiere en Teacher klasse og taler om klassens design, objekternes attributer og metoder

Encapsulation

Encapsulation

After having instantiated an object, we can call the methods immediately. The implementation is hidden and we should not be concerned about it.

1.Intro\3. AccessModifiers.php
1.Intro\4. ConstructorsDestructors.php

Vi gør et par metoder til private af sikkerhedshensyn
Og bliver eksplicitte med konstruktor og destruktor

Inheritance

Inheritance

In a hierarchy of general and more specialized classes we can make changes on the general object, and it will work in all classes which inherits (extends) from the general class.

Promotes reuse of code and flexibility

2. Inheritance

Vi laver en arvestruktur bestående af Person og Teacher

Vi placerer constructor og fordeler metoder og attributer i modellen der nu består af to clæasser

Polymorphism

Polymorphism

A child class can change behaviour of the parents functionality, we do this by overriding the implementation made in the parent Class

3. Abstract

Vi ændrer midfier til protected en række steder, og udtrykker i abstrakten default metoder

Vi tilføjer klassen Student og implementerer hvad der kræves for at opfylde kontrakten

Interface

Interface

Like an Abstract without properties.

Only methods, which the derriving classes must implement.

Usefull for database functionality, and frameworks where all classes should do the same, like we wnat it in forinstance services- and logging frameworks

3. Abstract

Vi ændrer midfier til protected en række steder, og udtrykker i abstrakten default metoder

Vi tilføjer klassen Student og implementerer hvad der kræves for at opfylde kontrakten

Kahoot

DRY

Don’t Repeat Yourself

“Every piece of knowledge must have a single, unambiguous, authoritative representation within a system”
The aim is to reduces repetition of software patterns
The principle has been formulated by Andy Hunt and Dave Thomas in their book *The Pragmatic Programmer*.

WET

Write everything twice

Or some more bias containing understandings as

"write every time", "we enjoy typing" or "waste everyone's time"

AHA

Avoid hasty abstractions

The trade choice.
Change first, design later
This can perhabs continue for a while until we may know more about the demands and requirements af whatever we were working with.
Untill duplication also is a problem, then redesign the thing.