```php
<?php

    /**
     * Klasse voor Studentenkunstjes
     * @throws YourLevenOverhoopException
     */
    class Student implements succes
    {
        public $JOUW_NAAM =
        "                                                    ";

    }

?>
```

Object-Oriented Programming
OEFENING

- ✓ **Casing**
  - Variabelen, Methods/Functions -> (lower) camel case
  - Interfaces-> I + (lower) camel case
  - Constantes -> upper case (all caps)
  - Klassen -> Pascal (upper camel) case
- ✓ **DocBlocks** *Verplichte (meer mag, minder niet) tags en annotatie:*
  - Bijna overal -> beschrijving van de bijbehorende code EN van de gebruikte tags
  - Klasse -> @property voor elk veld in de klasse
  - Functie -> @param *(per parameter)*, @return *(datatype indien niet void)*, @throws *(exceptions bij defensieve implementatie)*
  - Veld -> zie klasse (properties hoeven geen beschrijving, zelfs geen eigen DocBlock)
  - Getter/Setter -> geen beschrijving nodig
- ✓ **Paradigma** *Contractueel is niet toegelaten!*
  - Kies voor totaal of defensief programmeren en pas overal toe
  - !!! -> Vermeld je keuze in een comment in de Sequencer-klasse
- ✓ **Kies voor maximale encapsulatie van jouw code**
- ✓ **Voorzie 'static' en 'abstract' elementen van een verklaring voor die keuze**
- ✓ **Gebruik zinvolle en verstaanbare benamingen en wees consequent**
  - Engels en Nederlands mogen wel door elkaar gebruikt worden

✓ De super-diesel-turbo-zestien-kleppen-**SEQUENCER**-klasse met 'zijspan':

Objecten van de klasse **Sequencer** (her)ordenen arrays met strings op basis van een meegegeven argument van de klasse **Vergelijking**(smethode).
De klasse **Vergelijking** beschrijft objecten die algoritmes voor ordening van strings bevatten.
ALLE algoritmes implementeren de interface **sequenceMethode**.
Objecten van de klasse **StandaardMethode** implementeren EEN alfabetische methode om strings in een array te ordenen.
Objecten van de klasse **IndexMethode** implementeren EEN methode om een array met strings te ordenen op basis van een als argument meegegeven index.
De index bepaalt de hoeveelste letter van de te vergelijken strings gebruikt wordt als basis voor de vergelijking.

- ✓ TE SCHRIJVEN KLASSEN *OPGELET: contractueel paradigma NIET gebruiken!* : Sequencer, Vergelijking, StandaardMethode, IndexMethode
- ✓ GEGEVEN CODE *Is al volledig afgewerkt; niet meer aanpassen/aanvullen dus!* :

```php
<?php
    /**
     * Methods to be implemented by all implementing classes.
     */
    interface sequenceMethode
    {
        /**
         * Een functie die test of het eerste argument 'kleiner' is dan het tweede.
         * @param mixed $a -> Het eerste te vergelijken argument.
         * @param mixed $b -> Het tweede te vergelijken argument.
         * @return bool -> Returns true if $a < $b
         */
        public function kleiner($a, $b);

        /**
         * Een functie die test of het eerste argument 'groter' is dan het tweede.
         * @param mixed $a -> Het eerste te vergelijken argument.
         * @param mixed $b -> Het tweede te vergelijken argument.
         * @return bool -> Returns true if $a > $b
         */
        public function groter($a, $b);

        /**
         * Een functie die test of het eerste argument 'gelijk' is aan het tweede.
         * @param mixed $a -> Het eerste te vergelijken argument.
         * @param mixed $b -> Het tweede te vergelijken argument.
         * @return bool -> Returns true if $a == $b
         */
        public function gelijk($a, $b);

    }
?>
```

opdracht