	<b>Hogeschool PXL - Switch2IT</b> Academiejaar 2019-2020 <u>EXAMEN</u> Web Advanced
---	---

Vak	
Resultaat	/20
Periode	1e zit
Datum	2 dec 2019
Tijdstip	8u30
Klassen	2TIW
Lectoren	J. Willekens

Studentengegevens	
Naam student	
Voornaam student	
Klas	
Lector	
Lokaal	

Samenstelling bundel			
Onderdelen (*)	Examen	PE	
Inhoud		permanente evaluatie	
Pagina's	12 pagina's		
Puntenverdeling	.... /15p (75%)	... / 5p (25%)	
Digitaal beginbestand	ja, via Filezilla		
Digitale indiening	ja, via Filezilla		
Toegelaten hulpmiddelen:			
* rekenmachine	nee		
* laptop	ja		
* internet	nee		
* cursusmateriaal	ja (afgedrukt & op laptop)		
Opmerkingen:			

*Elke student(e) is verantwoordelijk voor de correcte samenstelling van zijn/haar bundeltje. Eventuele afwijkingen moeten onmiddellijk aan de toezichthouder signaleerd worden.*

MD5Hash:

Aanvangsuur examen: 8u30

Einde examen: 12u



## Checklist voor het afleggen van examen

- ☐ GSM's/smartphones liggen uitgeschakeld op tafel
- ☐ Studentenkaart ter beschikking houden
- ☐ Jassen en tassen vooraan in het lokaal
- ☐ GEEN GSM's of smartphones in jassen of tassen
- ☐ Bij laptopexamen strikt de opgelegde regels volgen
- ☐ Examenbundels blijven steeds samengeniet
- ☐ Toegestaan: 1 droog koekje en 1 drankje in hersluitbaar flesje
- ☐ Iedereen zwijgt tijdens het examen

- 
- ☐ Afgeven mag niet voor 9:00/14:00 uur

- ☐ Niet vergeten op je examenkopij te vermelden:

Naam van de lector

Eigen naam

Klas

MD5Hash



Je krijgt code toegestuurd via Filezilla bij de aanvang van het examen. Na het examen geef je af via Filezilla. De naam van het bestand dat je afgeeft is van de vorm  
naam\_voornaam.zip

*(1) Plaats in elk bestand dat je afgeeft je naam in commentaar.*

*(2) Verwijder op het einde van het examen de mappen `node_modules` en `vendor` uit de directories die je (in een zip) afgeeft.*

```
src
├── assets
│   ├── HasBorrower.php
│   └── Item.php
├── identifiable
│   └── Identifiable.php
└── users
    ├── Company.php
    ├── Person.php
    ├── UserFactory.php
    └── User.php
```

Maak in de map oef1/src/identifiable de klasse Identifiable.

Identifiable is abstract.

Deze klasse heeft \$id als private eigenschap. Verder is er ook nog een private static eigenschap usedIds (in deze eigenschap worden de al ingenomen id's bewaard).

De constructor krijgt een waarde voor \$id binnen als argument.

Er wordt gecontroleerd of

- id een integer is, groter dan 0 (hint: is\_init(12) )
- id nog niet in de array usedIds staat (hint: in\_array('a', ['a', 'b', 'c']))

Indien niet aan deze voorwaarden voldaan is wordt een InvalidArgumentException opgeworpen. Indien wel voldaan is aan de voorwaarde wordt \$id toegekend aan de eigenschap \$id en wordt \$id toegevoegd aan de array usedIds.

Maak in de map oef1/src/users de klassen User, Person, Company en UserFactory.

User is een abstracte klasse afgeleid van de klasse Identifiable. Verder hoeft je geen code toe te voegen aan deze klasse.

Person wordt afgeleid van de klasse User.

Person heeft bijkomende eigenschap \$name.

Via de public constructor wordt een Person-object aangemaakt. Bij het aanmaken worden de eigenschappen \$id en \$name toegekend. Alle spaties vooraan en achteraan \$name worden hierbij eerst verwijderd (hint: trim('aaa ')).

Voor \$id gelden dezelfde regels als in de klasse Identifiable. Voor \$name geldt dat

- \$name een string moet zijn (hint: is\_string('aaa')).
- \$name lengte tussen 3 en 20 mag hebben (hint: strlen('aaa')).

Indien niet aan deze voorwaarden voldaan is wordt een InvalidArgumentException opgeworpen.

Via de toString methode wordt een string van de vorm

1\_geert

terugggeven voor de Person met id 1 en name geert (id gevolgd door een underscore gevolgd door name).

Company wordt afgeleid van de klasse User.

Company heeft bijkomende eigenschap \$businessId.

Via de public constructor wordt een Company-object aangemaakt. De constructor heeft argumenten \$id en \$businessId. Bij het aanmaken worden de argumenten aan de eigenschappen \$id en \$businessId toegekend.

Voor \$id gelden dezelfde regels als in de klasse Identifiable. Voor \$businessId geldt dat

- \$businessId een string moet zijn (hint: is\_string('aaa')).
- \$businessId lengte 6 heeft.

Indien niet aan deze voorwaarden voldaan is wordt een InvalidArgumentException opgeworpen.

In de toString methode wordt een string van de vorm

2\_aabbcc

terugggeven voor de Company met id 1 en businessId aabbcc (id gevolgd door een underscore gevolgd door businessId).

In de klasse UserFactory worden de methoden makePerson en makeCompany voorzien. Via deze methoden kan een Person-object aangemaakt worden als

```
$person = UserFactory::makePerson(1, 'geert');
```

en een Company-object aangemaakt worden als

```
$company = UserFactory::makeCompany(2, 'aabbcc');
```

Maak in de map oef1/src/assets de klasse Item en de interface HasBorrower.

In de interface HasBorrower worden de onderstaande methoden voorzien:

```
...
interface HasBorrower{
    public function setBorrower(User $borrower);
    public function getBorrower();
    public function getPreviousBorrowers();
}
```

De klasse Item is afgeleid van Identifiable en implementeert de interface HasBorrower.

De klasse Item heeft bijkomende eigenschappen \$borrower en \$previousBorrowers.

De constructor krijgt een waarde binnen voor \$id en \$borrower. Deze worden toegekend aan de eigenschappen \$id en \$borrower. Verder wordt de eigenschap \$previousBorrowers gelijk gesteld aan een lege array.

Via de methode setBorrower wordt een nieuwe ontleners van het Item toegekend. De eigenschap \$borrower wordt hierbij eerst toegevoegd aan de array \$previousBorrowers. Vervolgens wordt het argument \$borrower toegekend aan de eigenschap \$borrower.

De methode getBorrower geeft de eigenschap \$borrower terug.

De methode getPreviousBorrowers geeft de eigenschap \$previousBorrowers terug.

De toString-methode geeft een string van de vorm

```
4#2_leen[1_geert,2_leen,3_aabbcc]
```

Waarbij 4 het id is van het Item. leen met id 1 de Person die het item nu uitleent. De personen geert, leen en het bedrijf aabbcc zijn de vorige uitleners. (id van het item gevolgd door hash-tag gevolgd door de toString van de huidige lener gevolgd door de toString van alle voorgaande leners tussen vierkante haakjes en gescheiden door een komma.

Maak een autoloader via composer en zorg ervoor dat de code in app.php correct uitgevoerd wordt.

**Maak een screenshot van de werking van app.php (in de prompt of in de browser) en noem deze oef1.png of oef1.jpg).**

**Maak de onderstaande unit-tests in de map tests** waarin het volgende geverifieerd wordt

- Er wordt een `InvalidArgumentException` opgeworpen vanuit de constructor van `Person` wanneer deze aangeroepen wordt met argumenten "mis" en "geert".
- De constructor van de klasse `Person` maakt een `Person`-object aan wanneer deze aangeroepen wordt met argumenten 1 en "geert".
- De `toString` methode voor een `Person` met id 1 en name geert geeft de string "1\_geert" terug.

**Maak een screenshot van de werking van de unit-tests en noem deze oef1b.png (of oef1b.jpg).**

**Verwijder zeker de map vendor uit de code die je afgeeft.**

De SQL-code in het bestand oef2.sql wordt je bezorgd via Filezilla.

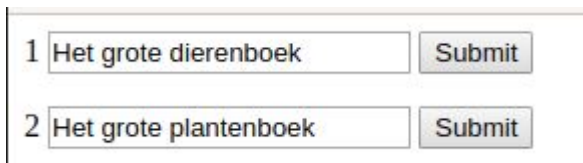
Maak het onderstaande formulier in oef2/index.php.



De onderstaande HTML-code wordt in index.php gebruikt (de id's en names van de authors worden opgehaald uit de databank).

```
<form action="showbooks.php">
  <select name="id">
    <option value="1">tim</option>
    <option value="2">sofie</option>
    <option value="3">jan</option>
  </select>
  <input type="submit" value="Submit">
</form>
```

In oef2/showbooks.php worden de id's en de titles van de geselecteerde auteur getoond. De titles worden hierbij in een tekstveld geplaatst.



De onderstaande HTML-code wordt in showbooks.php gebruikt (de id's en titles van de books worden opgehaald uit de databank).

```
<form action="updatetitle.php">
  1 <input type="hidden" name="id" value="1"/>
  <input type="text" name="title" value="Het grote dierenboek"/>
  <input type="submit" value="Submit">
</form>
<form action="updatetitle.php">
  2 <input type="hidden" name="id" value="2"/>
  <input type="text" name="title" value="Het grote plantenboek"/>
  <input type="submit" value="Submit">
</form>
```



In updatetitle.php wordt de gewijzigde titel afgedrukt. De wijziging wordt natuurlijk ook doorgevoerd in de databank. In het onderstaande voorbeeld werd de waarde “Het grote insectenboek” in het tekstveld bij id 1 geplaatst en werd op de bovenste submit-knop gedrukt.

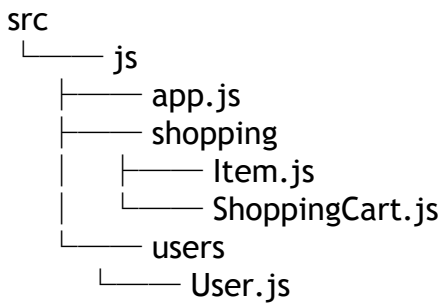
---

Het grote insectenboek

Hint:

```
SELECT * FROM author
SELECT * FROM book WHERE author_id = 1
UPDATE book SET title = 'Het grote insectenboek' WHERE id = 1
```

Neem screenshots van index.php, showbooks.php en updatetitle.php en noem deze oef2.png, oef2b.png en oef2c.png (of oef2.jpg, ...).



Maak in de map oef1/src/js/users de klasse User.

User heeft heeft eigenschap `_name`.

De constructor heeft als argument een waarde voor name. Deze waarde wordt toegekend aan de eigenschap `_name` op voorwaarde dat name een string met lengte groter dan 3 (hint: `typeof name==='string', name.length`).

In de `toString` methode wordt een string van onderstaande vorm teruggekeerd

(geert)

voor de User met `_name` geert (`_name` wordt tussen ronde haakjes geplaatst).

Maak in de map oef1/src/js/shopping de klasse Item.

Item heeft eigenschappen `_id` en `_price`. De constructor heeft als argumenten een waarde voor id en price. Deze worden toegekend aan de eigenschappen `_id` en `_price`.

Via de methode `getPrice` wordt de waarde van de eigenschap `_price` teruggeven.

In de `toString` methode wordt een string van onderstaande vorm teruggekeerd

1\_2.33

voor het Item met `_id` 1 en `_price` 2.33 (`_id` gevolgd door een underscore gevolgd door `_price`).

**Maak in de map oef1/src/js/shopping de klasse ShoppingCart.**

ShoppingCart heeft eigenschappen `_user` en `_items`.

De constructor heeft één argument: `user`. Er wordt gecontroleerd of `user` een object van de klasse `User` is. Indien dit niet zo is wordt een `Error` opgeworpen. Anders wordt de `user` toegekend aan de eigenschap `_user` en wordt de eigenschap `_items` gelijk gesteld aan een lege array (hint: `user instanceof User`).

Via de methode `calculatePrice` wordt de som van prijzen (`prices`) van alle items berekend en teruggegeven.

In de `toString`-methode wordt een string van onderstaande vorm teruggekeerd

(geert) [1\_2.33,1\_2.33,2\_12.332]

(de `toString` van `_user` gevolgd door tussen vierkante haken de `toString` van alle items in `_items` gescheiden door komma's).

**Maak een screenshot van de werking van `app.js` (in de browser) en noem deze `oef3.png` (of `oef3.jpg`).**

Maak de onderstaande unit-tests in de map `tests` waarin het volgende gecontroleerd wordt

- Er wordt een `Error` opgeworpen vanuit de constructor van `User` wanneer de string "m" ingegeven wordt als argument.
- Er wordt een `Error` opgeworpen vanuit de constructor van `User` wanneer de int 1 ingegeven wordt als eerste argument.
- De `toString` methode voor een `User` met `_name` "geert" geeft een string van de vorm (geert) terug.

**Maak een screenshot van de werking van de unit-tests en noem deze `oef3b.png` (of `oef3b.jpg`).**

Verwijder zeker de map `node_modules` uit de code die je afgeeft.

## Kladblad

