UML

Baptiste Bauer

Version v0.0.8.sip-221125094157, 2022-11-23 09:32:18

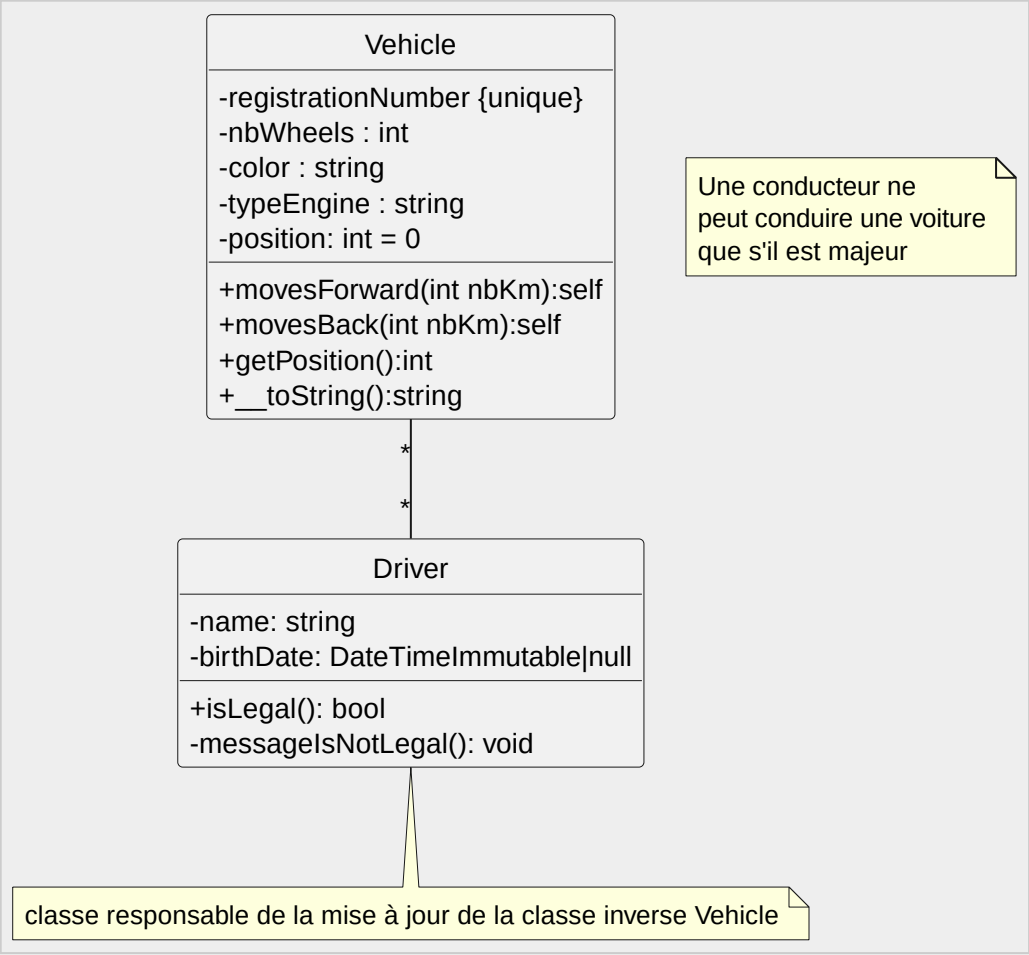
Table des matières

1. Quelques exercices d'implémentation

1.1. Implémenter une association bidirectionnelle many-to-many

Q1) Travail à faire

- Implémentez le diagramme suivant :



- Créez deux conducteurs majeurs Anna et Jalila
- Créez 4 véhicules A, B, C et D
- Associez les véhicules A et B à Anna.
- Associez les véhicules B, C et D à Jalila
- Afficher à partir d'Anna les véhicules qu'elle conduit
- Afficher à partir du véhicule B ses conducteurs.

Correction de Q1

```
1 class Vehicle implements Stringable
2 {
3     //Déclaration des variables d'objet
```

```php
    /**
     * @var string numéro d'immatriculation du véhicule
     */
    private string $registrationNumber;
    /**
     * @var int nombre de roues
     */
    private int $nbWheels;
    /**
     * @var string couleur du véhicule
     */
    private string $color;
    /**
     * @var string type de moteur (essence, diesel, électrique, hydrogène, ...)
     */
    private string $typeEngine;
    /**
     * @var int Position du véhicule
     */
    private int $position = 0;

    /**
     * @var array collection d'objets de type Driver
     */
    private array $drivers = []; ①


    //ici déclaration des autres attributs d'objets
    /**
     * @param Driver $driver conducteur associé au véhicule
     */
    public function __construct(
    )
    {
    }

    //mutateurs et accesseur pour la collection $drivers

    /**
     * @param Driver $driver ajoute un item de type Driver à la collection
     */
    public function addDriver(Driver $driver): bool ②
    {
        if (!in_array($driver, $this->drivers, true)) {
            $this->drivers[] = $driver;
            //mise à jour de l'objet inverse (lié)
            $driver->addVehicle($this); ⑤
            return true;
        }

```

```
54          return false;
55      }
56
57      /**
58       * @param Driver $driver retire l'item de la collection
59       */
60      public function removeDriver(Driver $driver): bool ③
61      {
62          $key = array_search($driver, $this->drivers, true);
63
64          if ($key !== false) {
65              unset($this->drivers[$key]);
66              //mise à jour de l'objet inverse (lié)
67              $driver->removeVehicle($this); ⑤
68              return true;
69          }
70
71          return false;
72      }
73
74      /**
75       * @return Driver[]
76       */
77      public function getDrivers(): array ④
78      {
79          return $this->drivers;
80      }
81
82
83      // ici les autres opérations (mutateurs, setters, ...)
84
85
86
87      //region ****** Mutateurs et accesseurs ******
88
89
90      /**
91       * @return string
92       */
93      public function getRegistrationNumber(): string
94      {
95          return $this->registrationNumber;
96      }
97
98      /**
99       * @return int
100      */
101     public function getNbWheels(): int
102     {
103         return $this->nbWheels;
104     }
```

```php

    /**
     * @return string
     */
    public function getColor(): string
    {
        return $this->color;
    }

    /**
     * @return string
     */
    public function getTypeEngine(): string
    {
        return $this->typeEngine;
    }

    /**
     * @return int
     */
    public function getPosition(): int
    {
        return $this->position;
    }

    /**
     * @param string $registrationNumber
     *
     * @return Vehicle
     */
    public function setRegistrationNumber(string $registrationNumber):
Vehicle
    {
        $this->registrationNumber = $registrationNumber;

        return $this;
    }

    /**
     * @param int $nbWheels
     *
     * @return Vehicule
     */
    public function setNbWheels(int $nbWheels): self
    {
        $this->nbWheels = $nbWheels;

        return $this;
    }

    /**
```

```php
     * @param string $color
     *
     * @return Vehicule
     */
    public function setColor(string $color): self
    {
        $this->color = $color;

        return $this;
    }

    /**
     * @param string $typeEngine
     *
     * @return Vehicule
     */
    public function setTypeEngine(string $typeEngine): self
    {
        $this->typeEngine = $typeEngine;

        return $this;
    }

    //endregion ****** Mutateurs et accesseurs ******


    //region ****** Autres opérations ******

    public function __toString(): string
    {
        return "Immatriculation : {$this->registrationNumber} ; Nombre de roues : {$this->nbWheels} ; couleur : {$this->color} ; type de moteur {$this->typeEngine} ; position : {$this->position}";
    }


    public function moveForward(int $nbKms): self
    {
        $this->position += $nbKms;

        return $this;
    }

    public function moveBack(int $nbKms): self
    {
        $this->position -= $nbKms;

        return $this;
    }

    //endregion ****** Autres opérations ******
```

```php

}


class Driver
{
    /**
     * @param string            $name      nom du conducteur
     * @param DateTimeImmutable $birthDate date de naissance du conducteur
     * @param array|Vehicle[]   $vehicles  collection d'objets de type
    Vehicle
     */
    public function __construct(
        private string $name,
        private DateTimeImmutable $birthDate,
        private array $vehicles = []
    ) {
    }

    //autres mutateurs, accesseurs et méthodes


    //region ****** mutateurs et accesseurs ******

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }

    /**
     * @param string $name
     *
     * @return Driver
     */
    public function setName(string $name): self
    {
        $this->name = $name;

        return $this;
    }

    /**
     * @return DateTimeImmutable
     */
    public function getBirthDate(): DateTimeImmutable
    {
        return $this->birthDate;
    }
```

```
254
255     /**
256      * @param DateTimeImmutable $birthDate
257      *
258      * @return Driver
259      */
260     public function setBirthDate(DateTimeImmutable $birthDate): self
261     {
262         $this->birthDate = $birthDate;
263
264         //si jamais la personne change sa date de naissance pour une date qui
    la conduit à être mineure, alors il faut désassocier les véhicules
265         if (!$this->isLegal()) {
266             $this->messageIsNotLegal();
267             $this->vehicle = [];
268         }
269
270         return $this;
271     }
272
273
274     /**
275      * @param Vehicle $vehicle ajoute un item de type Vehicle à la collection
276      */
277     public function addVehicle(Vehicle $vehicle): bool
278     {
279         //tester si le conducteur est mineur
280         if (!$this->isLegal()) {
281             $this->messageIsNotLegal();
282
283             return false;
284         }
285         if (!in_array($vehicle, $this->vehicles, true)) {
286             $this->vehicles[] = $vehicle;
287             ④
288             //pas de mise à jour de l'objet lié car c'est l'objet Vehicle qui
    est responsable de la navigabilité bidirectionnelle
289             return true;
290         }
291
292         return false;
293     }
294
295     /**
296      * @param Vehicle $vehicle retire l'item de la collection
297      */
298     public function removeVehicle(Vehicle $vehicle): bool
299     {
300
301         $key = array_search($vehicle, $this->vehicles, true);
302
```

```php
        if ($key !== false) {
            unset($this->vehicles[$key]);

            return true;
        }

        return false;
    }

    /**
     * @return Vehicle[]
     */
    public function getVehicles(): array
    {
        return $this->vehicles;
    }


    private function messageIsNotLegal(): void
    {
        echo "\n La personne est mineure, il n'est pas possible de lui associer un véhicule.\n";

    }

    //endregion ****** mutateurs et accesseurs ******

    //region ****** Autres opérations ******

    public function isLegal(): bool
    {
        //date courante
        $now = new DateTimeImmutable(
        ); //j'aurais pu prendre une date de type DateTime
        //calcul de la différente entre la date du jour et la date de naissance
        //documentation : https://www.php.net/manual/fr/datetime.diff.php
        $interval = $this->birthDate->diff($now);
        //cet interval est un objet, il faut retourner l'écart en nombre d'années
        $years = (int)$interval->format('%r%y');

        return $years > 18;

    }

    //endregion ****** Autres opérations ******
}
```

Mise en oeuvre des classes :

```php
echo "\n\n==== DEBUT DU TEST EXERCICE 6 ==== \n";

//création de Anna (majeure)
$driverAnna = new Driver('Anna', new DateTimeImmutable('2000-04-12'));

//création de Juliette (majeure)
$driverJalila = new Driver('Jalila', new DateTimeImmutable('1998-02-24'));

//création des 4 véhicules
$vA = (new Vehicle())->setRegistrationNumber('AAAA'); ①
$vB = (new Vehicle())->setRegistrationNumber('BBBB');
$vC = (new Vehicle())->setRegistrationNumber('CCCC');
$vD = (new Vehicle())->setRegistrationNumber('DDDD');

//Affectation des véhicules A et B à Anna
$vA->addDriver($driverAnna); ①
$vB->addDriver($driverAnna);

//Affectation des véhicule B, C et D à Jalila
$vB->addDriver($driverJalila);
$vC->addDriver($driverJalila);
$vD->addDriver($driverJalila);

//liste des véhicules conduits par Anna
echo "\n *** Liste des véhicules de {$driverAnna->getName()} ***";
foreach ($driverAnna->getVehicles() as $vehicle) {
    echo "\n-{$vehicle->getRegistrationNumber()}";
}

//liste des conducteur du véhicule B
echo "\n *** Liste des conducteur du véhicule immatriculé {$vB-
>getRegistrationNumber()} ***";
foreach ($vB->getDrivers() as $driver) {
    echo "\n-{$driver->getName()}";
}

echo "\n==== FIN DU TEST EXERCICE 6 ====";
```

① Attention à faire l'association du conducteur à la voiture depuis l'objet possédant (ici, l'objet possédant est le véhicule`)

Sortie :

```
==== DEBUT DU TEST EXERCICE 6 ====

 *** Liste des véhicules de Anna ***
-AAAA
-BBBB
 *** Liste des conducteur du véhicule immatriculé BBBB ***
-Anna
```

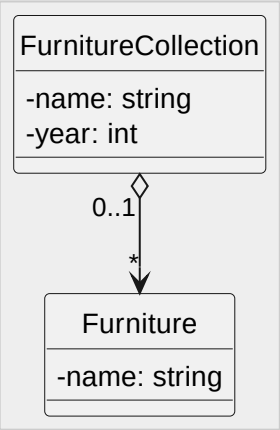
```
-Jalila
==== FIN DU TEST EXERCICE 6 ====
```

1.2. Implémenter une agrégation

Q2) Travail à faire

Un artisan réputé fabrique des meubles de grande valeur et propose chaque année une nouvelle collection. Il lui faut pouvoir gérer les meubles qu'il affecte à chaque collection. Effectivement, chaque meuble fabriqué n'est pas systématiquement placé dans une collection.

- Implémentez le diagramme suivant :



a. Créez une collection nommée "Covidanne" qui sera la collection de meuble pour l'année prochaine.

b. Créez les meubles "Repostar", "Sietanne" et "Letanne"

c. Affectez les meubles à la collection.

d. Depuis la collection, affichez la liste des meubles qui la compose.

e. Depuis un meuble, affichez le nom de la collection dans laquelle il figure.

f. Justifiez ici l'utilisation d'une agrégation plutôt qu'une composition.

Correction de Q2

Rappel : L'agrégation s'implémente comme une association classique.

- Implémentation de la classe `Furniture` :
  Cette classe ne connaît pas la classe `FurnitureCollection` *car nous sommes dans une association unidirectionnelle de* `FurnitureCollection` *vers `Furniture`*

```
1 class Furniture
2 {
```

```
    public function __construct(
        private string $name
    )
    {
    }

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }

    /**
     * @param string $name
     *
     * @return Furniture
     */
    public function setName(string $name): Furniture
    {
        $this->name = $name;

        return $this;
    }

}
```

- Implémentation de la classe `FurnitureCollection` :
  Cette classe connaît la classe `Furniture` car nous sommes dans une association unidirectionnelle de `FurnitureCollection` vers `Furniture`

```
class FurnitureCollection
{


    public function __construct(
        private string $name,
        private int $year,
        private array $furnitures = [] ①
    ) {
    }

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }
```

```php

    /**
     * @param string $name
     *
     * @return FurnitureCollection
     */
    public function setName(string $name): FurnitureCollection
    {
        $this->name = $name;

        return $this;
    }

    /**
     * @return int
     */
    public function getYear(): int
    {
        return $this->year;
    }

    /**
     * @param int $year
     *
     * @return FurnitureCollection
     */
    public function setYear(int $year): FurnitureCollection
    {
        $this->year = $year;

        return $this;
    }

    //mutateurs et accesseur de la collection stockées dans l'attribut furnitures ②

    /**
     * @param Furniture $furniture ajoute un item de type Furniture à la
     *                             collection
     */
    public function addFurniture(Furniture $furniture): bool
    {
        if (!in_array($furniture, $this->furnitures, true)) {
            $this->furnitures[] = $furniture;

            return true;
        }

        return false;
    }

```

```php
    /**
     * @param Furniture $furniture retire l'item de la collection
     */
    public function removeFurniture(Furniture $furniture): bool
    {
        $key = array_search($furniture, $this->furnitures, true);

        if ($key !== false) {
            unset($this->furnitures[$key]);

            return true;
        }

        return false;
    }

    /**
     * @return Furniture[]
     */
    public function getFurnitures(): array
    {
        return $this->furnitures;
    }



}
```

- Mise en oeuvre des classes :

```php
//a) Création de la collection

$furnitureCollection = new FurnitureCollection("Covidanne", 2022);

//b) Création des meubles
$repostartFurniture = new Furniture("Repostar");
$sietanneFurniture = new Furniture("Sietanne");
$letanneFurniture = new Furniture("Letanne");

//c) Affectation des meubles à la collection
$furnitureCollection->addFurniture($repostartFurniture);
$furnitureCollection->addFurniture($sietanneFurniture);
$furnitureCollection->addFurniture($letanneFurniture);

//d) Liste des meubles de la collection
echo "\n *** Liste des meubles de la collection {$furnitureCollection->
getName()} ***";
foreach ($furnitureCollection->getFurnitures() as $furniture) {
    echo "\n-{$furniture->getName()}";
}
```

```
20
21 //e) nom de la collection à partir du meuble
22 // IMPOSSIBLE, l'association est unidirectionnelle !
23
24 //f) Il est utilisé une agrégation plutôt qu'une composition car le meuble
   peut exister sans la collection. Dans une composition, le composant n'existe
   pas sans le composite car ce dernier est responsable de son cycle de vie.
```

① initialisation de la collection d'objets de type `Furniture`

② mutateurs et accesseur de la collection

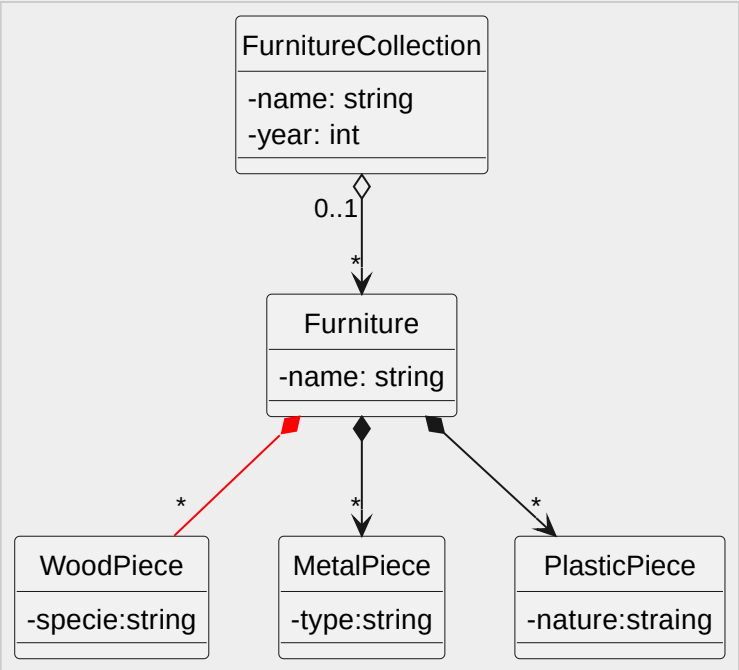
- Sortie :

```
 *** Liste des meubles de la collection Covidanne ***
-Repostar
-Sietanne
-Letanne
```

1.3. Implémenter une composition

Q3)

- Implémentez le diagramme suivant :



- Créez un meuble nommé "BeautyLazy" constitué de 3 pièces de bois (2 pièces en chêne et 1 pièce en hêtre), de 5 pièces de métal (1 en acier, 2 en chrome et 2 en fer) sans plastique.
- Créez un second meuble nommé "Nometal" constitué de 2 pièces de bois (1 pièce en noyer et 1 pièce en hêtre) et de 4 pièces de plastique (1 en PET et 3 en PEHD). Il n'y a pas de métal.

- Affectez les meubles créés à la collection "Covidanne"
- A partir de la collection, lister les meubles (leur nom) avec leur composition conformément à la sortie ci-dessous :

```
Contenu de la collection Covidanne :
Meuble "BeautyLazy" composé des éléments cumulatifs suivants :
- 1 pièce de chêne
- 1 pièce de chêne
- 1 pièce de hêtre
- 1 pièce de acier
- 1 pièce de chrome
- 1 pièce de chrome
- 1 pièce de fer
- 1 pièce de fer
Meuble "Nometal" composé des éléments cumulatifs suivants :
- 1 pièce de noyer
- 1 pièce de hêtre
- 1 pièce de PET
- 1 pièce de PEHD
- 1 pièce de PEHD
- 1 pièce de PEHD
```

Correction de Q3

- La classe `FurnitureCollection` est identique à celle de l'exercice précédent
- Implémentation de la classe `WoodPiece` :

```php
class WoodPiece
{
    /**
     * @param string    $specie    essence de la pièce de bois
     * @param Furniture $furniture meuble associé à la pièce de bois consommée
     */
    public function __construct(
        private string $specie,
        //attribut d'objet permettant la navigation vers Furniture
        private Furniture $furniture
    ) {
    }

    /**
     * @return Furniture meuble dans lequel la pièce de bois courante est
     *                   utilisée
     */
    public function getFurniture(): Furniture
    {
        return $this->furniture;
```

```
21      }
22
23      /**
24       * @return string
25       */
26      public function getSpecie(): string
27      {
28          return $this->specie;
29      }
30
31      /**
32       * @param string $specie
33       *
34       * @return WoodPiece
35       */
36      public function setSpecie(string $specie): WoodPiece
37      {
38          $this->specie = $specie;
39
40          return $this;
41      }
42
43
44  }
```

- Implémentation de la classe `MetalPiece` :

```
1  class MetalPiece
2  {
3
4      public function __construct(
5          private string $type
6      ) {
7      }
8
9      /**
10      * @return string
11      */
12     public function getType(): string
13     {
14         return $this->type;
15     }
16
17     /**
18      * @param string $type
19      *
20      * @return MetalPiece
21      */
22     public function setType(string $type): MetalPiece
23     {
```

```
24         $this->type = $type;
25
26         return $this;
27     }
28
29 }
```

- Implémentation de la classe `PlasticPiece` :

```
 1 class PlasticPiece
 2 {
 3
 4     public function __construct(
 5         private string $nature
 6     ) {
 7     }
 8
 9     /**
10      * @return string
11      */
12     public function getNature(): string
13     {
14         return $this->nature;
15     }
16
17     /**
18      * @param string $nature
19      *
20      * @return PlasticPiece
21      */
22     public function setNature(string $nature): PlasticPiece
23     {
24         $this->nature = $nature;
25
26         return $this;
27     }
28
29 }
```

- Implémentation de la classe `Furniture` :

la solution proposée n'est qu'une solution parmi d'autres. Mais elle respecte au plus près le diagramme qu'il fallait implémenter

```
 1 class Furniture
 2 {
 3     /**
 4      * @var array contient les pièces de bois utiles
 5      */
```

```
6     private array $woodPieces = []; ①
7     /**
8      * @var array contient les pièces de métal utiles
9      */
10    private array $metalPieces = []; ①
11    /**
12     * @var array contient les pièces de plastique utiles
13     */
14    private array $plasticPieces = []; ①
15
16    /**
17     * @param string $name
18     * @param array  $woodPiecesData        tableau contenant les
   informations
19     *                                     sur les pièces de bois à
   utiliser.
20     *                                     Chaque essence de bois différente
21     *                                     sera une entrée dans le tableau
   des
22     *                                     informations :
23     *                                     [['quantity'=> x,'specie'=>'xxx']
24     *                                     ['quantity'=> y,'specie'=>'yyy']
25     *                                     ['quantity'=> z,'specie'=>'zzz']]
26     */
27    public function __construct(
28        private string $name,
29        array $woodPiecesData = [], ②
30        array $metalPiecesData = [], ②
31        array $plasticPiecesData = [] ②
32    ) {
33
34        //création des composants en bois ③
35        //on parcourt le tableau qui contient les tableaux
36        foreach ($woodPiecesData as $woodPieceData) {
37            //pour chaque entrée du tableau, on crée le nombre de pièces de
   bois dans l'essence spécifiée
38            for ($i = 0; $i < $woodPieceData['quantity']; $i++) {
39
40                //en créant la pièce de bois, on passe une instance du meuble
   qu'il concerne ce qui met à jour l'objet inverse de type "Furniture" contenu
   dans "WoodPiece" ④
41                $woodPiece = new WoodPiece($woodPieceData['specie'], $this);
42
43                $this->woodPieces[] = $woodPiece;
44            }
45        }
46
47        //création des composants en métal ③
48        foreach ($metalPiecesData as $metalPieceData) {
49            for ($i = 0; $i < $metalPieceData['quantity']; $i++) {
50                $this->metalPieces[] = new WoodPiece(
```

```
51                      $metalPieceData['type'],
52                      $this
53                  );
54              }
55          }
56
57          //création des composants en plastique ③
58          foreach ($plasticPiecesData as $plasticPieceData) {
59              for ($i = 0; $i < $plasticPieceData['quantity']; $i++) {
60                  $this->plasticPieces[] = new PlasticPiece(
61                      $plasticPieceData['nature']
62                  );
63              }
64          }
65
66
67      }
68
69      //accesseur pour chaque matière utilisée dans le meuble
70
71      /**
72       * @return array
73       */
74      public function getWoodPieces(): array
75      {
76          return $this->woodPieces;
77      }
78
79      /**
80       * @return array
81       */
82      public function getMetalPieces(): array
83      {
84          return $this->metalPieces;
85      }
86
87      /**
88       * @return array
89       */
90      public function getPlasticPieces(): array
91      {
92          return $this->plasticPieces;
93      }
94
95      //autres mutateurs et setters
96
97
98      /**
99       * @return string
100      */
101     public function getName(): string
```

```
102     {
103         return $this->name;
104     }
105 
106     /**
107      * @param string $name
108      *
109      * @return Furniture
110      */
111     public function setName(string $name): Furniture
112     {
113         $this->name = $name;
114 
115         return $this;
116     }
117 
118 
119 }
```

① Initialisation de la collection

② Pour laisser la responsabilité de la création des composants à la classe composite, la quantité de chaque pièce de bois et le nom de l'essence sont passées en argument. Les composants sont ainsi créés **dans le composite**. Ces composants ne doivent pas être accessibles depuis l'extérieur de la classe composite d'où l'absence d'accesseur les concernant.

③ Les pièces de bois, de métal et de plastique sont créées depuis la classe `Furniture` car il s'agit d'une composition. Elle est responsable du cycle de vie de ses composants.

④ On profite de l'instanciation de la pièce de bois pour lui affecter le meuble courant que l'on est en train de créer. Si cette solution fonctionne en PHP, il n'est pas garanti que cela fonctionne avec tous les langages objet.

- Mise en oeuvre qui permet d'obtenir la sortie attendue :

```
1 //création du meuble BeautyLazy avec les éléments spécifiés
2 $woodPiecesInfo = [
3     ['specie' => "chêne", 'quantity' => 2],
4     ['specie' => "hêtre", 'quantity' => 1],
5 ];
6 
7 $metalPiecesInfo = [
8     ['type' => "acier", 'quantity' => 1],
9     ['type' => "chrome", 'quantity' => 2],
10     ['type' => "fer", 'quantity' => 2],
11 ];
12 
13 $beautyLazyFurniture = new Furniture(
14     "BeautyLazy",
15     $woodPiecesInfo,
16     $metalPiecesInfo,
```

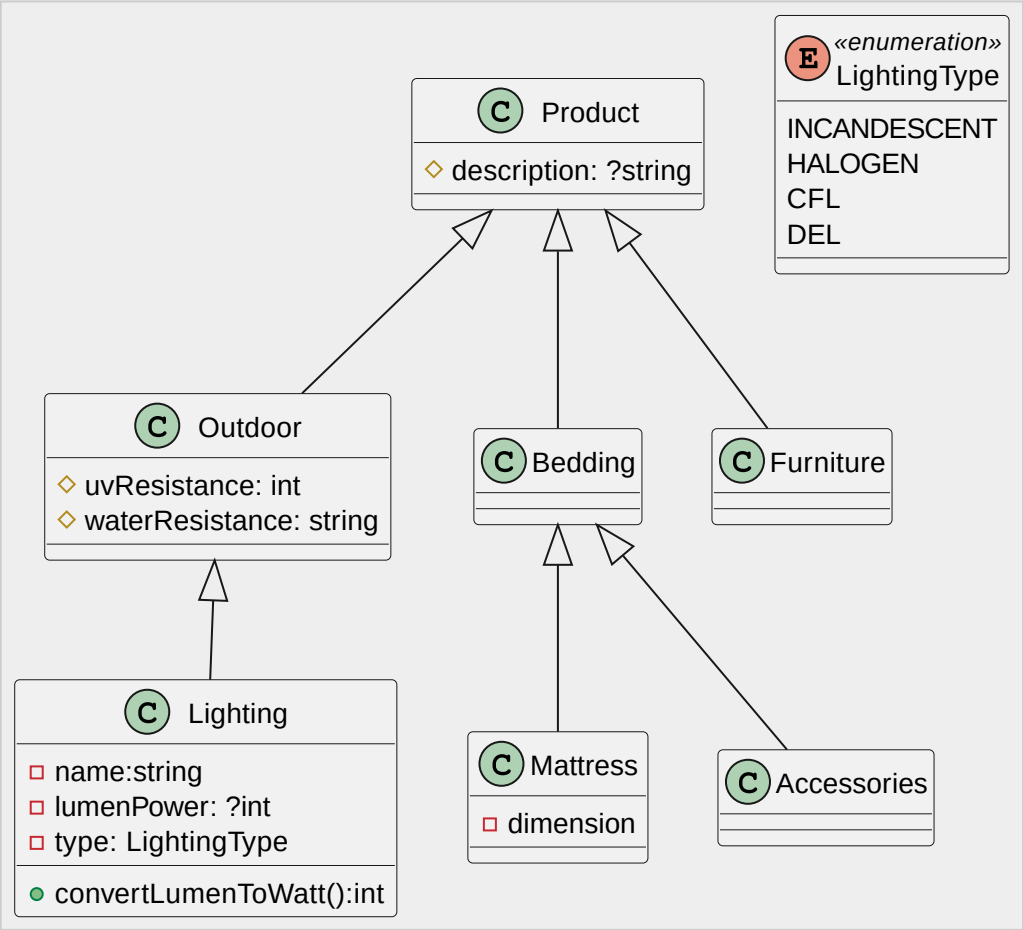
```php
    []
);

//création du meuble Nometal avec les éléments spécifiés
$woodPiecesInfo = [
    ['specie' => "noyer", 'quantity' => 1],
    ['specie' => "hêtre", 'quantity' => 1],
];

$metalPiecesInfo = [];

$plasticPiecesInfo = [
    ['nature' => "PET", 'quantity' => 1],
    ['nature' => "PEHD", 'quantity' => 3],
];

$nometalFurniture = new Furniture(
    "Nometal",
    $woodPiecesInfo,
    $metalPiecesInfo,
    $plasticPiecesInfo
);

//création de la collection
$furnitureCollection = new FurnitureCollection("Covidanne", 2022);

//affectation des meubles à la collection
$furnitureCollection->addFurniture($beautyLazyFurniture);
$furnitureCollection->addFurniture($nometalFurniture);

echo "`\nContenu de la collection {$furnitureCollection->getName()} :";

//liste des meubles (avec leur composition) à partir de la collection
foreach ($furnitureCollection->getFurnitures() as $furniture) {

    echo "\nMeuble \"{$furniture->getName()}\" composé des éléments cumulatifs suivants :";
    foreach ($furniture->getWoodPieces() as $woodPiece) {
        echo "\n- 1 pièce de {$woodPiece->getSpecie()}";
    }
    foreach ($furniture->getMetalPieces() as $metalPiece) {
        echo "\n- 1 pièce de {$metalPiece->getSpecie()}";
    }
    foreach ($furniture->getPlasticPieces() as $plasticPiece) {
        echo "\n- 1 pièce de {$plasticPiece->getNature()}";
    }


}
```

1.4. Implémenter un héritage

Q4)

- Implémentez le diagramme suivant :

Pourl'implémentation du type ENUM et de son utilisation, veuillez prendre connaissance de la documentation sur le type `enum`.



Correction de Q4

- Implémentation de la classe `Furniture` :

```php
class Furniture
{
    public function __construct(
        private string $name,
        private ?WoodPiece $woodPiece,
        private ?MetalPiece $metalPiece,
        private ?PlasticPiece $plasticPiece,
    )
    {

```

```php
    }

    /**
     * @return array
     */
    public function getMeaterialPieces(): array ①
    {
        return $this->MeaterialPieces;
    }

    /**
     * @param array $MeaterialPieces
     *
     * @return Furniture
     */
    public function setMeaterialPieces(array $MeaterialPieces): Furniture ①
    {
        $this->MeaterialPieces = $MeaterialPieces;

        return $this;
    }


    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }

    /**
     * @param string $name
     *
     * @return Furniture
     */
    public function setName(string $name): Furniture
    {
        $this->name = $name;

        return $this;
    }

}
```

① gestion de la collection qui permet de naviguer vers les isntances de `MaterialPiece`

- Implémentation de la classe `MaterialPiece` :

```
class MaterielPiece
{
```

```
    public function __construct(
        protected int $quantityByPiece
    ) {
    }

    /**
     * @return int
     */
    public function getQuantityByPiece(): int
    {
        return $this->quantityByPiece;
    }

    /**
     * @param int $quantityByPiece
     *
     * @return MaterielPiece
     */
    public function setQuantityByPiece(int $quantityByPiece): MaterielPiece
    {
        $this->quantityByPiece = $quantityByPiece;

        return $this;
    }
}
```

- Implémentation de la classe `WoodPiece` :

```

```

- Implémentation de la classe `MetalPiece` :

```
class MetalPiece  extends MaterielPiece
{

    public function __construct(
        private string $type
    ) {
    }

    /**
     * @return string
     */
    public function getType(): string
    {
        return $this->type;
    }
```

```
    /**
     * @param string $type
     *
     * @return MetalPiece
     */
    public function setType(string $type): MetalPiece
    {
        $this->type = $type;

        return $this;
    }

}
```

- Implémentation de la classe `PlasticPiece` :

```
class PlasticPiece extends MaterielPiece
{

    public function __construct(
        private string $nature
    ) {

    }

    /**
     * @return string
     */
    public function getNature(): string
    {
        return $this->nature;
    }

    /**
     * @param string $nature
     *
     * @return PlasticPiece
     */
    public function setNature(string $nature): PlasticPiece
    {
        $this->nature = $nature;

        return $this;
    }

}
```

- Mise en oeuvre des classes

- Sortie

1.5. Implémenter une classe association

A faire : exercice avec une classe association

Index