

UML

Baptiste Bauer

Version v0.0.8.sip-221124100530, 2022-11-23 09:32:18

Table des matières

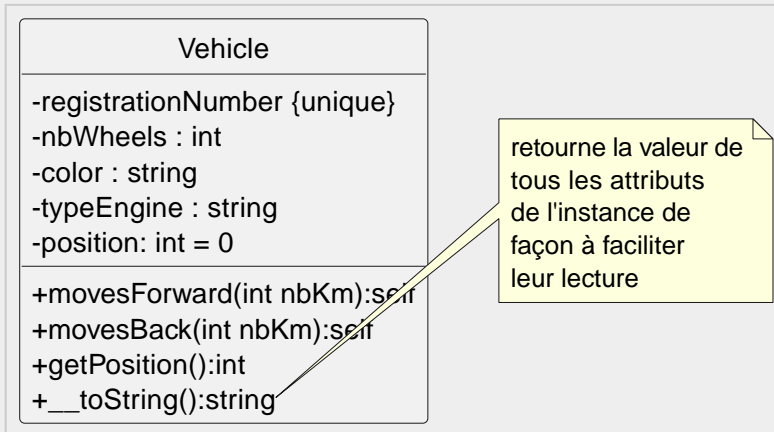
1. Quelques exercices d'implémentation	1
1.1. Implémenter une simple classe	1
1.2. Implémenter une association unidirectionnelle simple	5
1.3. Implémenter une association unidirectionnelle multiple avec cardinalité minimum à 0	9
1.4. Implémenter une association unidirectionnelle multiple avec cardinalité minimum à 1	14
1.5. Implémenter une association bidirectionnelle one-to-many	19

1. Quelques exercices d'implémentation

1.1. Implémenter une simple classe

Q1) Travail à faire

- Implémentez la classe suivante :



La position du véhicule ne doit pas pouvoir être affectée arbitrairement. Elle doit découler de la position initiale et de ses déplacements.

- Testez votre implémentation en créant deux véhicules différents et en les faisant avancer et reculer différemment, puis affichez les informations de chaque véhicule en mobilisant la méthode `__toString`.

Correction de Q1

```
1 <?php
2
3 //selon les recommandations PHP, lorsque la méthode __toString est utilisée,
  il faut implémenter l'interface Stringable.
4 class Vehicle implements Stringable
5 {
6
7     //Déclaration des variables d'objet
8     /**
9      * @var string numéro d'immatriculation du véhicule
10     */
11     private string $registrationNumber;
12     /**
13      * @var int nombre de roues
14     */
15     private int $nbWheels;
16     /**
```

```
17     * @var string couleur du véhicule
18     */
19     private string $color;
20     /**
21     * @var string type de moteur (essence, diesel, électrique, hydrogène,
22     ...)
23     */
24     private string $typeEngine;
25     /**
26     * @var int Position du véhicule
27     */
28     private int $position = 0;
29     //le constructeur aurait pu être omis puisqu'il n'exécute aucun code
30     public function __construct()
31     {
32     }
33
34
35
36     //region ***** Mutateurs et accesseurs *****
37
38     /**
39     * @return string
40     */
41     public function getRegistrationNumber(): string
42     {
43         return $this->registrationNumber;
44     }
45
46     /**
47     * @return int
48     */
49     public function getNbWheels(): int
50     {
51         return $this->nbWheels;
52     }
53
54     /**
55     * @return string
56     */
57     public function getColor(): string
58     {
59         return $this->color;
60     }
61
62     /**
63     * @return string
64     */
65     public function getTypeEngine(): string
66     {
```

```
67     return $this->typeEngine;
68 }
69
70 /**
71  * @return int
72  */
73 public function getPosition(): int
74 {
75     return $this->position;
76 }
77
78 /**
79  * @param string $registrationNumber
80  *
81  * @return Vehicle
82  */
83 public function setRegistrationNumber(string $registrationNumber):
Vehicle
84 {
85     $this->registrationNumber = $registrationNumber;
86
87     return $this;
88 }
89
90 /**
91  * @param int $nbWheels
92  *
93  * @return Vehicule
94  */
95 public function setNbWheels(int $nbWheels): self
96 {
97     $this->nbWheels = $nbWheels;
98
99     return $this;
100 }
101
102 /**
103  * @param string $color
104  *
105  * @return Vehicule
106  */
107 public function setColor(string $color): self
108 {
109     $this->color = $color;
110
111     return $this;
112 }
113
114 /**
115  * @param string $typeEngine
116  *
```

```
117     * @return Vehicule
118     */
119     public function setTypeEngine(string $typeEngine): self
120     {
121         $this->typeEngine = $typeEngine;
122
123         return $this;
124     }
125
126     //endregion ***** Mutateurs et accesseurs *****
127
128
129     //region ***** Autres opérations *****
130
131     public function __toString(): string
132     {
133         return "Immatriculation : {$this->registrationNumber} ; Nombre de
roues : {$this->nbWheels} ; couleur : {$this->color} ; type de moteur {$this
->typeEngine} ; position : {$this->position}";
134     }
135
136
137     public function moveForward(int $nbKms): self
138     {
139         $this->position += $nbKms;
140
141         return $this;
142     }
143
144     public function moveBack(int $nbKms): self
145     {
146         $this->position -= $nbKms;
147
148         return $this;
149     }
150
151     //endregion ***** Autres opérations *****
152
153
154 }
155
156 $v1 = new Vehicle();
157 $v1->setRegistrationNumber('AAAA')->setNbWheels(4)->setColor('Jaune')-
>setTypeEngine('diesel');
158 $v1->moveForward(100)->moveForward(50)->moveBack(30);
159
160 $v2 = new Vehicle();
161 $v2->setRegistrationNumber('BBBB')->setNbWheels(2)->setColor('Rouge')-
>setTypeEngine('essence');
162 $v2->moveForward(230)->moveBack(50)->moveBack(10)->moveForward(140);
163
```

```

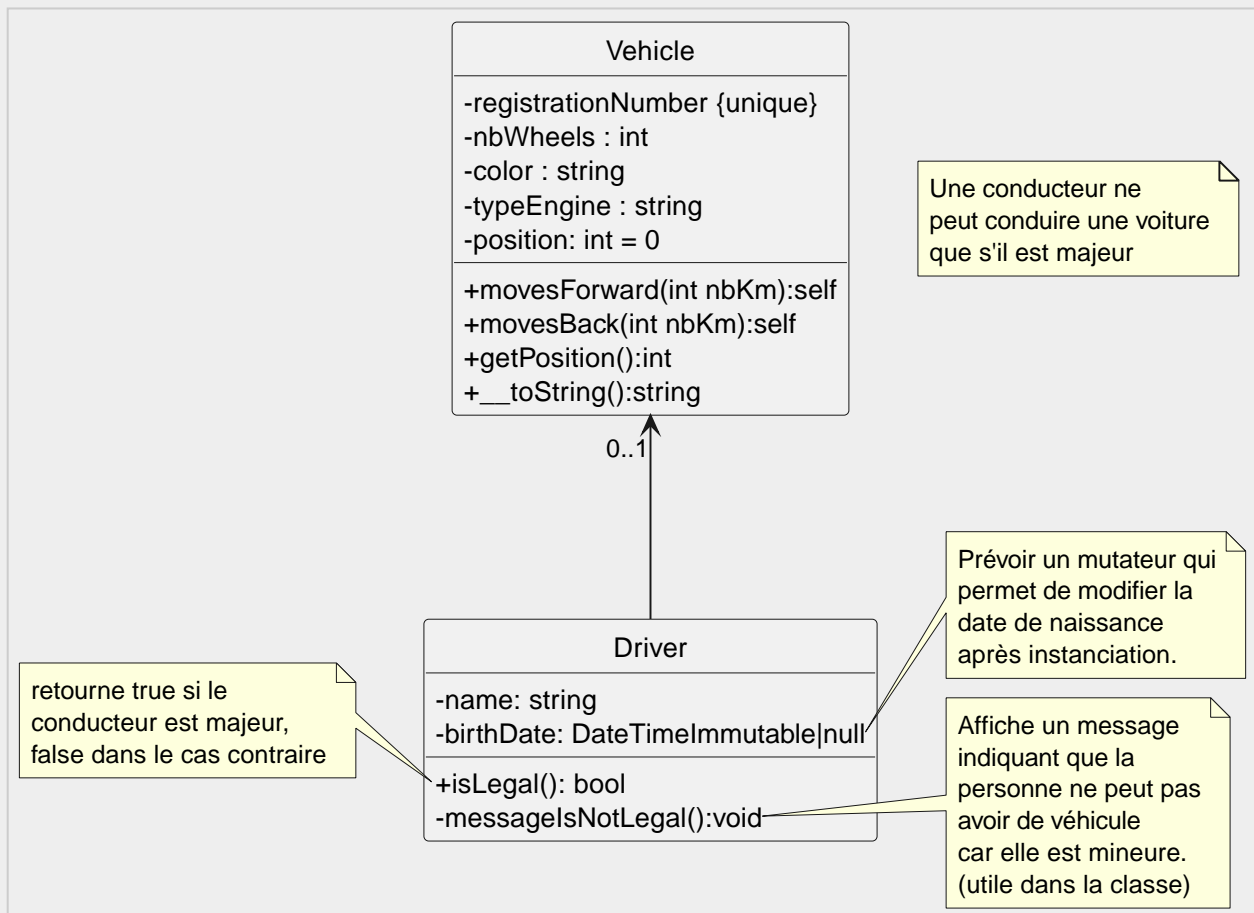
164 //mobilisation de la méthode toString en affichant directement les objets
    avec echo (normalement, il n'est pas possible d'afficher un élément non
    scalaire (une chaîne, un entier) de cette façon)
165 echo "==== DEBUT DU TEST EXERCICE 1 ==== \n";
166 echo "Voiture 1 : $v1";
167 echo "\n";
168 echo "Voiture 1 : $v2";
169 echo "\n==== FIN DU TEST EXERCICE 1 =====";

```

1.2. Implémenter une association unidirectionnelle simple

Q2) Travail à faire

- Implémentez le diagramme suivant :



Si un véhicule est associé à un conducteur mineur, l'association ne doit pas se faire. Un message devra s'afficher indiquant que la personne est mineure et qu'il n'est pas possible d'associer un véhicule à un mineur. Attention à la situation qui consisterait à changer la date de naissance après que la personne se soit vue affecter un véhicule.

- Testez votre implémentation en essayant d'affecter un conducteur mineur à un premier

véhicule et un autre conducteur à un second véhicule qui avance et recule selon votre bon vouloir.

- Donnez via le code la position du véhicule du conducteur majeur depuis la variable qui référence celui-ci.

Correction de Q2

```
1 <?php
2
3 //inclusion de la classe Vehicle
4 include_once 'exo-1-correction.php';
5
6 /**
7  * Navigabilité unidirectionnelle de Driver vers Vehicle
8  */
9 class Driver
10 {
11     /**
12      * @param string          $name      nom du conducteur
13      * @param DateTimeImmutable $birthDate date de naissance du conducteur
14      * @param Vehicle|null     $vehicle   instance du véhicule associé au
15      *                                     conducteur
16      */
17     public function __construct(
18         private string $name,
19         private DateTimeImmutable $birthDate,
20         private ?Vehicle $vehicle = null
21     ) {
22
23     }
24
25     //region ***** mutateurs et accesseurs *****
26
27     /**
28      * @return string
29      */
30     public function getName(): string
31     {
32         return $this->name;
33     }
34
35     /**
36      * @param string $name
37      *
38      * @return Driver
39      */
40     public function setName(string $name): self
41     {
```



```
42     $this->name = $name;
43
44     return $this;
45 }
46
47 /**
48  * @return DateTimeImmutable
49  */
50 public function getBirthDate(): DateTimeImmutable
51 {
52     return $this->birthDate;
53 }
54
55 /**
56  * @param DateTimeImmutable $birthDate
57  *
58  * @return Driver
59  */
60 public function setBirthDate(DateTimeImmutable $birthDate): self
61 {
62     $this->birthDate = $birthDate;
63
64     //si jamais la personne change sa date de naissance pour une date qui
    la conduit à être mineure, alors il faut désassocier le véhicule
65     if (!$this->isLegal()) {
66         $this->messageIsNotLegal();
67         $this->vehicle = null;
68     }
69
70     return $this;
71 }
72
73 /**
74  * @return Vehicle|null
75  */
76 public function getVehicle(): ?Vehicle
77 {
78     return $this->vehicle;
79 }
80
81 /**
82  * @param Vehicle|null $vehicle
83  *
84  * @return Driver
85  */
86 public function setVehicle(?Vehicle $vehicle): self
87 {
88     //si le conducteur est majeur, on peut lui affecter un véhicule,
    sinon on ne lui affecte pas
89     if ($this->isLegal()) {
90         $this->vehicle = $vehicle;
```

```
91     } else {
92         $this->messageIsNotLegal();
93         $this->vehicle = null;
94     }
95
96     return $this;
97 }
98
99 private function messageIsNotLegal(): void
100 {
101     echo "\n La personne est mineure, il n'est pas possible de lui
associer un véhicule.\n";
102
103 }
104
105 //endregion ***** mutateurs et accesseurs *****
106
107 //region ***** Autres opérations *****
108
109 public function isLegal(): bool
110 {
111     //date courante
112     $now = new DateTimeImmutable(
113     ); //j'aurais pu prendre une date de type DateTime
114     //calcul de la différence entre la date du jour et la date de
naissance
115     //documentation : https://www.php.net/manual/fr/datetime.diff.php
116     $interval = $this->birthDate->diff($now);
117     //cet interval est un objet, il faut retourner l'écart en nombre
d'années
118     $years = (int)$interval->format('%r%y');
119
120     return $years > 18;
121 }
122
123
124 //endregion ***** Autres opérations *****
125 }
126
127 echo "\n\n==== DEBUT DU TEST EXERCICE 2 ==== \n";
128
129 $driverPaul = new Driver('Paul', new DateTimeImmutable('2017-04-12'));
130 $driverPaul->setVehicle($v1);
131 $driverJuliette = new Driver('Juliette', new DateTimeImmutable('2000-02-
24'));
132 $driverJuliette->setVehicle($v1);
133 $driverJuliette->getVehicle()->moveForward(100)->moveBack(40);
134 //on donne la position du véhicule depuis l'objet "Juliette"
135 echo "\n La position du véhicule de {$driverJuliette->getName()} est
{$driverJuliette->getVehicle()->getPosition()}";
```

```
136 echo "\n==== FIN DU TEST EXERCICE 2 ====";
```

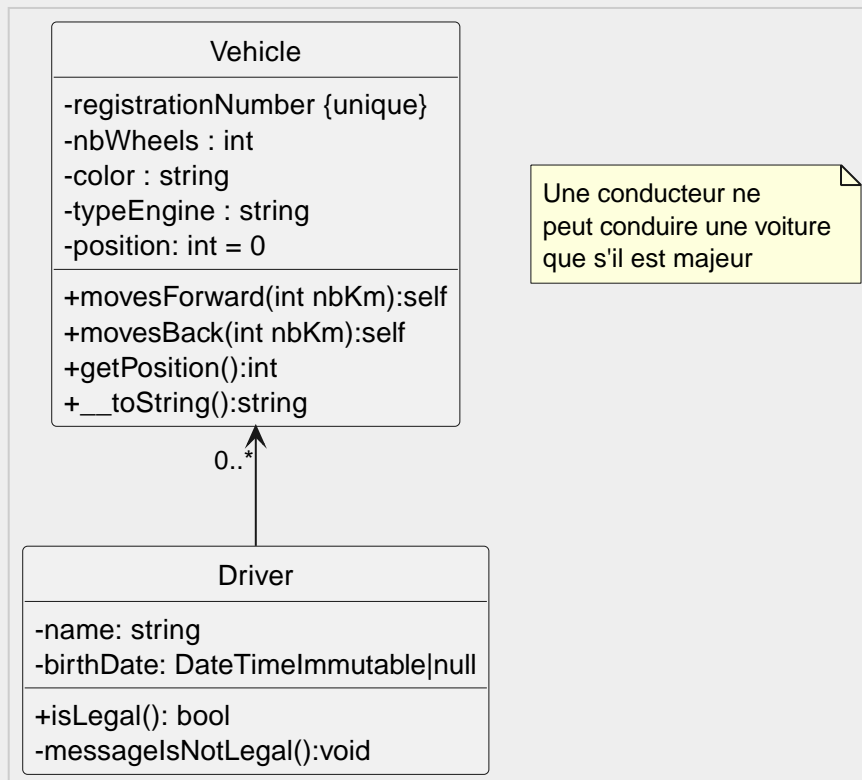
Sortie console :

```
1 ==== DEBUT DU TEST EXERCICE 2 ====
2
3 La personne est mineure, il n'est pas possible de lui associer un véhicule.
4
5 La position du véhicule de Juliette est 180
6 ==== FIN DU TEST EXERCICE 2 ====
```

1.3. Implémenter une association unidirectionnelle multiple avec cardinalité minimum à 0

Q3) Travail à faire

- Implémentez le diagramme suivant :



- Créez le conducteur Paul dont la date de naissance le conduit à être majeur
- Créez la conductrice Juliette dont la date de naissance la conduit à être mineure
- Créez les véhicules A, B, C, D et E avec les caractéristiques de votre choix
- Tentez d'affecter les véhicules A, C et E à Paul
- Tentez d'affecter les véhicules A et B à Juliette

- Faire avancer les véhicules A et C respectivement de 120km et 84km
- Faire reculer le véhicule C de 25km
- Afficher la liste des véhicules affectés à Paul
- Afficher la liste des véhicules affectés à Juliette
- Retirez les véhicules A et E à Paul
- Afficher la liste des véhicules restant à Paul

Correction de Q3

```

1 <?php
2
3 //inclusion de la classe Vehicle
4 include_once 'exo-1-correction.php';
5
6 /**
7  * Navigabilité unidirectionnelle de Driver vers Vehicle
8  */
9 class Driver
10 {
11     /**
12      * @param string          $name          nom du conducteur
13      * @param DateTimeImmutable $birthDate date de naissance du conducteur
14      * @param array|Vehicle[] $vehicles      collection d'objets de type
Vehicle
15      */
16     public function __construct(
17         private string $name,
18         private DateTimeImmutable $birthDate,
19         private array $vehicles = [] ①
20     ) {
21
22     }
23
24     //region ***** mutateurs et accesseurs *****
25
26     /**
27      * @return string
28      */
29     public function getName(): string
30     {
31         return $this->name;
32     }
33
34     /**
35      * @param string $name
36      *

```

```
37     * @return Driver
38     */
39     public function setName(string $name): self
40     {
41         $this->name = $name;
42
43         return $this;
44     }
45
46     /**
47     * @return DateTimeImmutable
48     */
49     public function getBirthDate(): DateTimeImmutable
50     {
51         return $this->birthDate;
52     }
53
54     /**
55     * @param DateTimeImmutable $birthDate
56     *
57     * @return Driver
58     */
59     public function setBirthDate(DateTimeImmutable $birthDate): self
60     {
61         $this->birthDate = $birthDate;
62
63         //si jamais la personne change sa date de naissance pour une date qui
        //la conduit à être mineure, alors il faut désassocier les véhicules
64         if (!$this->isLegal()) {
65             $this->messageIsNotLegal();
66             $this->vehicle = [];
67         }
68
69         return $this;
70     }
71
72     /**
73     * @param Vehicle $vehicle ajoute un item de type Vehicle à la collection
74     */
75     public function addVehicle(Vehicle $vehicle): bool
76     {
77         //tester si le conducteur est mineur
78         if (!$this->isLegal()) {
79             $this->messageIsNotLegal();
80             return false;
81         }
82         if (!in_array($vehicle, $this->vehicles, true)) {
83             $this->vehicles[] = $vehicle;
84
85             return true;
86         }
87     }
```

```

87
88     return false;
89 }
90
91 /**
92  * @param Vehicle $vehicle retire l'item de la collection
93  */
94 public function removeVehicle(Vehicle $vehicle): bool
95 {
96     $key = array_search($vehicle, $this->vehicles, true);
97
98     if ($key !== false) {
99         unset($this->vehicles[$key]);
100
101         return true;
102     }
103
104     return false;
105 }
106
107 /**
108  * @return Vehicle[]
109  */
110 public function getVehicles(): array
111 {
112     return $this->vehicles;
113 }
114
115 private function messageIsNotLegal(): void
116 {
117     echo "\n La personne est mineure, il n'est pas possible de lui
associer un véhicule.\n";
118 }
119 }
120
121 //endregion ***** mutateurs et accesseurs *****
122
123 //region ***** Autres opérations *****
124
125 public function isLegal(): bool
126 {
127     //date courante
128     $now = new DateTimeImmutable(
129     ); //j'aurais pu prendre une date de type DateTime
130     //calcul de la différence entre la date du jour et la date de
naissance
131     //documentation : https://www.php.net/manual/fr/datetime.diff.php
132     $interval = $this->birthDate->diff($now); ②
133     //cet intervalle est un objet, il faut retourner l'écart en nombre
d'années
134     $years = (int)$interval->format('%r%y'); ③

```

```
135
136     return $years > 18;
137
138 }
139
140 //endregion ***** Autres opérations *****
141 }
142
143 echo "\n\n==== DEBUT DU TEST EXERCICE 3 ==== \n";
144
145 //création des conducteurs
146 $driverPaul = new Driver('Paul', new DateTimeImmutable('2000-04-12'));
147 $driverJuliette = new Driver('Juliette', new DateTimeImmutable('2019-02-
    24'));
148
149 //création des véhicules
150 $vA = (new Vehicle())->setRegistrationNumber('AAAA');
151 $vB = (new Vehicle())->setRegistrationNumber('BBBB');
152 $vC = (new Vehicle())->setRegistrationNumber('CCCC');
153 $vD = (new Vehicle())->setRegistrationNumber('DDDD');
154 $vE = (new Vehicle())->setRegistrationNumber('EEEE');
155
156 //affectation des véhicules à Paul
157 $driverPaul->addVehicle($vA);
158 $driverPaul->addVehicle($vB);
159 $driverPaul->addVehicle($vC);
160 $driverPaul->addVehicle($vD);
161
162 //affectation des véhicules à Juliette
163 $driverJuliette->addVehicle($vA);
164 $driverJuliette->addVehicle($vB);
165
166 //déplacement des véhicules
167 $vA->moveForward(120);
168 $vC->moveForward(84);
169 $vC->moveBack(25);
170
171 //liste des véhicules conduits par Paul
172 echo "\n *** Liste des véhicules de {$driverPaul->getName()} ***";
173 foreach ($driverPaul->getVehicles() as $vehicle) {
174     echo "\n-{$vehicle->getRegistrationNumber()}";
175 }
176
177 //liste des véhicules conduits par Juliette
178 echo "\n *** Liste des véhicules de {$driverJuliette->getName()} ***";
179 foreach ($driverJuliette->getVehicles() as $vehicle) {
180     echo "\n-{$vehicle->getRegistrationNumber()}";
181 }
182
183 //suppression des véhicules A et E pour Paul
184 $driverPaul->removeVehicle($vA);
```

```

185 $driverPaul->removeVehicle($vE);
186
187 //liste des véhicules restant à Paul
188 echo "\n *** Liste des véhicules restant à {$driverPaul->getName()} ***";
189 foreach ($driverPaul->getVehicles() as $vehicle ){
190     echo "\n-{$vehicle->getRegistrationNumber()}";
191 }
192
193 echo "\n==== FIN DU TEST EXERCICE 3 ====";

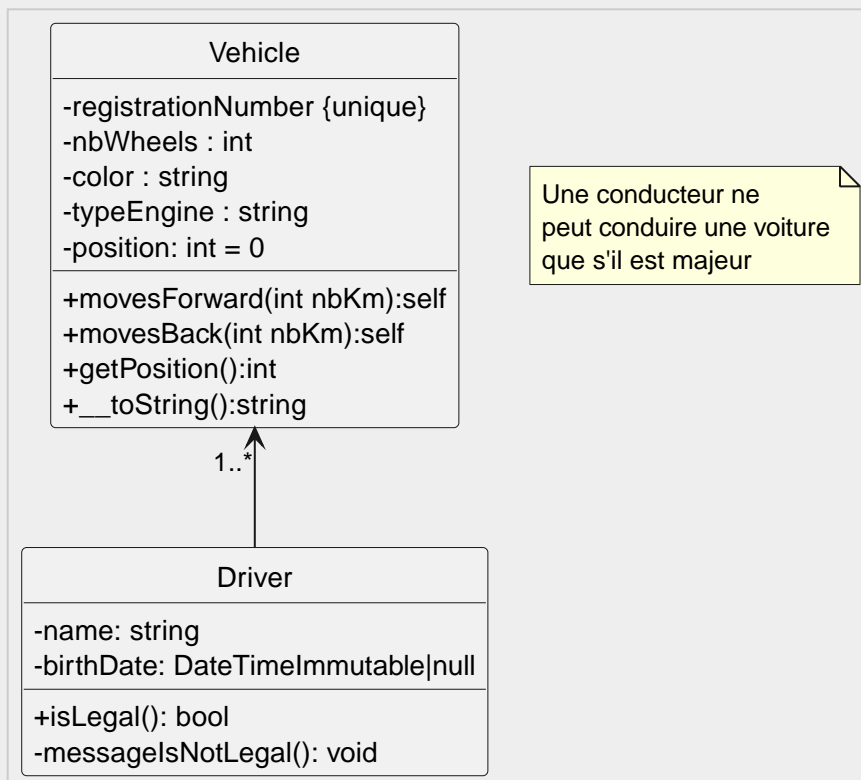
```

- ① Ne pas oublier d'initialiser la collection
- ② La méthode `diff` sur un objet de type `DateTimeInterface` permet de calculer un écart entre deux dates.
- ③ Formate l'intervalle de façon à retourner le nombre d'années sous forme de chaîne (qui est ensuite casté pour être un entier)

1.4. Implémenter une association unidirectionnelle multiple avec cardinalité minimum à 1

Q4) Travail à faire

- Implémentez le diagramme suivant :



- Créez le conducteur Paul dont la date de naissance le conduit à être majeur. Il conduira le véhicule A.
- Affectez un véhicule supplémentaire B à Paul

- Créez un véhicule C (sans conducteur)
- Faire avancer les véhicules A et C respectivement de 123km et 257km
- Faire reculer le véhicule A de 70km
- Afficher depuis la variable référençant Paul la liste de ses véhicules et leur position.
- Retirez les véhicules A et C à Paul
- Afficher depuis la variable référençant Paul la liste de ses véhicules et leur position.

Correction de Q4

```

1 <?php
2
3 //inclusion de la classe Vehicle
4 include_once 'exo-1-correction.php';
5
6 /**
7  * Navigabilité unidirectionnelle de Driver vers Vehicle
8  */
9 class Driver
10 {
11     /**
12      * @var array|Vehicle[] $vehicles collection d'objets de type Vehicle
13      */
14     private array $vehicles = []; //ne pas oublier d'initialiser la
        collection !
15
16     /**
17      * @param string          $name      nom du conducteur
18      * @param DateTimeImmutable $birthDate date de naissance du conducteur
19      * @param Vehicle          $vehicle   un objet de type Vehicle (ce n'est pas
        une variable d'objet !)
20      */
21     public function __construct(
22         private string $name,
23         private DateTimeImmutable $birthDate,
24         Vehicle $vehicle ①
25     ) {
26         //conformément au diagramme, on associe un véhicule dès la création
        du conducteur
27         $this->addVehicle($vehicle); ②
28     }
29
30     //region ***** mutateurs et accesseurs *****
31
32     /**
33      * @return string
34      */

```

```
35     public function getName(): string
36     {
37         return $this->name;
38     }
39
40     /**
41      * @param string $name
42      *
43      * @return Driver
44      */
45     public function setName(string $name): self
46     {
47         $this->name = $name;
48
49         return $this;
50     }
51
52     /**
53      * @return DateTimeImmutable
54      */
55     public function getBirthDate(): DateTimeImmutable
56     {
57         return $this->birthDate;
58     }
59
60     /**
61      * @param DateTimeImmutable $birthDate
62      *
63      * @return Driver
64      */
65     public function setBirthDate(DateTimeImmutable $birthDate): self
66     {
67         $this->birthDate = $birthDate;
68
69         //si jamais la personne change sa date de naissance pour une date qui
        //la conduit à être mineure, alors il faut désassocier les véhicules
70         if (!$this->isLegal()) {
71             $this->messageIsNotLegal();
72             $this->vehicle = [];
73         }
74
75         return $this;
76     }
77
78     /**
79      * @param Vehicle $vehicle ajoute un item de type Vehicle à la collection
80      */
81     public function addVehicle(Vehicle $vehicle): bool
82     {
83         //tester si le conducteur est mineur
84         if (!$this->isLegal()){
```

```
85         $this->messageIsNotLegal();
86         return false;
87     }
88     if (!in_array($vehicle, $this->vehicles, true)) {
89         $this->vehicles[] = $vehicle;
90
91         return true;
92     }
93
94     return false;
95 }
96
97 /**
98  * @param Vehicle $vehicle retire l'item de la collection
99  */
100 public function removeVehicle(Vehicle $vehicle): bool
101 {
102     //s'il ne reste plus qu'un véhicule, il ne faut pas le retirer
    conformément au diagramme
103     if(count($this->vehicles) === 1){ ③
104         return false;
105     }
106
107     $key = array_search($vehicle, $this->vehicles, true);
108
109     if ($key !== false) {
110         unset($this->vehicles[$key]);
111
112         return true;
113     }
114
115     return false;
116 }
117
118 /**
119  * @return Vehicle[]
120  */
121 public function getVehicles(): array
122 {
123     return $this->vehicles;
124 }
125
126 private function messageIsNotLegal(): void
127 {
128     echo "\n La personne est mineure, il n'est pas possible de lui
    associer un véhicule.\n";
129 }
130
131
132 //endregion ***** mutateurs et accesseurs *****
133
```

```
134 //region ***** Autres opérations *****
135
136 public function isLegal(): bool
137 {
138     //date courante
139     $now = new DateTimeImmutable(
140     ); //j'aurais pu prendre une date de type DateTime
141     //calcul de la différence entre la date du jour et la date de
naissance
142     //documentation : https://www.php.net/manual/fr/datetime.diff.php
143     $interval = $this->birthDate->diff($now);
144     //cet interval est un objet, il faut retourner l'écart en nombre
d'années
145     $years = (int)$interval->format('%r%y');
146
147     return $years > 18;
148 }
149
150
151 //endregion ***** Autres opérations *****
152 }
153
154 echo "\n\n==== DEBUT DU TEST EXERCICE 4 ==== \n";
155
156 //Création du véhicule A
157 $vA = (new Vehicle())->setRegistrationNumber('AAAA');
158
159 //création de Paul
160 $driverPaul = new Driver('Paul', new DateTimeImmutable('2000-04-12'), $vA);
161
162 //Création du véhicule B
163 $vB = (new Vehicle())->setRegistrationNumber('BBBB');
164 //affectation du véhicule B à Paul
165 $driverPaul->addVehicle($vB);
166 //Création du véhicule C
167 $vC = (new Vehicle())->setRegistrationNumber('CCCC');
168 //circulation des véhicules A et C
169 $vA->moveForward(123);
170 $vC->moveForward(257);
171 $vA->moveBack(70);
172
173 //liste des véhicules conduits par Paul
174 echo "\n *** Liste des véhicules de {$driverPaul->getName()} ***";
175 foreach ($driverPaul->getVehicles() as $vehicle) {
176     echo "\n-{$vehicle->getRegistrationNumber()}";
177 }
178
179 //suppression des véhicules A et C pour Paul
180 $driverPaul->removeVehicle($vA);
181 $driverPaul->removeVehicle($vC);
182
```

```

183 //liste des véhicules conduits par Paul
184 echo "\n *** Liste des véhicules restant à {$driverPaul->getName()} ***";
185 foreach ($driverPaul->getVehicles() as $vehicle ){
186     echo "\n-{$vehicle->getRegistrationNumber()}";
187 }
188
189 echo "\n==== FIN DU TEST EXERCICE 4 ====";

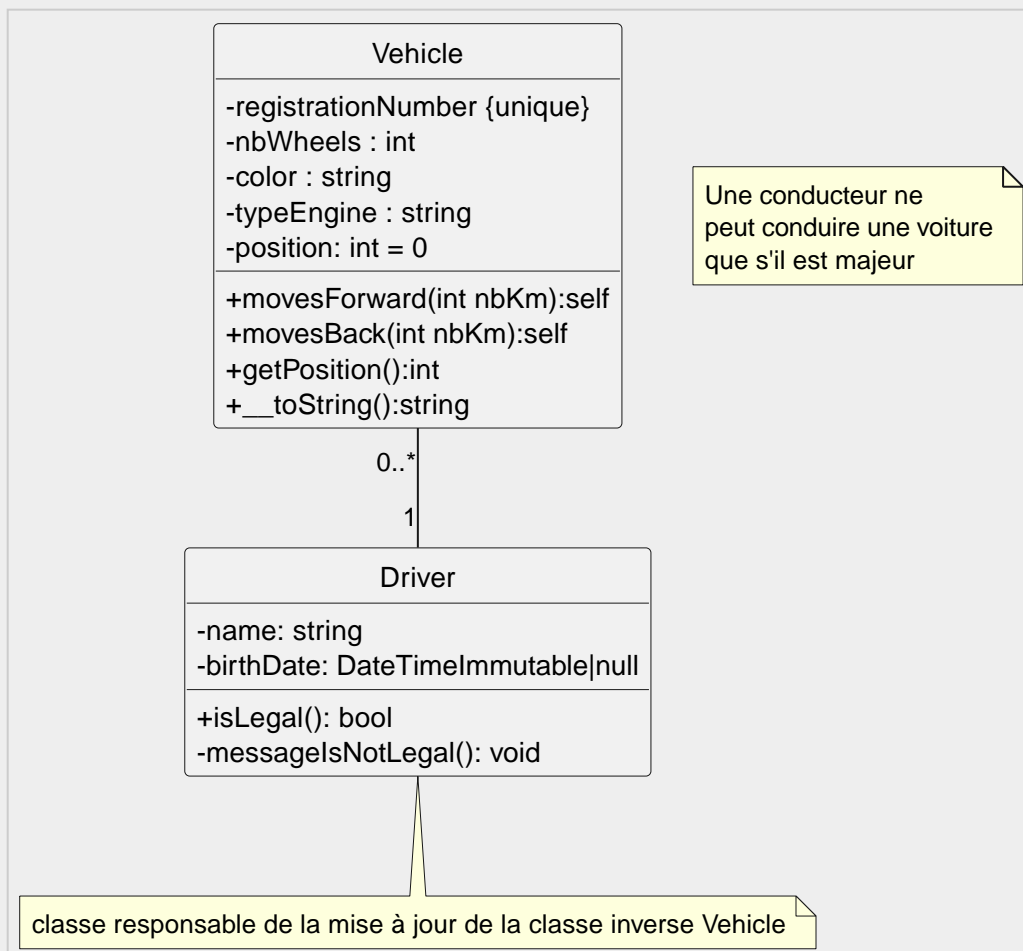
```

- ① à l'instanciation d'un **Driver**, il est attendu un véhicule afin de l'associer au conducteur dès sa création
- ② on ajoute le premier véhicule au conducteur
- ③ lorsque l'on retire un véhicule à un conducteur, il faut s'assurer qu'il lui en reste forcément un avant de procéder à sa suppression

1.5. Implémenter une association bidirectionnelle one-to-many

Q5) Travail à faire

- Implémentez le diagramme suivant :



- Créez le conducteur Paul dont la date de naissance le conduit à être majeur.

- Créez le conducteur Juliette dont la date de naissance la conduit à être mineure.
- Créez un véhicule A et tenter de lui affecter Juliette.
- Créez un véhicule B et tenter de lui affecter Paul.
- Afficher la liste des véhicules associés à Paul.
- Afficher la liste des véhicules associés à Juliette.

Correction de Q5

- Voici les modifications à apporter aux classes **Vehicle** et **Driver** utilisée précédemment :

```

1 ①
2 class Vehicle implements Stringable
3 {
4
5
6     //ici déclaration des autres attributs d'objets
7     /**
8      * @param Driver $driver conducteur associé au véhicule
9      */
10    public function __construct(
11        private Driver $driver ②
12    )
13    {
14        // ATTENTION ATTENTION ATTENTION ATTENTION
15        // ATTENTION ATTENTION ATTENTION ATTENTION
16
17        //Même si vous utilisez la promotion des arguments, il faut passer
18        //par la méthode setDriver de façon à mettre à jour l'objet inverse (ici le
19        //conducteur)
20        $this->setDriver($driver); ⑤
21    }
22
23    /**
24     * @return Driver
25     */
26    public function getDriver(): Driver
27    {
28        return $this->driver;
29    }
30
31    /**
32     * @param Driver $driver
33     *
34     * @return Vehicle
35     */
36    public function setDriver(Driver $driver): Vehicle

```

```

36     {
37         $this->driver = $driver;
38         //mise à jour de l'objet inverse (le conducteur)
39         $driver->addVehicle($this); ③
40
41         return $this;
42     }
43
44     // ici les autres opérations (mutateurs, setters, ...)
45
46 }
47
48 class Driver
49 {
50     /**
51      * @param string          $name      nom du conducteur
52      * @param DateTimeImmutable $birthDate date de naissance du conducteur
53      * @param array|Vehicle[] $vehicles  collection d'objets de type
54      Vehicle
55      */
56     public function __construct(
57         private string $name,
58         private DateTimeImmutable $birthDate,
59         private array $vehicles = []
60     ) {
61
62         //autres mutateurs, accesseurs et méthodes
63
64
65     /**
66      * @param Vehicle $vehicle ajoute un item de type Vehicle à la collection
67      */
68     public function addVehicle(Vehicle $vehicle): bool
69     {
70         //tester si le conducteur est mineur
71         if (!$this->isLegal()) {
72             $this->messageIsNotLegal();
73
74             return false;
75         }
76         if (!in_array($vehicle, $this->vehicles, true)) {
77             $this->vehicles[] = $vehicle;
78             ④
79             //pas de mise à jour de l'objet lié car c'est l'objet Vehicle qui
            est responsable de la navigabilité bidirectionnelle
80             return true;
81         }
82
83         return false;
84     }

```

```
85
86  /**
87   * @param Vehicle $vehicle retire l'item de la collection
88   */
89  public function removeVehicle(Vehicle $vehicle): bool
90  {
91
92      $key = array_search($vehicle, $this->vehicles, true);
93
94      if ($key !== false) {
95          unset($this->vehicles[$key]);
96
97          return true;
98      }
99
100     return false;
101 }
102
103 /**
104  * @return Vehicle[]
105  */
106 public function getVehicles(): array
107 {
108     return $this->vehicles;
109 }
110 }
111 }
112
113
114 echo "\n\n==== DEBUT DU TEST EXERCICE 5 ==== \n";
115
116 //création de Paul (majeur)
117 $driverPaul = new Driver('Paul', new DateTimeImmutable('2000-04-12'));
118
119 //création de Juliette (mineure)
120 $driverJuliette = new Driver('Juliette', new DateTimeImmutable('2019-02-
24'));
121
122 //Tentative d'affectation d'un véhicule A à Juliette
123 $vA = (new Vehicle($driverJuliette))->setRegistrationNumber('AAAA');
124
125 //Tentative d'affectation d'un véhicule B à Paul
126 $vB = (new Vehicle($driverPaul))->setRegistrationNumber('BBBB');
127
128 //liste des véhicules conduits par Paul
129 echo "\n *** Liste des véhicules de {$driverPaul->getName()} ***";
130 foreach ($driverPaul->getVehicles() as $vehicle) {
131     echo "\n-{$vehicle->getRegistrationNumber()}";
132 }
133
134 //liste des véhicules conduits par Juliette
```



```
135 echo "\n *** Liste des véhicules de {$driverJuliette->getName()} ***";
136 foreach ($driverJuliette->getVehicles() as $vehicle) {
137     echo "\n-{$vehicle->getRegistrationNumber()}";
138 }
139
140 echo "\n==== FIN DU TEST EXERCICE 5 =====";
```

- ① La classe véhicule est obligatoirement la classe responsable de la mise à jour de la classe inverse puisqu'elle doit être associée à un Driver dès son instantiation
- ② Attribut qui va contenir l'objet lié
- ③ Mise à jour de l'objet lié (navigabilité de **Driver** vers Vehicle)
- ④ pas de mise à jour de l'objet lié car c'est l'objet Vehicle qui est responsable de la navigabilité bidirectionnelle
- ⑤ Même si vous utilisez la promotion des arguments pour initialiser l'attribut **Vehicle::driver**, il faut passer par la méthode **setDriver** de façon à mettre à jour l'objet inverse (ici le conducteur)