La Plateforme

RunTrack PHP : POO

Why did the class go to therapy? It couldn't handle its parent's static methods.

Introduction du sujet

Aujourd'hui, nous allons faire des rappels sur la **programmation orientée objet** et comment celles-ci peuvent nous aider à organiser et mieux comprendre notre code. Nous allons revoir les **notions de classes, de propriétés, de méthodes et d'instances.**
Pour ce faire, nous allons imaginer que nous allons représenter les différentes tables de notre base de données par des classes dans notre code. Nous allons donc retrouver dans notre code les classes suivantes :

- **Student**
- **Room**
- **Floor**
- **Grade**

Dans le dossier **runtrack-b3-php**, créez un dossier **jour-03**. Pensez à faire des commits réguliers et à **respecter les consignes dans le sujet**.

Job 01

Dans le dossier **jour-03**, faites un dossier **class** dans lequel vous ferez toutes les classes de votre projet. Dans ce dossier **class**, nous allons créer une première class pour représenter les données de la table **student**. Pour ce faire, créez un fichier **Student.php** dans lequel vous allez créer la **class Student**. Cette classe va tout d'abord devoir comporter les attributs suivant afin de représenter au mieux notre table en base de données :

- **id**, int
- **grade_id**, int
- **email**, string
- **fullname**, string
- **birthdate**, datetime
- **gender**, string

Dans le **constructeur** de votre classe, mettez des **paramètres optionnels** pour initialiser les champs de votre classe lorsque vous créez une nouvelle instance. Le but est de pouvoir **instancier une class** de deux façons :

```php
$student = new Student(1, 1, "email@email.com", "Terry Cristinelli", new DateTime("1990-01-18"), "male");

$student = new Student();
```

Faites les différents tests d'instances de votre classe dans un fichier **index.php** à la racine de votre projet.

N'oubliez pas de **typer correctement votre méthode**. Vous pourriez être capable de deviner leur signature en observant les déclarations des instances ci-dessus.

Job 02

Dans le dossier **class**, ajouter les différentes **classes permettant de représenter les autres entités** présentes en base de données. Faites du coup la même chose pour les tables **grade, room et floor**. Pour **chacune de ces tables, créez une class** correspondante dans le fichier du même nom.

Une fois ceci fait, nous pouvons donc instancier les classes de la manière suivante :

```php
$grade = new Grade(1, 8, "Bachelor 1", new DateTime("2023-01-09"));

$grade = new Grade();

$room = new Room(1, 1, "RDC Food and Drinks", 90);

$room = new Room();

$floor = new Floor(1, "Rez-de-chaussée", 0);

$floor = new Floor();
```

Job 03

Maintenant que vous avez fait vos différentes classes, vous allez opter pour des normes de code un peu plus propres et vous **assurer que toutes les propriétés de vos classes sont en privées**.
Pour les modifier et y accéder, vous allez donc devoir **créer des setters et des getters**.
En voici un exemple pour la classe Student pour la propriété email :

```php
public function getEmail(): ?string
{
    /**
     * Your code here
     */
}

public function setEmail(string $email): static
{
    /**
     * Your code here
     */
}
```

Faites de même pour **toutes les propriétés de toutes vos classes dans votre dossier class**.

Job 04

Maintenant que vous avez un moyen uniforme de représenter vos entités dans votre application, nous allons pouvoir interagir avec notre base de données.

Dans un fichier **functions.php** à la racine de votre projet, faites quatres fonctions qui **retournent une instance d'une des classes de l'application** :

- **findOneStudent(int $id)**
- **findOneGrade(int $id)**
- **findOneFloor(int $id)**
- **findOneRoom(int $id)**

```php
public function findOneStudent(int $id): Student
{
    /**
     * Your code here
     */
}
```

Pour interagir avec la base de données, pensez à :

1. Créer une connexion à votre base de données.
2. Utiliser des requêtes préparées pour éviter les injections SQL.

Job 05

Nous allons aller un peu plus loin en faisant une méthode permettant de récupérer tous les étudiants d'une promotion. Dans la **class Grade**, faites une méthode **getStudents()**

afin de récupérer les étudiants liés à une promotion en particulier. Cette méthode va donc **retourner un tableau d'instance d'étudiants**.

Dans votre fichier **index.php**, récupérez une promotion avec la fonction **findOneGrade(int $id)** avec un id aléatoire et récupérez tous les étudiants de cette promotion :

```php
public function getStudent(): ?array
{
    /**
     * Your code here
     */
}
```

```php
$grade = findOneGrade(2);

$students = $grade->getStudents();
```

Job 06

Dans le même esprit que le job précédent, récupérez les différentes lignes avec des liaisons à d'autres tables. Créez donc les fonctions suivantes :

- **getGrades()** dans la **class Room**
- **getRooms()** dans la **class Floor**

Compétences visées

- **PHP :** 10 points
- **Programmation orientée objet :** 10 points

Rendu

Le projet est à rendre sur **https://github.com/prenom-nom/runtrack-b2-php**.

C'est le deuxième jour de RunTrack, donc normalement, vous n'avez que le dossier **jour-03** à push et les dossiers **jour-01** et **jour-02** sont déjà présents. Faites attention à bien **respecter la signature de chacune de vos fonctions** et les **noms de fichiers et de dossiers à la lettre !**