# Approche POO

## Classes et objets, méthodes : analyse

**ANALYSE1** : lors du premier PPE **CarteENC**, vous avez créé un controller CarteEncController et modifié son code : cf le fichier joint ANALYSE2\_CarteEncController.php

Question 1.1 :

Listez les méthodes de cette classe.

De quelle classe dérive t’elle ?

Question 1.2 :

Quelles méthodes prennent des paramètres ? Quelles méthodes n’ont pas de paramètre ?

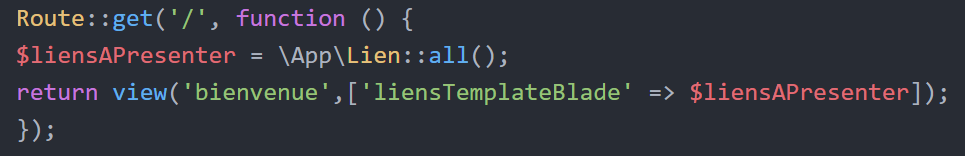
**ANALYSE2** : lors du dernier PPE **liens**, dans le fichier de routage web.php , vous avez défini 2 instructions utilisant la classe Lien.

Question 2.1 : quelle instruction a permis de créer cette classe Lien ?

Question 2.2 : de qui hérite cette classe Lien ?

Question 2.3 :

Ci-joint le code suivant :



Quelle méthode de la classe Lien est appelée ? Ou la trouve t’on ?

Question 2.4 :

Ci-joint le code suivant : (dans la route POST /soumettre)



Quelle méthode de la classe Lien est appelée ? Ou la trouve t’on ?

Question 2.5 :

Quelle est la différence dans la façon dont sont appelées les 2 méthodes (all() et save() ) ? A qui correspondent ces 2 syntaxes ?

Sources :

Le source se trouve dans le sous répertoire à partir de la racine suivante : vendor\laravel\framework\src

Par exemple quand on écrit :



Il s’agit de la classe Model située dans : C:\laragon\www\liens\vendor\laravel\framework\src\Illuminate\Database\Eloquent

Copiée dans le répertoire du jour et renommée en ANALYSE1\_Model.php

## Recherches de définitions

**ANALYSE3** :

1. Indiquez l’utilisation et le rôle de la variable spéciale $this
2. Comment sont définies et appelées les constantes dans les classes PHP ?

## Classes et objets, méthodes : exercices

**EXO1 :**

Utilisez le script **EXO1\_classeCompte.php**. Ce code contient la définition d’une classe. A la suite du code existant de définition de la classe, on vous demande de :

1. Créez un objet de cette classe Compte nommé compteProfesseur
2. Initialisez son montant à la valeur de 100
3. Créez un nouvel objet instance de la classe Compte nommé compteEleve
4. Initialisez son montant à la valeur de 100
5. Effectuez un virement de 50 du compteEleve vers le compteProfesseur
6. Affichez les montants finaux de chacun des comptes

## Méthodes, constructeur, chargement des classes

**EXO2 :**

On dispose d’une classe Rectangle, définie dans le script **EXO2\_Rectangle.php**.

On souhaite définir puis afficher quelques propriétés des objets Rectangle que l’on instancie, dans un script nomme **EXO2\_utiliseRectangle.php**

1. Il vous est demandé d’ajouter des fonctionnalités dans le fichier **EXO2\_utiliseRectangle.php**
2. Création d’un rectangle
3. Initialisation du rectangle avec la longueur $longueur et la largeur $largeur définies dans le programme
4. Récupération et affichage de *l’aire* du rectangle
5. Récupération et affichage du *périmètre* du rectangle
6. Ajout d’une méthode pour tester si le rectangle est un carré
7. Ajout d’un constructeur
8. Ajouter un constructeur à la classe Rectangle (script **EXO2\_Rectangle.php)**, enregistrez la sous un nouveau nom
9. Modifier le programme **EXO2\_utiliseRectangle.php** en l’enregistrant sous un nouveau nom pour
   1. qu’il utilise votre classe Rectangle modifiée
   2. il initialise le rectangle avec les bons paramètres dès la création
   3. modifiez maintenant le rectangle, une fois créé, pour qu’il devienne un carré
10. Chargement et utilisation de classes existantes
11. A quoi sert l’instruction ***require*** dans le script **EXO2\_utiliseRectangle.php ?**
12. Remplacer l’instruction require par la fonctionnalité d’auto chargement de classes

## Héritage

**EXO3 :**

1. Exécutez le code du programme EXO3\_Heritage.php

Ajoutez un nouvel objet de type Personne, et affichez le.

1. Créez une nouvelle classe, Employe, qui hérite de Personne, avec 1 nouvel attribut salaire
2. Redéfinissez la méthode afficheTout pour … qu’elle affiche tout
3. Instanciez cette nouvelle classe et affichez tout