# Approche POO

## Composition

Exercice n°1 : Introduction

Vous disposez d’un programme nommé EXO1\_Composition.php, comportant 2 classes :

* 1 classe Personne
* 1 classe Adresse

1. Une personne donnée est définie par son nom, mais aussi par son adresse

* Ajoutez un attribut adresse dans la classe Personne
* Créez un constructeur dans la classe Personne, qui prend en paramètre un nom, dans lequel vous initialiserez le **nom** et également cet attribut **adresse** comme une instance d’objet de type Adresse

1. Test de la solution
   1. Créez une nouvelle Personne
   2. Spécifiez une ville (attribut de l’adresse) pour cette personne (« chainage »)
   3. Affichez le nom et la ville de cette personne pour vérifier l’implémentation de votre code
2. Compréhension

Quel type d’utilisation avez-vous implémenté ? Pourquoi n’utilise-t-on pas de l’héritage ici ? Pourquoi ne met-on pas directement les attributs (ville et pays) de la classe Adresse dans la classe Personne ?

1. Amélioration

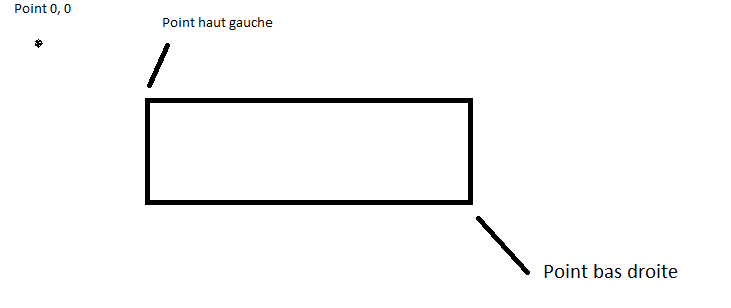
Améliorez votre programme en ajoutant un constructeur à la classe Adresse.

Exercice n°2 : Point, Rectangle, Cercle et dessin

Nous allons faire un peu de dessin.

Nous avons, dans un fichier nommé **EXO2\_DEB\_DefinitionFigures.php**:

* une classe **Point**, constituée d’une abscisse (X) et d’une ordonnée (Y) ainsi qu’une
* classe **Rectangle**, composée d’un point en haut à gauche et en bas à droite (c’est la faute à la fonction de dessin)



* classe **Cercle**, composée d’un point Centre et d’un rayon.

L’objectif est de finaliser nos classes Point, Rectangle, Cercle et de dessin.

1. Réalisez un diagramme de classe avec les classes Point, Rectangle et Cercle
2. Dans notre classe **Point**, présente dans le fichier « **EXO2\_DEB\_DefinitionFigures.php**», nous voulons ajouter une *méthode permettant d’avoir des informations sur l’objet courant, dans un contexte où on affiche l’objet (ici une instance de la classe Point) comme une chaîne de caractère*.

* A partir de votre fiche sur les méthodes de base Objet, implémentez la méthode correspondante.
* Pour tester cette méthode, complétez le programme « EXO2\_FIN\_UtiliseClasses.php »

1. Dans la classe **Rectangle**, présente dans le fichier « **EXO2\_DEB\_DefinitionFigures.php**», ajoutez également une méthode permettant d’afficher *des informations sur l’objet courant, dans un contexte où on affiche l’objet (ici une instance de la classe Rectangle)*

* Utilisez les 2 attributs Point de la classe
* Pour tester cette méthode, complétez le programme « EXO2\_FIN\_UtiliseClasses.php »
* Quel est le résultat ? D’où provient l’affichage ?

1. Dans la classe **Cercle**, ajoutez 2 méthodes pour calculer la surface et le périmètre, sachant que M\_PI est la constante PHP pour le nombre Pi.

* Pour tester cette méthode, complétez le programme « EXO2\_FIN\_UtiliseClasses.php »

1. Dessin de rectangles
2. Exécutez le programme : EXEMPLE1\_DessineRectangle.php
3. Inspirez-vous de ce programme pour compléter le programme EXO2\_FIN\_DessineRectangle
4. Dans le programme EXO2\_FIN\_DessineRectangle, que signifient-les … ?
5. Dessin de cercle

Inspirez-vous de la partie précédente et de la documentation ci-joint : <http://php.net/manual/fr/function.imagearc.php> pour compléter le programme ci joint de dessin de 3 cercles EXO2\_FIN\_DessineCercle

## Interfaces et Traits

1. Création et implémentation d’une interface

Cette partie va consister en la création d’un fichier PHP dans lequel vous allez définir une interface, une classe et utilisez ensuite cette classe et cette interface.

* 1. Créez un fichier PHP nommé EXO3\_Interface.php
  2. Créez une interface nommée NomInterface, avec 2 méthodes getNom et setNom($unNom)
  3. Créez une classe Livre qui implémente l’interface NomInterface, et qui a pour attribut privé $nomLivre
  4. Ajoutez une méthode à cette classe Livre pour afficher le nom du livre.
  5. Créez ensuite un objet $monLivrePrefere de type Livre
  6. Affectez-lui un nom.
  7. Appelez sa méthode définie en d

1. Création et utilisation d’un trait
   1. Récupérez le programme EXO3\_Traits.php
   2. Ajoutez le code d’instanciation d’un objet de la classe Livre, avec le choix d’un nom et l’affichage : identique à la partie 1
   3. Quelles différences entre Interface et Trait ?
2. Meilleure pratique
   1. Créez un nouveau fichier EXO3\_BestPratique.php
   2. Déclarez une interface, comme dans 1.b
   3. Reprenez le trait de 2.a
   4. Créez une classe implémentant l’interface et utilisant le trait
   5. Instanciez un objet, comme dans 2.b

## Le coin des champions : traits, gestion erreurs, méthode magique …

Exécutez le programme PHP EXEMPLE2\_TraitsChampion.php

1. Que fait le programme ?
2. Que fait le **trait** affiche moi ?
3. Qu’est ce que la méthode \_\_get() ?
4. Que fait la fonction trigger\_error() ?
5. Qu’est ce que la constante \_\_CLASS\_\_ ? Pourquoi fonctionne elle dans le **trait** ?
6. Quels sont les avantages des **traits** en PHP ?