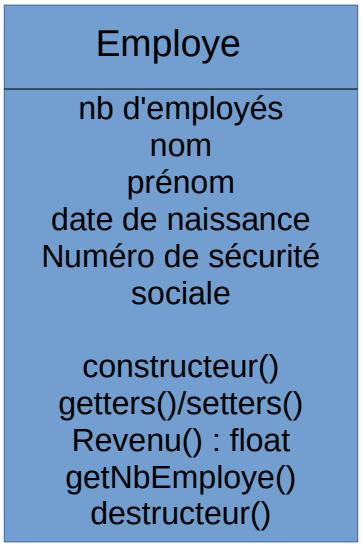
Programmation orientée objet avec PHP

Exercice 1 :

- Créer une classe abstraite Employé dans un fichier Employe.class.php. Les propriétés ne sont pas accessibles que depuis la classe et ses sous-classes.



(nb d'employés doit être statique)

(revenu() méthode abstraite)
(getNbEmploye() méthode statique)

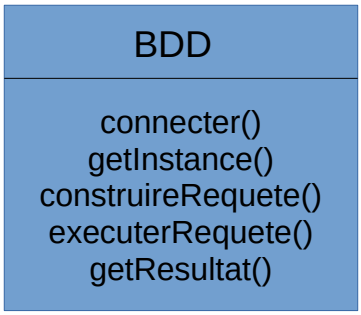
- Créer une sous-classe de Employé appelée EmployeSalarié qui possède :
  1. un attribut « salaireHebdo »
  2. un nouveau constructeur qui appelle le constructeur de la classe mère
  3. un getter et un setter pour salaireHebdo
  4. la définition de la méthode abstraite revenu()
- Créer une sous-classe de Employé appelée EmployeParHeure qui possède :
  1. un attribut «salaireH» (salaire horaire)
  2. un attribut «heures»
  3. un nouveau constructeur qui appelle le constructeur de la classe mère
  4. les getters et les setters pour les deux nouveaux attributs
  5. la définition de la méthode abstraite revenu() :
     si (heures<=40) retourne salaireH*heures
     sinon retourne 40*salaireH+(heures-40)*1,5*salaireH
- Créer une sous-classe de Employé appelée EmployeParCommission qui possède :
  1. un attribut «total vente»
  2. un attribut «taux de commission»
  3. un nouveau constructeur qui appelle le constructeur de la classe mère
  4. les getters et les setters pour les deux nouveaux attributs
  5. la définition de la méthode abstraite revenu() : totalVente*tauxCommission
- Créer une sous-classe de EmployéParCommission appelée BasePlusEmployeParCommission qui possède :
  1. un attribut « salaire de base »
  2. un nouveau constructeur qui appelle le constructeur de la classe mère
  3. un getter et un setter pour Salaire de base
  4. la redéfinition de la méthode revenu() : base+totalVente*tauxCommission

- Écrire un script en PHP pour tester toutes les classes implémentées. La sortie du script doit ressembler à l'image suivante :

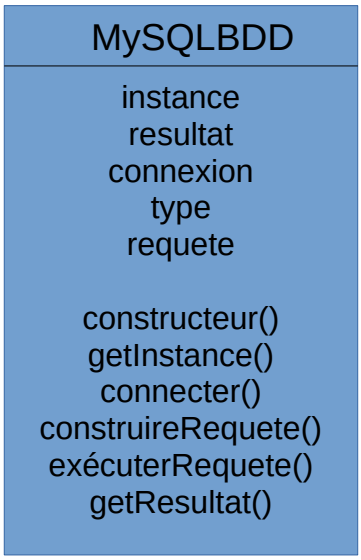
Revenu du salarié : 500€
Revenu de l'employé par heure : 950€
Revenu de l'employé par commision : 600€
Revenu de l'employé base + commision : 900€
Nombre d'employés créés : 4

Exercice 2 :

- Écrire l'interface « BDD » suivante :



- Écrire une classe appelée « MySQLBDD » qui implémente l'interface « BDD » :



(instance statique)

(getInstance() méthode statique)

1. getInstance() retourne l'unique instance de MySQLBDD si elle est déjà créée, sinon elle créée une nouvelle instance et l'affecte à l'attribut statique $instance.
2. connecter($host,$user,$pass,$db) pour se connecter à la bdd MySQL avec l'API mysqli. La connesion est stockée dans $connexion.
3. construireRequête($type,$columns,$tables, $values,$where)
   - $type = 'select', 'insert', 'update' ou 'delete', stocké dans $type
   - $columns =nom, prenom,... (utilisé par le select (select $columns), l'insert et l'update)
   - $tables= user, command, …
   - $values= (colonne1='valeur1', colonne2='valeur2', …) (utilisé par le update et le insert, update $tables set $set)
   - $where = colonne1='valeur1' and colonne2='valeur2' and …
     Exemple : if($type=='select')

$requete= $type.' '.$columns.' from '.$tables.' where '.$where ;

4. exécuterRequête() : exécute la requête stockée dans $requête et stocke le résultat dans $résultat.
5. récupérerRésultat() : retourne un tableau associatif si la requête était un select et a retourné des données.

- Utiliser la classe MySQLDBB pour exécuter les requêtes suivantes sur la table Products créée dans le TP précédent :

Requête : select * from Products where category='DVD'
Array ([0] => Array ([id] => 2 [name] => Matrix [category] => DVD [price] => 10) [1] => Array ([id] => 7 [name] => Avatar [category] => DVD [price] => 25))
Requête : update Products set price=13 where id=2
Requête : insert into Products (name, category, price) values ('Doom', 'DVD', 20)
Requête : select * from Products where category='DVD'
Array ([0] => Array ([id] => 2 [name] => Matrix [category] => DVD [price] => 13) [1] => Array ([id] => 7 [name] => Avatar [category] => DVD [price] => 25)
[2] => Array ([id] => 14 [name] => Doom [category] => DVD [price] => 20))