TP POO

> ⓘ **Encapsulation**
>
> Toutes les classes implémentées dans ces exercices doivent respecter la règle de l'**encapsulation**.

Exercice 1

> Créer une classe "Entreprise" possédant un nom, une adresse et un numéro de SIRET composé de 14 chiffres.

> Lorsqu'on print un objet issu de cette classe le résultat doit être le suivant: "L'entreprise {nom}, ayant son siège social au {adresse}, possède le numéro de SIRET {siret}".

> Créer une instance de cette classe et printer l'objet obtenu.

> Printer le nom de l'entreprise

> Changer le numéro SIRET de l'entreprise et printer l'objet résultant

Exercice 2

> A l'aide d'une `dataclass`, créer une classe "DatabaseConnection" possédant un type de base (ex: MySQL, MariaDB, PostgreSQL...), un utilisateur, un mot de passe, et un hôte. Toutes ces propriétés sont des `string`. L'hôte a une valeur par défaut "localhost", si rien n'est précisé.

> La classe possède une propriété statique `nb_instance` qui va s'incrémenter à chaque création d'un objet issu de cette classe.

> La classe possède une méthode statique retournant le nombre total d'instance sous la forme : "La classe DatabaseConnection possède actuellement {x} instance(s).".

> La classe possède une méthode de classe permettant de créer une instance avec les informations suivante : type -> "mariadb", hôte -> "76.287.872.12", utilisateur -> "root", mot de passe -> "1234".

> Initialiser un objet de cette classe sans spécifier l'hôte et printer le résultat obtenu.

> Initialiser un objet de cette classe en appelant votre factory, et printer le résultat obtenu.

Exercice 3

> Créer une classe "Client" possédant un nom, un prénom, une adresse, et un numéro de sécurité sociale (NIR) composé de 15 chiffres. Un contrôle doit être réalisé sur le NIR au moment de la création d'un nouvel objet.

> Créer une classe "CompteBancaire" dont le constructeur accepte les 3 paramètres suivants : - une date de création de type `string` au format "YYYY-MM-DD" - un client de type `Client` - un solde de type `float`

> La classe a 4 propriétés : - Une date de création au format `date` - Un client de type `Client` - Un identifiant interne composé de 4 lettres majuscules aléatoires suivies de la date de création du compte au format `DDMMYYYY` (exemple: IYSQ26052020) - Un solde de type `float`

> La classe "CompteBancaire" possède également une propriété statique renvoyant la somme des soldes de tous les clients de la banque.

> Deux comptes bancaires sont considérés comme égaux lorsque leur soldes sont égales.

> Créer 2 objets comptes bancaires, printer leur identifiant interne respectif, et printer leur égalité l'un avec l'autre.

> Printer le solde total de tous les comptes bancaires créés.

Exercice 4

> Créer une classe `Movie`, possédant un titre de type `string`, une date de sortie au format `string`, et un résumé au format `string`.

> À la première instance de la classe `Movie`, un nouveau fichier `.json` est créé dans un dossier data de votre projet si il n'existe pas, et est rempli de la manière suivante.

```
{
  "movies": [
    {
      "titre": "titre de votre premier film",
      "date_de_sortie": "12/12/2022",
      "description": "Votre description"
    }
  ]
}
```

> À chaque nouvelle instance de la classe Movie, un nouvel élément est inscrit dans la liste de "movies" présent dans votre `.json` de sauvegarde.

> La classe `Movie` possède une méthode pour supprimer un film de la liste de movies présente dans votre `.json`.

> La classe `Movie` prévoie le changement du titre, de la date de sortie ou de la description, des films présent dans le `json` de sauvegarde.

> Enfin, la classe permet de printer toutes les informations d'un film présent en sauvegarde dans le cadre d'une recherche par le titre du film.

> Construire enfin une application de terminal dans lequel l'utilisateur peut choisir 4 commandes : create, read, update, delete. - Create -> il doit renseigner un titre, une date de sortie au format DD/MM/YYYY, et une description et le script inscrit alors le film en sauvegarde. Le script renvoi alors la liste complète des films. - Read -> il peut choisir de lire les informations d'un

> film en particulier en le recherchant par son titre, ou alors d'afficher tous les films par ordre croissant de date de sortie. - Update -> Il choisis un film par son titre et choisis la propriété qu'il veut modifier. À la fin de la démarche, le script lui print le film après modification. - Delete -> Suppression d'un film par son titre. Le script renvoi alors la liste de tous les films.

> Les recherches par titre de film fonctionne avec ou sans majuscules

> Les titres de films sont stockés avec une majuscule à chaque mot.

Correction