[![Review Assignment Due Date](https://classroom.github.com/assets/deadline-readme-button-22041afd0340ce965d47ae6ef1cefeee28c7c493a6346c4f15d667ab976d596c.svg)](https://classroom.github.com/a/0SBo15kW) = Exercice : Manipulation de collections

IMPORTANT

Dans cet exercice, vous respecterez les contraintes suivantes:

- Vous utiliserez `git` en effectuant des commits réguliers comportant des messages informatifs.
- Le *build* sera assuré par Maven et plus précisément Maven wrapper (déjà intégré dans le projet). Aucune manipulation en dehors de Maven ne devra être nécessaire.
- La version de Java à utiliser est la version 17.
- Le *build* devra intégrer `checkstyle` pour la vérification des règles de codages Google. Le projet devra donc les respecter. `checkstyle` devra être exécuté automatiquement durant la phase `validate` du cycle de vie par défaut.
- Des tests unitaires JUnit 4 (version *4.13.2*) devront être disponibles pour la plupart des méthodes développées.
- Les fonctionnalités du langage Java devront être utilisées au mieux (exceptions, librairie de collections, I/O, ...).

L'objet de cet exercice est de simuler l'interrogation d'un DNS (*Domain Name Server*). Un DNS convertit une adresse IP (`192.168.0.1` par exemple) en un nom qualifié de machine (`machine.domaine.local` par exemple) et inversement. Un nom qualifié comporte le nom de la machine (avant le premier `'.'`) et un nom de domaine (après le premier `'.'`).

L'interface proposera une ligne de commande à partir de laquelle les commandes suivantes devront être interprétées:

- `nom.qualifié.machine` : l'adresse IP de la machine est affichée;

  par exemple

  ```
  > www.uvsq.fr
  193.51.31.90
  ```

- `adr.es.se.ip` : le nom qualifié de cette machine est affiché;

  par exemple

  ```
  > 193.51.31.90
  www.uvsq.fr
  ```

- `ls [-a]` *domaine* : la liste des machines du domaine *domaine* sera affichée triée selon le nom des machines ou selon les adresses IP (si `-a` est présent).

par exemple

```
> ls uvsq.fr
193.51.25.12 ecampus.uvsq.fr
193.51.31.154 poste.uvsq.fr
193.51.31.90 www.uvsq.fr
```

- `add adr.es.se.ip nom.qualifié.machine` : ajoute une adresse IP associée à un nom de machine. Si l'adresse ou le nom de machine existe déjà, un message d'erreur est affiché.

  par exemple

```
> add 193.51.25.24 pikachu.uvsq.fr
> ls -a uvsq.fr
193.51.25.12 ecampus.uvsq.fr
193.51.25.24 pikachu.uvsq.fr
193.51.31.90 www.uvsq.fr
193.51.31.154 poste.uvsq.fr
> add 193.51.25.90 www.uvsq.fr
ERREUR : Le nom de machine existe déjà !
```

IMPORTANT

- La base de données du serveur sera conservée dans un fichier texte (une ligne par machine au format "un_nom_de_machine une.adresse.IP") chargé au lancement du programme.
- En cas d'ajout d'une machine par la commande `add`, le fichier texte doit être mis à jour
- Pour lire et écrire le fichier texte, vous pouvez utiliser java.nio.file.Files.readAllLines et java.nio.file.Files.write.
- Le nom du fichier devra être stocké dans un fichier de propriétés (cf. Tutoriel sur les propriétés).

1. Réaliser les classes `AdresseIP`, `NomMachine` et `DnsItem` qui représentent respectivement une adresse IP, un nom qualifié de machine et une entrée du DNS.
2. Réaliser la classe `Dns` qui proposera les opérations suivantes:
   - un constructeur qui chargera la base de données,
   - deux méthodes `getItem` qui retourneront une instance de `DnsItem` soit à partir d'une adresse IP, soit à partir d'un nom de machine,
   - une méthode `getItems` qui retournera une liste d'items à partir d'un nom de domaine,
   - une méthode `addItem` qui prend une adresse et un nom de machine pour les ajouter à la base de données. Une exception est levée en cas d'erreur.
3. Réaliser la classe `DnsTUI` qui se chargera des interactions avec l'utilisateur. Cette classe fournira une méthode `nextCommande` qui analysera le texte saisi par l'utilisateur et retournera un objet implémentant l'interface `Commande` (cf. question suivante) et une méthode `affiche` qui affichera un résultat.

4. Les commandes DNS seront implémentées à l'aide du modèle de conception Commande.
    - créer l'interface `Commande` comportant une seule méthode `execute`,
    - créer une classe implémentant cette interface pour chaque action (rechercher une IP, rechercher un nom, rechercher les machines d'un domaine, ajouter un item dans la BD, quitter l'application).
5. Réaliser la classe principale `DnsApp`. La méthode `run` de cette classe interagira avec l'interface utilisateur pour récupérer la prochaine commande, l'exécutera puis affichera la résultat.