# TD N°4 : Gestion de la persistance

Philippe Lahire

On met à votre disposition le squelette des classes Personne, Adresse et Etudiant.

## Exercice 1

### Exercice 1a

Créer une classe **Promotion2020** qui initialise un tableau d'étudiants appelé *etudiants* avec une dizaine d'étudiants (en incluant leur adresse et leurs parents). Modifier les classes **Personne**, **Etudiant**, **Adresse** afin de permettre leur sérialisation (mécanisme par défaut) sachant que le numéro de téléphone ne doit pas être sauvegardé pour respecter la confidentialité. Les champs *rootEtudiant* et *rootAdresse* seront rajoutés et référenceront respectivement le premier étudiant du tableau et son adresse.

Ajouter ensuite à la classe **Promotion2020** deux méthodes *saveEtudiants* et *retrieveEtudiants* permettant pour la première de sauvegarder le tableau d'étudiants dans un fichier dont le nom est passé en paramètre, et pour la seconde de récupérer ce tableau à partir d'un fichier dont le nom est lui aussi passé en paramètre. Le nom du fichier est « exercice1Etudiants »

Vous exécuterez le programme en utilisant la première fois la méthode *saveEtudiants* et la seconde fois en utilisant *retrieveEtudiants*. Vous vérifierez le contenu des trois champs (*étudiants*, *rootEtudiant* et *rootAdresse*) avant la désérialisation et après la désérialisation en effectuant l'affichage adéquat.

Il faudra vérifier aussi que les informations relatives au 1$^{er}$ étudiant sont bien toujours partagées et non dupliquées après la désérialisation.

### Exercice 1b

Ajouter un champ *prenom2* à la classe Personne et l'initialiser dans le constructeur par défaut de la classe à une valeur quelconque. Réexécuter alors la désérialisation du fichier « exercice1Etudiants ».

Qu'en déduisez-vous ?

Renommer ensuite le champ *prenom* en *prenom1*. Réexécuter alors la désérialisation du fichier « exercice1Etudiants ».

Qu'en déduisez-vous ?

## Exercice 2

### Exercice 2a

Faire les changements nécessaires dans la classe **Personne** pour que l'adresse ne soit pas sérialisée lorsqu'un objet de type **Personne** (qui peut être soit une instance de la classe **Personne**, soit une instance de la classe **Etudiant**) est sauvegardé. Vous pourrez proposer deux solutions.

### Exercice 2b

On demande que pour les instances de la classe **Etudiant** l'adresse soit sérialisée mais que pour les instances de la classe Personne celle-ci ne le soit toujours pas. Proposer une solution. Que pensez vous des solutions que vous avez proposées précédemment ?

# Exercice 3

Utiliser le champ *serialPersistentFields* pour spécifier la liste des champs à sauvegarder dans la classe **Personne**, dans la classe **Etudiant** et la classe **Adresse**.

## Exercice 3a

Dans un premier temps, les champs *ville* et *numero* de la classe **Adresse**, les champs *nom* et *age* de **Personne** et le champ *numTel* de Etudiant seront marqués comme sérialisable.

Adapter la classe **Promotion2020** pour vous assurer que les champs spécifiés et eux seuls sont sérialisés et désérialisés.

## Exercice 3b

Même chose en rajoutant le champ *adresse* dans la classe Personne.

# Exercice 4

Expérimenter la sérialisation à travers l'interface *Externalizable* plutôt qu'à travers l'interface *Serializable*. Pour cela dupliquer les classes fournies en les renommant (exemple : **Personne** devient **Personne2**.

## Exercice 4a

Dans un premier temps, vous ferez en sorte que tous les champs des trois classes soient sérialisés et désérialisés. Qu'en déduisez vous de l'utilisation des constructeurs par défaut ?

## Exercice 4b

Dans un deuxième temps, faire en sorte que Personne2 ne soit pas sérialisable (n'implémente pas **Serializable** ni **Externalizable**[1]. Modifier **Etudiant2**[2] en conséquence pour avoir le même effet que pour l'exercice 4a.

[1] Vous pourrez faire ces modifications dans une classe **Personne3**.
[2] Vous pourrez aussi faire ces modifications dans une classe **Etudiant3**. Cela nécessitera de modifier légèrement la classe contenant la méthode « main ».