

TP Tables d'associations et Ensembles

Il est fortement conseillé de lire la totalité du sujet avant de se lancer dans la réalisation.

1 Introduction

Dans ce TP, vous allez concevoir et implémenter une structure de données inspirée du fonctionnement d'un réseau social. Les relations entre utilisateurs seront représentées à l'aide d'une table d'associations par hachage, proche de l'implémentation de `HashMap` en Java.

1.1 Objectifs

Ce TP a pour but de :

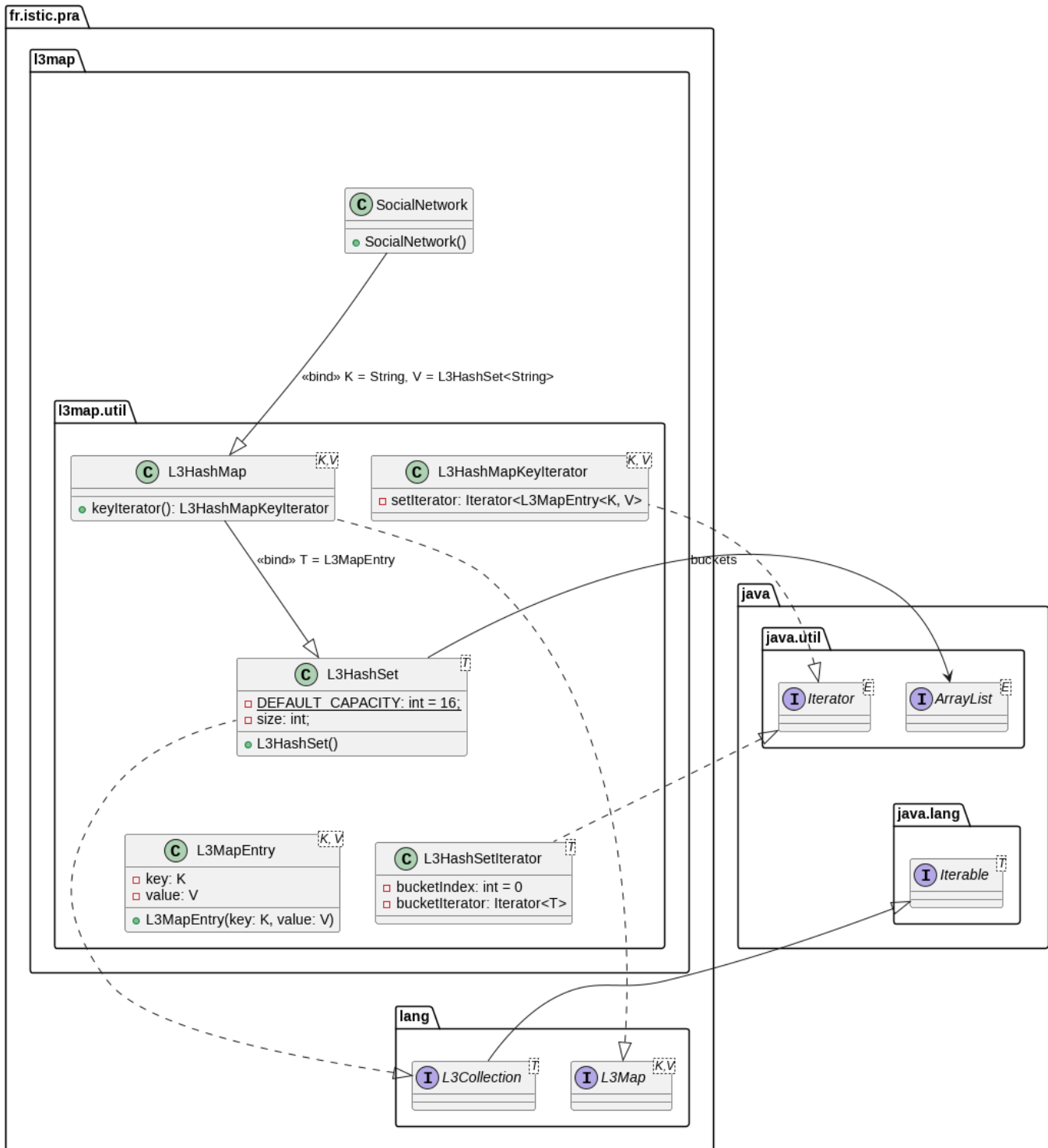
- Manipuler les concepts de table d'associations et d'ensemble par hachage, et de leurs itérateurs.
- Implémenter des structures de données génériques.
- Utiliser efficacement les collections personnalisées dans une application concrète.

1.2 Architecture logicielle

Comme pour le TP précédent, nous vous fournissons un squelette de projet divisé en deux grandes parties :

1. **La structure de données** (package `fr.istic.pra.l3map.util`), qui comprend :
 - `L3HashSet<T>` : ensemble haché, basé sur des buckets (16 buckets au maximum).
 - `L3HashSetIterator<T>` : itérateur associé.
 - `L3HashMap<K, V>` : table d'associations par hachage contenant des couples clé-valeur.
 - `L3HashMapKeyIterator<K>` : itérateur sur les clés d'une `L3HashMap`.
2. **L'application concrète** (package `fr.istic.pra.l3map`), composée uniquement de la classe :
 - `SocialNetwork` : modèle de réseau social vu en TD.

Un aperçu simplifié de ces classes est disponible ci-dessous :



Remarque : l'interface **L3Collection**, utilisée dans le TP précédent, est toujours présente. Cependant, elle hérite désormais de **Iterable**, ce qui permet une intégration avec les itérateurs Java et une utilisation avec le `foreach` de Java (`for (type variableName : collectionName) {}`).

2 Manipulation du projet

2.1 Structure du projet

Vous devez récupérer le squelette du projet maven sur le git du cours au lien <https://gitlab2.istic.univ-rennes1.fr/pa/tp-map> et l'ouvrir avec l'IDE de votre choix (plus de détails sur comment faire dans le README du projet). Le projet contient :

- Les squelettes des classes `SocialNetwork`, `L3HashSet`, et `L3HashMap`.
- Une bibliothèque compilée `pa-map-2.0.jar` contenant des versions déjà prêtes de `L3HashSet` et `L3HashMap`.

2.2 Tester le projet

- *Tests unitaires* : Vous disposez d'un jeu de tests dans la classe `TestSocialNetwork`.

3 Travail à réaliser

Tout au long de ce TP, vous pouvez, et à plusieurs occasions, devez ajouter des attributs ou des fonctions privées dans les classes que l'on vous fournit.

3.1 Implémentation de `SocialNetwork`

Les fonctions déjà développées en TD sont préremplies dans cette classe, en prenant en compte l'implémentation par hachage. Dans un premier temps, vous devez finir de compléter cette classe.

3.2 Implémentation de `L3HashSet` et `L3HashMap`

Vous devez ensuite implémenter une structure de table de hachage :

1. Commencez par `L3HashSet` et son itérateur. Pensez à changer l'import dans `SocialNetwork` pour utiliser votre `L3HashSet`. Une classe `TestL3HashSet` est disponible pour tester votre implémentation avant d'avoir tout implémenter.
2. Ensuite, implémentez `L3HashMap`. Pensez à changer l'import dans `SocialNetwork` pour utiliser votre `L3HashMap` (et gardez l'import de votre `L3HashSet`).

Pensez à repasser les tests de `TestSocialNetwork` pour être sûr que tout fonctionne !

3.3 Implémentation de plusieurs itérateurs pour `L3HashMap` (bonus)

En Java, on peut itérer non seulement sur les clés d'une `HashMap`, mais aussi sur les valeurs ou les couples clé-valeur.

Implémentez deux itérateurs supplémentaires :

- `L3HashMapValueIterator` : pour itérer sur les valeurs.
- `L3HashMapEntryIterator` : pour itérer sur les paires clé/valeur.

4 Annexe

4.1 Model UML complet

