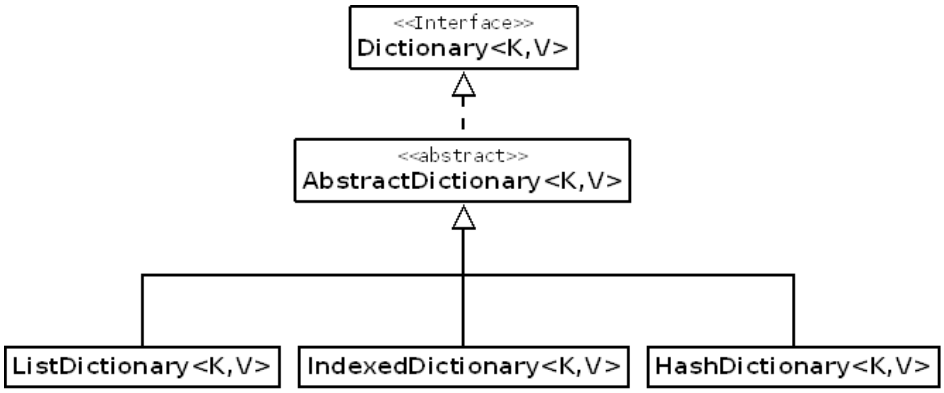
ESIAL 1A – Structures de Données
Implantation de tables

L'objectif de ce TP est de réaliser et tester des implantations du type algébrique Dictionary[K,V]. Certaines classes vous sont fournies dans le répertoire `/home/depot/1A/SD/TP-Tables`. Nous vous rappelons le diagramme de classes considéré pour l'implantation des tables. L'utilisation d'un IDE est non autorisée pour cette séance.



1. Complétez la classe `HashDictionary<K,V>` correspondant à une représentation par hashcode d'une table. Cette classe hérite de la classe abstraite `AbstractDictionary<K,V>`. Les sous-tables seront implantées à l'aide d'une `ArrayList`. Vous exécuterez votre implantation à l'aide de la classe `TestHashDictionary` (qui est une classe très sommaire). Vous noterez que la classe `HashDictionary<K,V>` utilise la fonction de hachage `hashCode` héritée de la classe `Object` pour calculer le hashcode de la clé.

2. Complétez la méthode d'accès (`value`) pour qu'elle affiche le nombre de comparaisons nécessaires pour accéder à une clé. Dans la classe de test `TestHashDictionary`, observez l'influence du nombre de sous-tables sur le nombre de comparaisons nécessaires pour accéder à la valeur d'une clé.

3. Nous avons redéfini la méthode `hashCode` dans les classes `KeyType1`, `KeyType2`, `KeyType3` (déjà compilées). Vous utiliserez ces classes pour créer les clés : par exemple, `dico.add(new KeyType1("Arbre"),"Tree")`. Dans chacun des trois cas, observez la distribution des clés dans la table de hachage et identifiez pourquoi ces fonctions de hachage ne sont pas appropriées.

4. Complétez la classe `KeyType4` en implantant la fonction de hachage vue en TD, et vérifiez la distribution obtenue dans la table. Vous consulterez la documentation (javadoc) pour déterminer les méthodes de la classe `String` dont vous aurez besoin pour calculer le hashcode.

5. **Pour ceux qui avancent vite**, réécrivez la classe `TestHashDictionary` pour qu'elle vérifie les différents axiomes de la spécification algébrique.

6. **Pour ceux qui avancent très vite**, implantez la classe `ListDictionary<K,V>`, puis la classe `IndexedDictionary<K,V>`, et vérifiez qu'elles respectent les axiomes de la spécification algébrique. Vous réutiliserez les tests de la classe `TestHashDictionary`, et introduirez une classe abstraite `TestDictionary` pour factoriser le code.