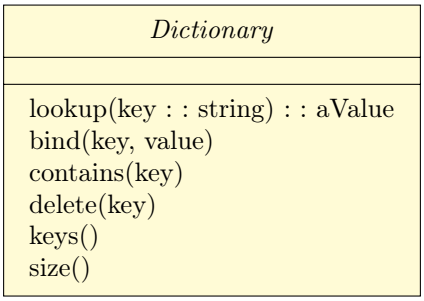
Fiche de TP no 3

Rappelez vous la structure de données abstraite `Dictionary` :



Voici une représentation ce cette structure abstraite dans le langage **python** en tant que classe abstraite (les méthodes sont décrits, mais pas implémentés) :

```python
class Dictionary():
    """Classe abstraite pour la structure de données Dictionary"""

    def lookup(self, key):
        """Cherche la valeur correspondante à key.
        Retourne None si key ne se trouve pas
        dans le dictionnaire"""

    def bind(self, key, value):
        """Ajoute la paire (key, value) au dictionnaire.
        Si une paire de la forme (key, x) se trouve déjà dans le dictionnaire,
        alors la valeur de key sera modifiée"""

    def contains(self, key):
        """Vérifie s'il y a une paire (key, value) dans le dictionnaire"""

    def delete(self, key):
        """Élimine toute paire de la forme (key, value) du dictionnaire"""

    def keys(self):
        """Retourne la liste de toutes les clés dans le dictionnaire"""

    def size(self):
        """Retourne le nombre de clés dans le dictionnaire"""
```

Exercice 1. Le but de cet exercice est apprendre à utiliser une classe (ou une structure de données) avant qu'elle soit implémentée et/ou sans connaître les détails des l'implémentation.
On écrira un script `dictionaryTest.py`. Ce script devrait, en principe, contenir un test pour chaque méthode. Nous nous limiterons aux méthodes `lookup` et `delete`.
On manipulera un objet d'une classe héritant depuis `Dictionary` et un dictionnaire natif du langage **python**— les dictionnaires sont ceux que, par exemple, on initialise avec l'instruction `ma_var = {}`. On suppose l'implémentation **python** des dictionnaires est correcte, et on testera un l'implémentation en faisant les mêmes opérations sur les deux dictionnaires un nombre répété de fois.

Par exemple, voici ci-dessous la classe abstraite `DictionaryTestAbstract` dont vous pouvez donner une implémentation. Cette classe nous servira plus tard pour tester nos implémentations de la classe `Dictionary`.

```python
class DictionaryTestAbstract():

    def __init__(self, aDictionary):
        self.aDictionary = aDictionary
        self.dico = {}

    @staticmethod
    def generate_key(size=6):
        """Retourne une clé (chaine de caracteres) au hasard de longueur size"""

    def ensureEmpty(self):
        """Verifie que au debut d'un test,
        self.dico et self.aDictionary sont vides"""

    def inhabit(self, noKeys=100):
        """Ajoute à self.aDictionary et à self.fico noKeys clés."""

    def compare(self):
        """Fait la comparaison entre self.dico et self.aDictionary.
        Affiche le numero de résultat positifs et négatifs"""

    def lookupTest(self, noKeys=100):
        """Test de la methode lookup :
            On ajoute noKeys keys à self.dico et self.aDictionary
            On affiche la comparaison entre self.dico est self.aDictionary"""

    def deleteTest(self, noKeys=100):
        """Test de la methode delete :
            On ajoute noKeys keys à self.dico et self.aDictionary
            On retire de self.dico et self.aDictionary des clés au hasard.
            On affiche la comparaison entre self.dico est self.aDictionary"""
```

N'oubliez pas de jeter un coup d'oeil sur la documentation du module `random` qui vous sera utile :

https://docs.python.org/3.7/library/random.html

Exercice 2. Donnez une implémentation de `Dictionary` en utilisant les listes d'association (listes de paires clé/valeur). Pour cela, on écrira un script contenant la définition d'une classe (concrète) `AssocList` qui héritera depuis la classe `Dictionary`.

Exercice 3. On utilisera le script `dictionaryTest.py` pour tester bon fonctionnement la classe `AssocList`.

Exercice 4. Donnez une implémentation de cette structure basé en utilisant les tables hachage (vues en cours). Écrivez ainsi une classe `HashTable` qui héritera depuis la classe `Dictionary`.

Exercice 5. De même, utilisez `dictionaryTest.py` pour tester la classe `HashTable`.

Exercice 6. Si pas encore fait, organisez votre script `dictionaryTest.py` en une classe. Écrivez une extension de cette classe contenant des méthodes pour tester la performance avec des différentes implémentations des dictionnaires (celle native de **python** incluse).

Pour les travailleurs

Exercice 7. Donnez une implémentation de `Dictionary` en utilisant les arbres de recherche binaires. Testez comme auparavant.