# Définitions de cas de tests

Au cours de cette séance inspirée d'un sujet du professeur Burkhart Wolff, nous allons réfléchir aux différents tests à appliquer sur une classe `Conteneur` pour en respecter les spécifications.

L'objectif est de tester l'implémentation d'un conteneur d'associations *clé–valeur*. On ne se soucie pas ici de la façon dont le conteneur est implémenté, simplement de la façon dont on va le tester au vu de ses spécifications.

À la création d'un conteneur, on indique la taille du conteneur, qui doit être au moins 2, le construction lève une exception `IllegalArgumentException` si le paramètre n'est pas suffisant.

On peut ajouter une association au conteneur avec la méthode `ajouter` qui prend en paramètre une clé sous la forme d'une simple chaîne de caractères (`String`), et une valeur, une instance de `Double`. Une même valeur peut être associé à plusieurs clés différentes, mais une clé n'est associée qu'à une unique valeur. Ainsi, lorsqu'on ajoute une association clé–valeur correspondant à une clé déjà intégrée, il faudra substituer la nouvelle valeur à la valeur existante. Si on ajoute une nouvelle association clé–valeur dans un conteneur plein, cela lève une exception `ExceptionConteneurPlein`.

On peut chercher la valeur associée à une clé dans le conteneur, et si la clé est manquante, cela levera une exception `ExceptionCleManquante`. On peut aussi consulter le nombre d'entrées utilisées dans le conteneur avec la méthode `getNbEntrees`. Ce nombre d'entrée doit correspondre avec le nombre de clés avec une association.

On peut retirer des associations clé–valeur à partir de leur clé ou de la valeur. Si on retire une association absente du conteneur, rien ne se passe. Si on retire une valeur présente dans le conteneur et associée à plusieurs clés, cela retire toutes les associations correspondantes.

La méthode `redimensionner` pourra être utilisée dans deux cas :

— quand le conteneur est plein, et uniquement dans ce cas, elle pourra être utilisée pour augmenter la taille du conteneur.

— quand le conteneur est vide, et uniquement dans ce cas, elle pourra être utilisée pour réduire la taille du conteneur, tout en maintenant une taille minimum de 2.

Cette méthode n'aura alors d'effet que si l'on se trouve dans l'un de ces deux cas, et elle ne modifiera pas les associations contenues dans le conteneur.

On peut enfin vider intégralement un conteneur de ses associations avec `retirerTout`, opération qui ne change pas la taille du conteneur.

Voici la liste récapitulative des méthodes que l'on souhaite trouver dans le conteneur.

```
public class Conteneur {
        Conteneur(int taille) throws IllegalArgumentException;
        int getTaille();
        boolean isEmpty();
        void ajouter(String cle, Double valeur) throws ExceptionConteneurPlein;
        Double getValeur(String cle) throws ExceptionCleManquante;
        int getNbEntrees();
        void retirer(String cle);
        void retirer(Double valeur);
        void retirerTout();
        void redimensionner(int nouvelleTaille);
}
```

Au cours de ce TD, vous allez détailler les tests à effectuer pour garantir qu'une implémentation de Conteneur vérifie bien toutes les spécifications ci-dessus. Vous devez tester aussi bien les cas de succès que les cas d'échecs.

1. Listez l'ensemble des points à tester parmi les spécifications.
2. Dans une démarche d'écrire les tests en préparation du code, définissez dans quel ordre implémenter les différents tests. On rappelle que lors de l'ajout du test, le test et censé échouer pour être corrigé immédiatement après par l'ajout du code de production correspondant.