Algorithmique avancée

Introduction aux structures de données

Frédéric Guyomarch

IUT-A
Université de Lille

2020/2021 - Semestre 3

Intervenants

- Groupe G : Alexis Gras
- Groupe H : Frédéric Guyomarch
- Groupe I : Patricia Everaere
- Groupe J : Delphine Poux

Organisation du cours

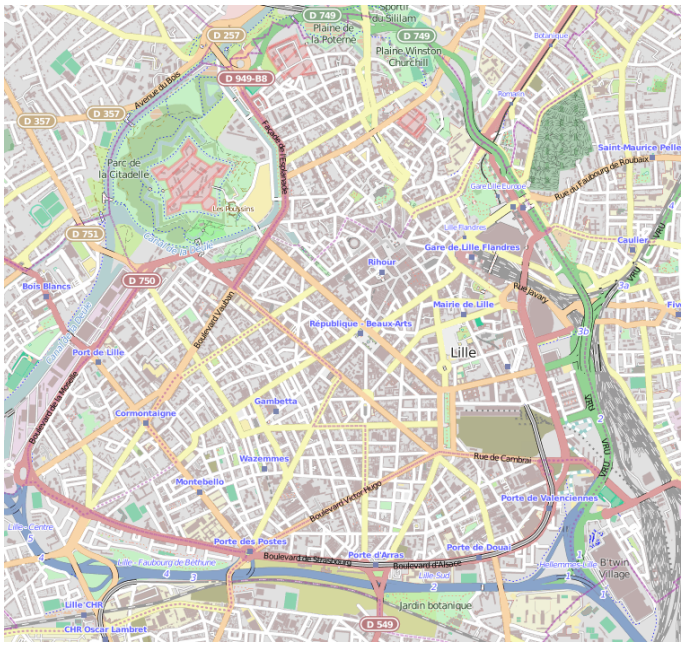
- Volume horaire
  - 6h de CM (1h/semaine)
  - 7h30 de TD (1h30/semaine)
  - 15h de TP en Java (3h/semaine)
- Evaluation
  - un examen final (4 nov, 14 :00-16 :00)
  - un TP noté (18 nov, 14 :00-16 :00)

Des structures...

de données !

Les données sont *partout*, parfois *complexes* et souvent *structurées* par nature !

Dictionnaire



Une carte

Introduction aux structures de données

Definition

Une structure de données (SDD) est un agencement, une manière d'organiser les données en mémoire.

Algorithms + Data Structures = Programs (Niklaus Wirth)

Introduction aux structures de données

Objectifs de la matière

- Découvrir les principales structures de données (SDD),
- ainsi que les algorithmes associés,
- savoir les modéliser,
- pour au final utiliser les SDD adaptées aux problèmes.

Introduction aux structures de données

Contenu de la matière

- Algorithmes de tris
- Listes chaînées
- Piles/Files
- Tables de hachage
- Arbres binaires

Introduction aux structures de données

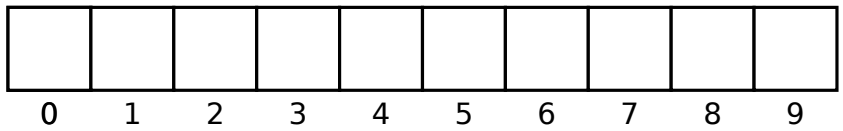
Une classification grossière des SDD

- Stockage de données réelles (listes chaînées, tables de hachage, etc)
- Outils de programmeurs pour les programmeurs (files avec priorités, piles, etc.)
- Modélisation (graphes, files,etc.)

Rappel sur les tableaux

En Java

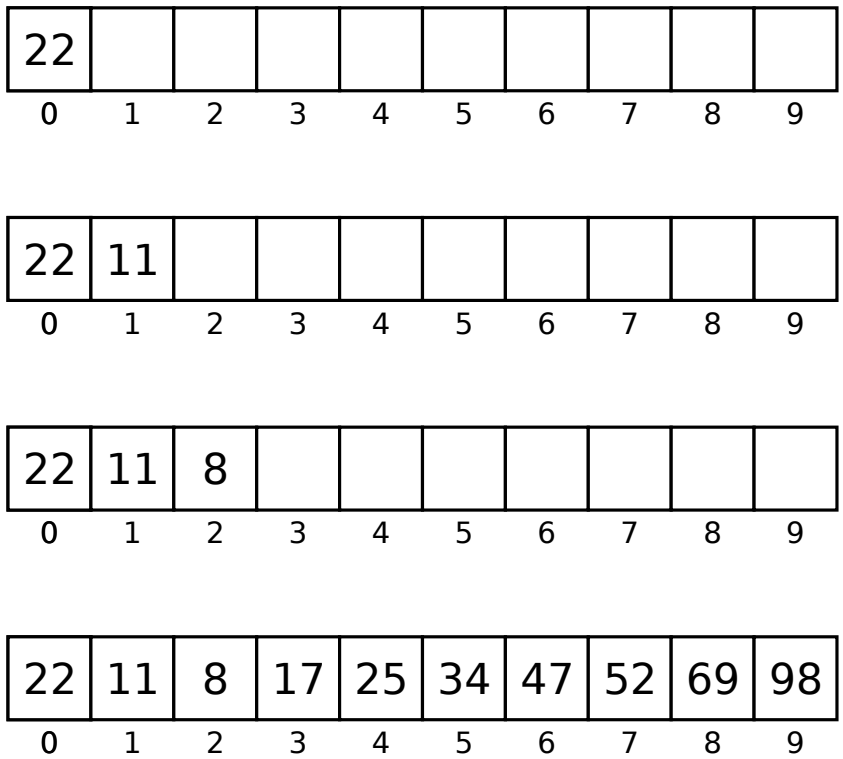
```java
/*Declaration d'un tableau d'entiers */
int [] tab;
/*Allocation de ce tableau de taille 10 */
tab = new int[10];
```



Rappel sur les tableaux

Insertion

Ajouts successifs des valeurs 22, 11, 8, 17, 25, 34, 47, 52, 69 et 98.

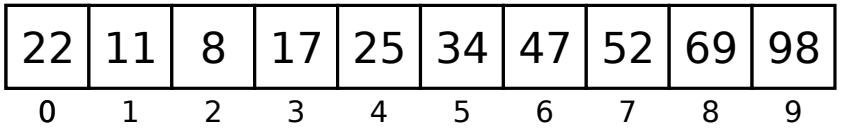


Il suffit d'incrémenter un compteur pour placer chaque valeur au bon endroit.

Rappel sur les tableaux

Recherche Séquentielle

Recherche de la valeur 47 :




```
int search(int val, int[] tab){
  for(int i=0; i < tab.length; ++i){
    if(tab[i]==val){
     return i;
    }
  }
 return -1;
}
```

Beurk, faire plus beau...

Rappel sur les tableaux

Recherche Séquentielle

```
int search(int val, int[] tab){
  int i=0;
  for(; i < tab.length && tab[i]!=val; ++i){
  }
  if(i==tab.length)
      return -1;
  return i;
  /* return i != tab.length ? -1 : i; */
}
```

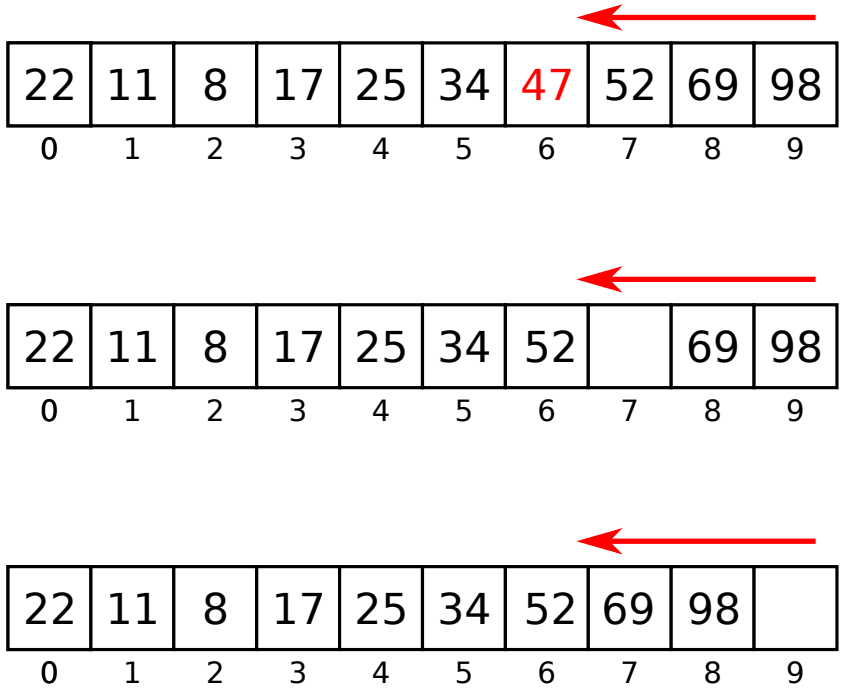
Quel est le coût maximal de cette recherche ?

Dans le pire cas n itérations...

Rappel sur les tableaux

Suppression

Suppression (sans trou dans le tableau) de la valeur 47 :

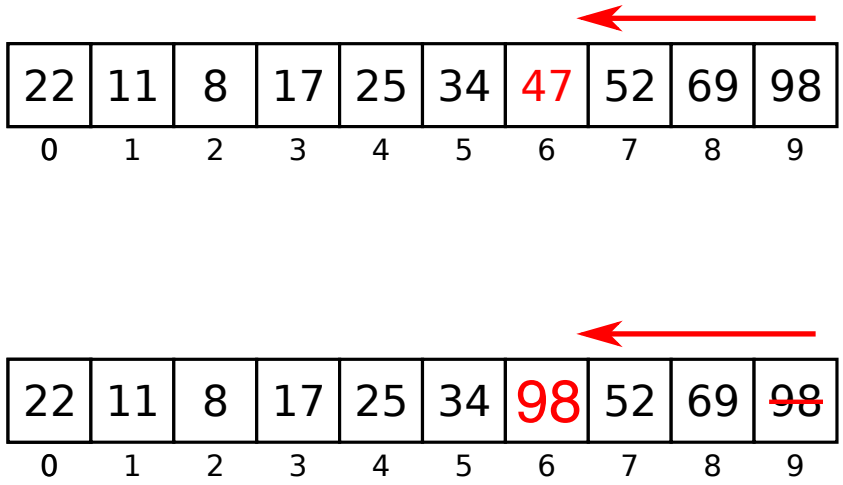


1. Recherche de la valeur 47 (index $i = 6$)
2. Décalage de -1 pour les cases d'index $i + 1$ à $n - 1$

Rappel sur les tableaux

Suppression

Autre suppression (**bien plus efficace !**) de la valeur 47 :



1. Recherche de la valeur 47 (index $i = 6$)
2. Remplacement de 47 par la dernière valeur du tableau.

Rappel sur les tableaux

Suppression

Deux remarques :

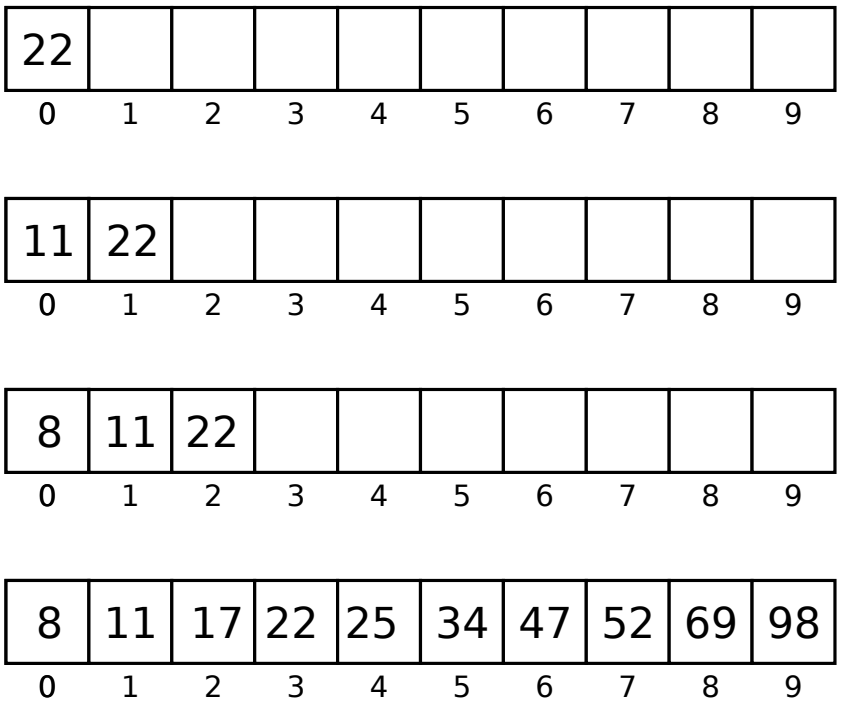
1. Généralement inutile d'effacer la valeur de cette dernière case
2. Plusieurs solutions pour accomplir une même tâche : pour choisir il faut évaluer les coûts !!!

Ici clairement le choix est en faveur de la seconde solution

Tableau ordonné

Insertion

Ajouts successifs des valeurs 22, 11, 8, 17, 25, 34, 47, 52, 69 et 98.



Chaque valeur doit être re-positionnée à la bonne place à chaque insertion

Tableau ordonné

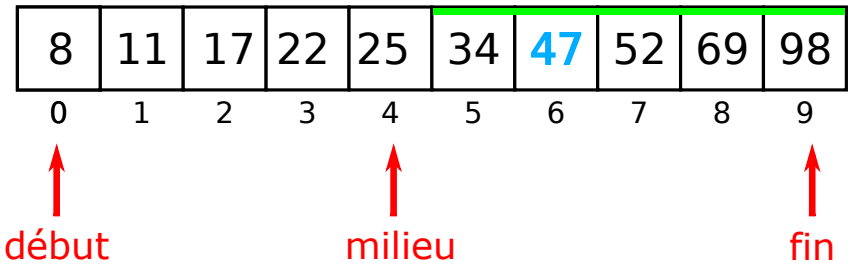
Recherche

- Recherche séquentielle inefficace
- Utilisation du caractère trié du tableau
- Principe de la **recherche dichotomique**

Tableau ordonné

Recherche dichotomique

Recherche de la valeur 47 :



L'espace de recherche est réduit de moitié

Recherche dichotomique

Algorithme

```
int binary_search(int val, int[] tab){
  int b=0; int e=tab.length;
  while(e-b>1){
    int m=(b+e)/2;
    if(val<tab[m])
      e=m;
    else
      b=m;
  }
  return val==tab[b]? b : -1;
}
```

Dans tous les cas $log_2 n$ itérations

Comparaison

- L'insertion est plus rapide dans le tableau non ordonné.
- La recherche est beaucoup plus rapide dans un tableau ordonné.
- La suppression d'une valeur (recherche inclue) prend un temps équivalent dans les deux cas (sinon plus rapide).

Quelle est la meilleure des deux SDD ?

Ça dépend de la situation !

Tableau ordonné ou non ordonné ?

Un tableau ordonné serait plus adapté pour un registre d'employés :

- car peu d'ajouts,
- mais des recherches fréquentes.

Et un tableau non ordonné à un historique des achats :

- car insertions fréquentes,
- mais recherches plus rares.

Les types de données abstraits

Definition

Un type de données abstrait (TDA) est une spécification de données et des opérations réalisables sur ces données.

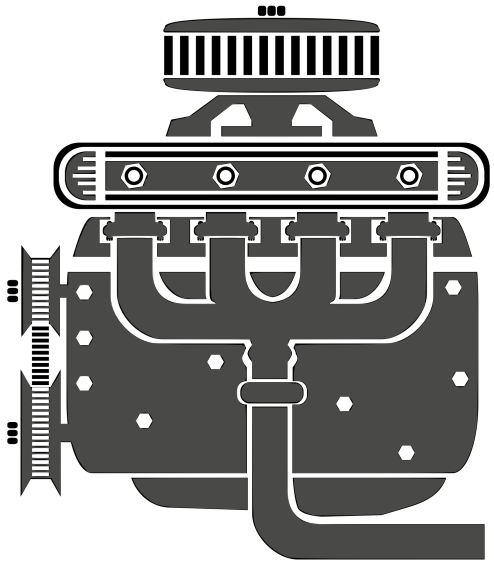
- La spécification est **indépendante** de l'implémentation
- Analogie avec les classes du paradigme objet

Les types de données abstraits

Exemple de la voiture

Les types de données abstraits

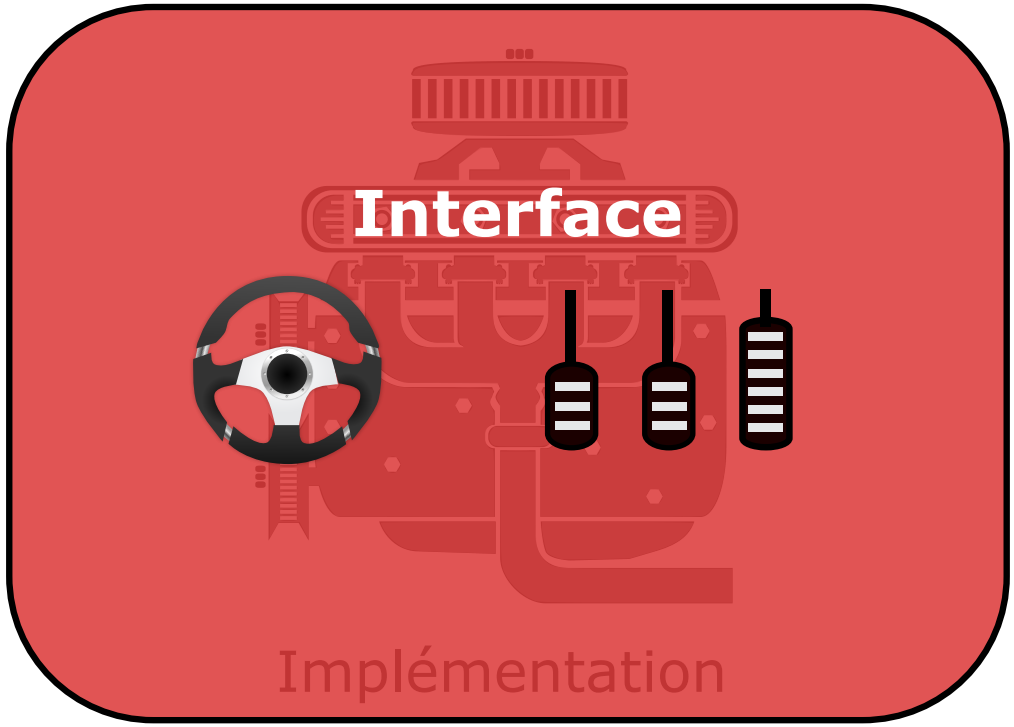
Exemple de la voiture



Implémentation

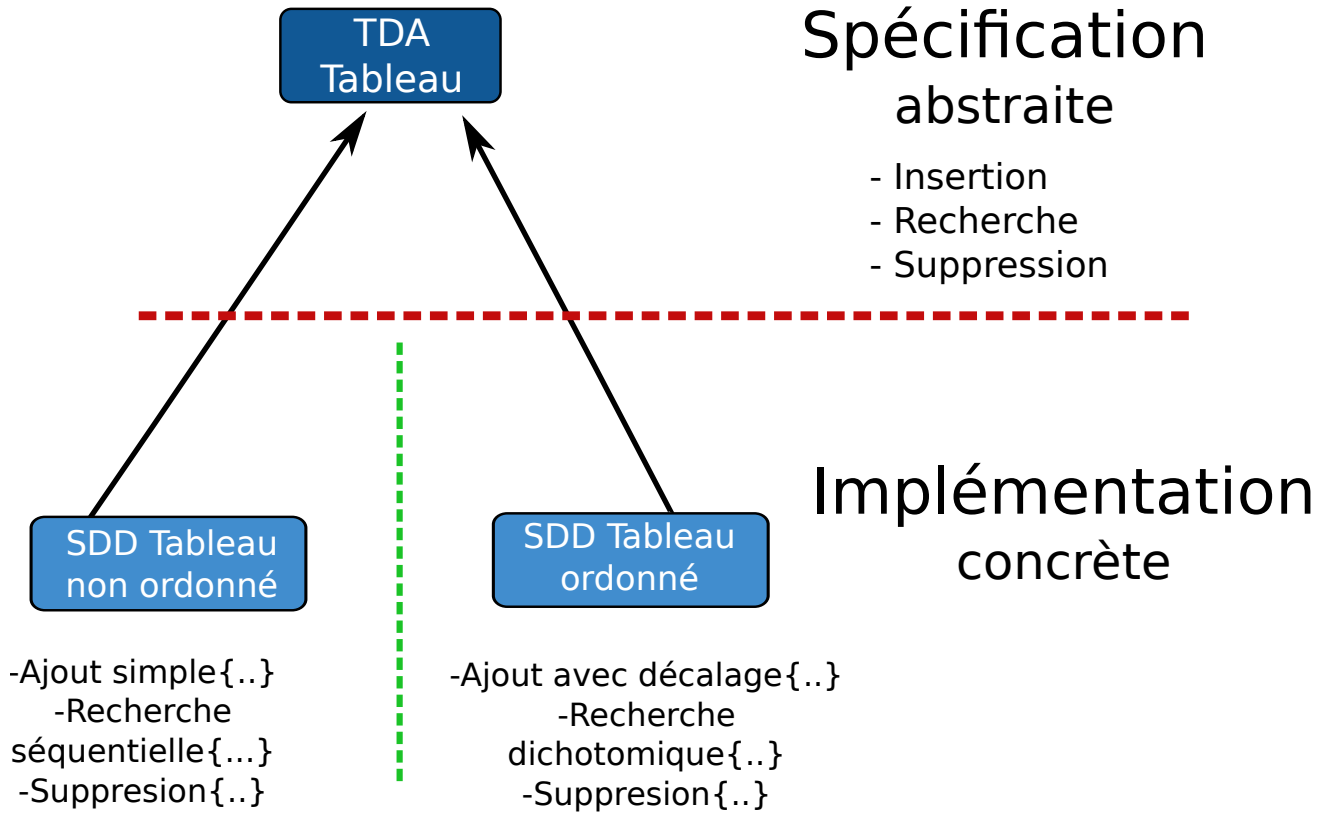
Les types de données abstraits

Exemple de la voiture



L'interface correspond aux opérations possibles sur le TDA voiture.

TDA Tableau


TDA
Tableau
Spécification
abstraite
- Insertion
- Recherche
- Suppression
SDD Tableau
non ordonné
SDD Tableau
ordonné
Implémentation
concrète
-Ajout simple{..}
-Recherche
séquentielle{...}
-Suppresion{..}
-Ajout avec décalage{..}
-Recherche
dichotomique{..}
-Suppresion{..}

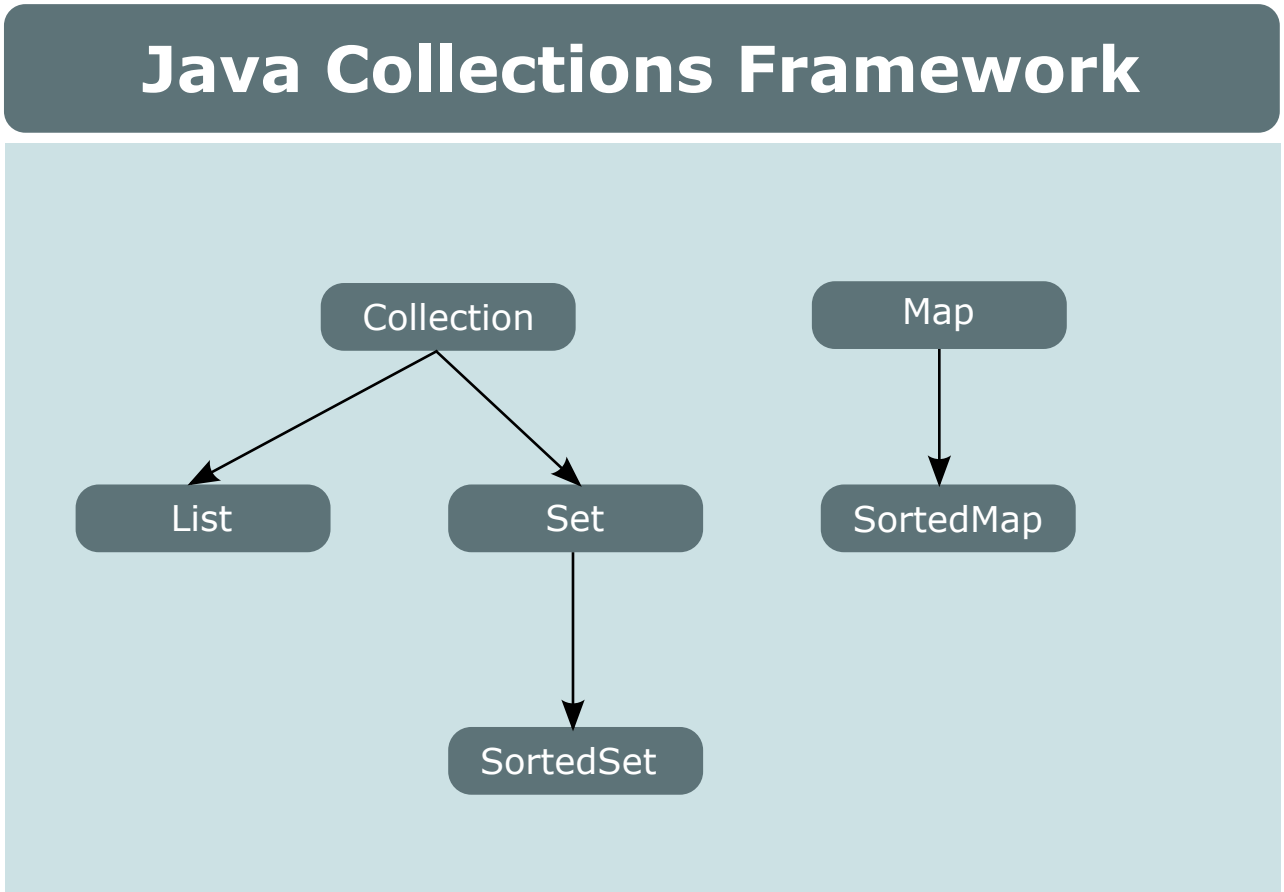
Opérations

Généralement on retrouvera dans les TDA :

- une opération d'insertion,
- une opération de recherche,
- une opération de suppression.

Java Framework Collections

Ensemble de classes et interfaces (les TDA) pour les structures de données en Java.



(R)appel

La généricité

- Élimine plus d'erreurs à la compilation par vérification de type
- Permet de se passer de transtypage (cast)
- Agit par paramétrage de type
- Permet d'écrire des algorithmes fonctionnant avec tous les types

$\Rightarrow$ + d'abstraction

Sans généricité

```
List list = new ArrayList();
list.add("hello N3");
String s = (String) list.get(0);
```

Avec :

```
List<String> list = new ArrayList<String>();
list.add("hello N3");
String s = list.get(0);    // pas de transtypage
```

Une classe non générique :

```
public class Box {
    private Object object;

    public void set(Object object) {
      this.object = object;
    }
    public Object get() { return object; }
}
```

Une classe générique paramétrée :

```
public class Box<T> {
    private T t;    // T pour "Type" par convention

    public void set(T t) {
     this.t = t;
    }
    public T get() { return t; }
}
```

Invocation :

```
Box<Integer> integerBox = new Box<Integer>();
```

Bilan

Nous avons vu dans ce cours :

- Le fonctionnement de la matière pour le semestre,
- pourquoi utiliser des SDD,
- qu'il faut choisir une structure adaptée à sa situation,
- et la notion de type de données abstrait.