

Les listes et les maps en Dart

Table des matières

I. Listes en Dart	3
A. Mise en pratique	4
II. Exercice : Quiz	7
III. Maps en Dart	7
A. Mise en pratique	9
IV. Exercice : Quiz	11
V. Essentiel	12
VI. Auto-évaluation	13
A. Exercice :	13
B. Test	13
Solutions des exercices	14

I. Listes en Dart

Contexte

Lorsque l'on code des fonctionnalités en Dart, il est fréquent de faire appel à des structures de données complexes pour manipuler les données. Par exemple, on voudra stocker un ensemble de valeurs dans un même objet pour les récupérer ensuite de façon ordonnée.

Parmi les structures de données qui existent en Dart, deux sont largement utilisées : **les listes** et **les maps**. Ces deux types d'objets facilitent grandement la manipulation de données de façon ordonnée et intuitive, en mettant à disposition des propriétés et des méthodes.

Pour développer certaines fonctionnalités au sein du code, ces structures de données se révèlent indispensables. Un développeur se doit donc de les maîtriser parfaitement afin de programmer efficacement.

À travers ce cours, nous allons définir ce que sont les listes et les maps, voir comment les utiliser, les déclarer et les manipuler avec diverses méthodes, et mettre tout cela en application.

Définition

Une liste en Dart est un groupe ordonné d'objets, ce que l'on appelle plus communément « *un tableau* » dans les autres langages.

Une liste contient donc une collection d'objets du même type (exemple : String, int). Ceux-ci sont accessibles via leur index (leur position dans la liste). On peut ajouter ou supprimer des objets à cette liste grâce à des méthodes dédiées.

Fondamental Déclaration et assignation

Pour créer une liste, on place les valeurs entre des crochets, séparées par des virgules :

// On crée une liste de nombres.

```
var a = [1, 2, 3];
```

// On crée une liste de String.

```
var b = ["Car", "Boat", "Plane"];
```

// On crée une liste vide (pour la remplir plus tard par exemple).

```
var c = [];
```

Fondamental Propriétés d'une liste

Les propriétés vues ci-dessous seront mises en application dans la sous-partie « *Mise en pratique* ».

Voici les propriétés à connaître concernant une liste :

- **Length** : retourne le nombre d'objets dans la liste.
- **First** : retourne le premier élément de la liste.
- **Last** : retourne le dernier élément de la liste.
- **IsEmpty** : retourne *true* si la liste est vide, *false* sinon.

- Pour accéder à un objet précis d'une liste, on utilise un index (exemple : `list[0]`).

Fondamental Méthodes d'une liste

Les méthodes vues ci-dessous seront mises en application dans la sous-partie « *Mise en pratique* ».

Voici les méthodes à connaître concernant une liste :

- **Add** : ajoute un élément à la fin de la liste.
- **AddAll** : ajoute plusieurs éléments à la fin de la liste.
- **Clear** : efface tous les objets de la liste.
- **IndexOf** : renvoie l'index d'un élément donné au sein de la liste (permet également de tester si un élément donné est présent dans la liste).
- **Insert** : ajoute un élément à un index précis de la liste.
- **InsertAll** : ajoute des éléments à un index précis de la liste.
- **Remove** : efface la première occurrence d'un élément donné.
- **RemoveLast** : efface le dernier élément de la liste.
- **RemoveAt** : efface l'élément à un index précis.
- **Shuffle** : mélange la liste de façon aléatoire.
- **Sort** : permet de trier selon un critère à spécifier.

A. Mise en pratique

Exemple Exemples concernant les propriétés liées à une liste

// On crée une liste de nombres.

```
var a = [1, 2, 3];
```

// On utilise la propriété *length* pour récupérer la longueur de la liste.

```
var b = a.length;
```

// La variable *b* est donc égale à 3.

// On utilise la propriété *first* pour récupérer le premier élément.

```
var c = a.first;
```

// La variable *c* est donc égale à 1.

// On utilise la propriété *last* pour récupérer le dernier élément.

```
var d = a.last;
```

// La variable *d* est donc égale à 3.

// On utilise la propriété *isEmpty* pour savoir si la liste est vide.

```
var e = a.isEmpty;
```

// La variable *e* est donc égale à *false*.

```
// On cherche à accéder au 2e élément de cette liste (et donc à l'index 1, car le 1er élément commence à l'index 0).
```

```
var f = a[1];
```

```
// La variable f est donc égale à 2.
```

Exemple	Exemples pour les méthodes add, addAll, insert, insertAll et clear
----------------	---

```
// On crée une liste de nombres.
```

```
var a = [1, 2, 3];
```

```
// On utilise la méthode add pour ajouter le nombre 4 à la fin de la liste.
```

```
a.add(4);
```

```
print(a);
```

```
// Cela va afficher dans la console : [1, 2, 3, 4]
```

```
// On utilise la méthode addAll pour ajouter plusieurs nombres à la fin de la liste.
```

```
a.addAll([10, 12, 14]);
```

```
print(a);
```

```
// Cela va afficher dans la console : [1, 2, 3, 4, 10, 12, 14]
```

```
// On utilise la méthode insert pour insérer à l'index 4 le nombre 5.
```

```
a.insert(4, 5);
```

```
print(a);
```

```
// Cela va afficher dans la console : [1, 2, 3, 4, 5, 10, 12, 14]
```

```
// On utilise la méthode insertAll pour insérer à l'index 1 plusieurs nombres.
```

```
a.insertAll(1, [2, 2, 2]);
```

```
print(a);
```

```
// Cela va afficher dans la console : [1, 2, 2, 2, 2, 3, 4, 5, 10, 12, 14]
```

```
// On utilise la méthode clear pour effacer tous les objets de la liste.
```

```
a.clear();
```

```
print(a);
```

```
// Cela va afficher dans la console : []
```

Exemple	Exemples pour les méthodes indexOf, remove, removeLast, removeAt
----------------	---

```
// On crée une liste de nombres.
```

```
var a = [1, 2, 3, 4, 5, 6, 7];
```

// On utilise la méthode *indexOf* pour tester si la liste contient le nombre 4 et, si c'est le cas, connaître l'index de cette valeur.

```
var b = a.indexOf(4);
```

// La variable *b* est égale à 3, car le nombre 4 est bien présent à l'index 3.

// On utilise la méthode *remove* pour effacer le nombre 2 de la liste.

```
a.remove(2);
```

```
print(a);
```

// Cela va afficher dans la console : [1, 3, 4, 5, 6, 7]

// On utilise la méthode *removeLast* pour effacer le dernier objet de la liste.

```
a.removeLast();
```

```
print(a);
```

// Cela va afficher dans la console : [1, 3, 4, 5, 6]

// On utilise la méthode *removeAt* pour effacer le nombre présent à l'index 2.

```
a.removeAt(2);
```

```
print(a);
```

// Cela va afficher dans la console : [1, 3, 5, 6]

Exemple Exemples pour les méthodes *shuffle* et *sort*

// On crée une liste de nombres.

```
var a = [1, 48, 63, 4, 1240, 50];
```

// On utilise la méthode *shuffle* pour mélanger la liste.

```
a.shuffle();
```

```
print(a);
```

// Cela va afficher dans la console : [1240, 63, 48, 1, 50, 4]

// Cela aurait pu afficher n'importe quel autre mélange aléatoire de la liste.

// On utilise la méthode *sort* pour trier la liste.

// Par défaut, la méthode *sort* appliquée sur une liste de nombres va la trier dans un ordre croissant.

```
a.sort();
```

```
print(a);
```

// Cela va afficher dans la console : [1, 4, 48, 50, 63, 1240]

// Si la méthode *sort* avait été appliquée sur une liste de *String*, elle les aurait triées par défaut dans l'ordre alphabétique.

Exercice : Quiz

Question 1

Qu'est-ce qu'une liste en Dart ?

- ☐ Un tableau d'objets du même type
- ☐ Une pile d'objets de types différents

Question 2

Quelle réponse ci-dessous est correcte pour implémenter une liste ?

- ☐ { 1, 2, 3 }
- ☐ (1, 2, 3)
- ☐ [1, 2, 3]

Question 3

Il existe une propriété d'une liste qui permet d'en connaître la longueur.

- ☐ Vrai
- ☐ Faux

Question 4

Comment accéder à un objet précis d'une liste en utilisant son index « *i* » ?

- ☐ List.i
- ☐ List[i]
- ☐ List(i)

Question 5

Quelle méthode permet de mélanger une liste ?

- ☐ Sort
- ☐ Clear
- ☐ Shuffle

II. Maps en Dart

Définition

Une map en Dart est une collection de paires clé-valeur, dans laquelle chaque valeur est récupérable via sa clé associée. La map contient un nombre fini de clés, et chaque clé a une valeur qui lui est associée.

Une clé est unique (elle ne peut apparaître qu'une fois dans la map), tandis qu'une même valeur peut être utilisée pour plusieurs clés. L'ensemble des clés sont du même type et l'ensemble des valeurs sont du même type, mais ces deux ensembles ne sont pas nécessairement du même type.

Fondamental **Déclaration et assignation**

Pour créer une map, il existe plusieurs manières équivalentes :

// On crée une map avec des clés de type *String* et des valeurs de type *String*.

```
var person = {
  'firstname': 'thomas',
  'lastname': 'jean',
  'age', '56'
}
```

// On crée une map avec des clés de type *int* et des valeurs de type *String*.

```
var nobleGases = {
  2: 'helium',
  10: 'neon',
  18: 'argon',
} ;
```

// On crée les mêmes maps en utilisant un constructeur, puis en les remplissant via les clés.

```
var person2 = Map<String, String>();
person2['firstname'] = 'thomas' ;
person2['lastname'] = 'jean' ;
person2['age'] = '56' ;

var nobleGases2 = Map<int, String>();
nobleGases2[2] = 'helium' ;
nobleGases2[10] = 'neon' ;
nobleGases2[18] = 'argon' ;
```

Fondamental **Propriétés d'une map**

Les propriétés vues ci-dessous seront mises en application dans la sous-partie « *Mise en pratique* ».

Voici les propriétés à connaître concernant une map :

- **Length** : retourne le nombre de paires clé-valeur dans la map.
- **IsEmpty** : retourne *true* si la map est vide, *false* sinon.
- **Keys** : retourne un objet itérable comprenant les clés de la map.
- **Values** : retourne un objet itérable comprenant les valeurs de la map.

Fondamental **Méthodes d'une map**

Les méthodes vues ci-dessous seront mises en application dans la sous-partie « *Mise en pratique* ».

Voici les méthodes à connaître concernant une map :

- **AddAll** : ajoute plusieurs paires clé-valeur à la map.
- **Clear** : efface toutes les paires clé-valeur de la map.
- **ContainsKey** : teste si la map contient une clé donnée.
- **ContainsValue** : teste si la map contient une valeur donnée.
- **ForEach** : permet de parcourir la map en effectuant une action à chaque itération.
- **Remove** : efface une paire clé-valeur de la map pour une clé donnée.

A. Mise en pratique**Exemple** **Exemples concernant les propriétés liées à une map**

// On crée une map qui lie des entiers à des Strings.

```
var a = {  
  1 : "Hello",  
  43 : "Bonjour",  
  24 : "Hola"  
};
```

// On utilise la propriété *length* pour récupérer la longueur de la map.

```
var b = a.length;
```

// La variable *b* est donc égale à 3.

// On utilise la propriété *isEmpty* pour savoir si la map est vide.

```
var c = a.isEmpty;
```

// La variable *c* est donc égale à *false*.

// On utilise la propriété *keys* pour récupérer les clés de la map.

```
for (var key in a.keys)  
  print(key);
```

// Cela va afficher dans la console :

// 1

// 43

// 24

// On utilise la propriété *values* pour récupérer les valeurs de la map.

```
for (var value in a.values)  
  print(value);
```

```
// Cela va afficher dans la console :
// Hello
// Bonjour
// Hola
```

Exemple Exemples pour les méthodes addAll, remove et clear

```
// On crée une map qui lie des entiers à des Strings.
var a = {
  1 : "Hello",
  43 : "Bonjour",
  24 : "Hola"
};

// On utilise la méthode addAll pour ajouter des paires clé-valeur.
a.addAll({34: "Hey", 2: "Salut"});
print(a);
// Cela affiche dans la console :
// {1: Hello, 43: Bonjour, 24: Hola, 34: Hey, 2: Salut}

// On utilise la méthode remove pour effacer la paire clé-valeur ayant pour clé la valeur 24.
a.remove(24);
print(a);
// Cela affiche dans la console :
// {1: Hello, 43: Bonjour, 34: Hey, 2: Salut}

// On utilise la méthode clear pour effacer toutes les paires.
a.clear();
print(a);
// Cela affiche dans la console :
// {}
```

Exemple Exemples pour les méthodes containsKey et containsValue

```
// On crée une map qui lie des entiers à des Strings.
var a = {
  1 : "Hello",
  43 : "Bonjour",
  24 : "Hola"
};
```

```
// On teste si la map contient la clé 49.
```

```
var b = a.containsKey(49);
```

```
// La variable b est égale à false, car il n'y a pas de clé égale à 49.
```

```
// On teste si la map contient la valeur "Bonjour".
```

```
var c = a.containsValue("Bonjour");
```

```
// La variable c est égale à true, car il y a bien une valeur égale à "Bonjour".
```

Exemple	Exemple pour la méthode <code>forEach</code>
----------------	---

```
// On crée une map qui lie des entiers à des Strings.
```

```
var a = {  
  1 : "Hello",  
  43 : "Bonjour",  
  24 : "Hola"  
};
```

```
// On utilise forEach afin de parcourir toutes les paires clé-valeur.
```

```
a.forEach((k, v) {  
  print("Clé : ${k}");  
  print("Valeur : ${v}");  
});
```

```
// Cela affiche dans la console :
```

```
// Clé : 1
```

```
// Valeur : Hello
```

```
// Clé : 43
```

```
// Valeur : Bonjour
```

```
// Clé : 24
```

```
// Valeur : Hola
```

Exercice : Quiz

Question 1

Qu'est-ce qu'une map en Dart ?

- ☐ Une liste d'objets
- ☐ Une collection de paires clé-valeur

Question 2

Quelle réponse ci-dessous est correcte pour implémenter une map ?

- ☐ `var map = { 2: 'hello', 10: 'bonjour' };`
- ☐ `var map = [2: 'hello', 10: 'bonjour'];`
- ☐ `var map = { 2: 'hello' | 10: 'bonjour' };`

Question 3

Il existe une propriété d'une map qui permet d'en récupérer toutes les clés.

- ☐ Vrai
- ☐ Faux

Question 4

Comment accéder à une valeur en utilisant une clé ?

- ☐ `map.key`
- ☐ `map(key)`
- ☐ `map[key]`

Question 5

Quelle méthode permet de tester si la map contient une valeur en particulier ?

- ☐ `ContainsKey`
- ☐ `ContainsValue`
- ☐ `ForEach`

III. Essentiel

Les structures de données telles que les listes et les maps sont très utiles lorsqu'une solution informatique requiert des ensembles de valeurs complexes. Que ce soit pour stocker des valeurs de façon ordonnée ou définir des associations clé-valeur, il est nécessaire de savoir les manipuler en langage Dart.

De nombreuses méthodes accompagnent les listes et les maps. Elles permettent au développeur d'être plus efficace et apportent des fonctionnalités indispensables : tri, test, longueur, manipulation en tout genre.

Dans ce cours, nous avons analysé et mis en application les principales connaissances à avoir concernant ces structures de données. Lorsqu'un doute subsiste ou que vous souhaitez avoir davantage de détails concernant une propriété ou une méthode, vous pouvez consulter la documentation fournie sur le site Dart SDK¹

Avec la pratique, les listes et les maps se révèlent être de précieux outils pour coder une solution, et il devient facile de les manipuler pour exploiter leur potentiel.

¹ <https://api.dart.dev/stable/2.9.0/index.html>

IV. Auto-évaluation

A. Exercice :

Dans cet exercice, nous allons utiliser et mettre en application les différentes structures de données vues dans ce cours. Les réponses doivent être ajoutées successivement à un programme en Dart que vous devez exécuter (pour simplifier l'environnement, le DartPad peut être utilisé : DartPad¹)

Question 1

Déclarez une liste comprenant les entiers 103, 45 et 68. Affichez la longueur de cette liste dans la console.

Question 2

Utilisez la méthode de votre choix pour récupérer le dernier élément de la liste de façon générique. Affichez le résultat dans la console.

Question 3

Utilisez la méthode correspondante pour ajouter l'entier 74 à l'index 2 de la liste. Affichez le résultat dans la console.

Question 4

Utilisez la méthode correspondante pour trier les entiers de la liste dans un ordre croissant. Affichez le résultat dans la console.

Question 5

Déclarez une map qui à la clé "Hello" associe 3, et à la clé "Bonjour" associe 5. Affichez la longueur de cette map dans la console. Affichez le résultat dans la console.

Question 6

Utilisez la méthode correspondante pour ajouter la paire clé-valeur { "Hi" : 1 } à la map. Affichez le résultat dans la console.

Question 7

Affichez l'ensemble des clés de la map dans la console en utilisant la propriété correspondante.

Question 8

Effacez toutes les paires clé-valeur de la map puis affichez le résultat dans la console.

B. Test

Dans cet exercice, nous allons utiliser et mettre en application les différentes structures de données vues dans ce cours. Les réponses doivent être ajoutées successivement à un programme en Dart que vous devez exécuter (pour simplifier l'environnement, le DartPad peut être utilisé : DartPad²)

Exercice : Quiz

Question 1

Comment sont définies les listes et les maps en programmation ?

- ☐ Ce sont des structures de données
- ☐ Ce sont des fonctions complexes

Question 2

À quelle structure de données correspond la notion clé-valeur ?

- ☐ Les listes
- ☐ Les maps

¹ <https://dartpad.dev/>

² <https://dartpad.dev/>

Question 3

Quelle structure de données peut-on trier avec la méthode *sort* ?

- ☐ Les listes
- ☐ Les maps

Question 4

Dans une map, les clés et les valeurs doivent être du même type.

- ☐ Vrai
- ☐ Faux

Question 5

Dans quel cas a-t-on besoin de faire appel à des listes ou des maps dans notre code ?

- ☐ Lorsque la manipulation des données se complexifie
- ☐ Lorsque l'on a besoin de fonctions mathématiques avancées
- ☐ Pour générer des nombres aléatoires

Solutions des exercices


Exercice p. 7 Solution n°1

Question 1

Qu'est-ce qu'une liste en Dart ?

☒ Un tableau d'objets du même type

☐ Une pile d'objets de types différents

 Une liste peut être considérée comme un tableau (un groupe ordonné) d'objets du même type qui sont accessibles via leur index (leur position dans la liste). Il s'agit de la définition que l'on a d'un tableau dans d'autres langages.


Question 2

Quelle réponse ci-dessous est correcte pour implémenter une liste ?

☐ { 1, 2, 3 }

☐ (1, 2, 3)

☒ [1, 2, 3]


 Les valeurs d'une liste sont toujours comprises entre des crochets.

Question 3

Il existe une propriété d'une liste qui permet d'en connaître la longueur.

☒ Vrai

☐ Faux

 Il s'agit de la propriété *length*, très souvent utilisée.


Question 4

Comment accéder à un objet précis d'une liste en utilisant son index « *i* » ?

☐ List.i

☒ List[i]

☐ List(i)

 L'index doit être placé entre crochets.


Question 5

Quelle méthode permet de mélanger une liste ?

☐ Sort

☐ Clear


☒ Shuffle

 La méthode *shuffle* (« mélanger » en anglais) permet de mélanger une liste de façon aléatoire, tandis que la méthode *clear* permet d'effacer le contenu d'une liste et que la méthode *sort* permet d'en trier les éléments.

Exercice p. 11 Solution n°2


Question 1

Qu'est-ce qu'une map en Dart ?

- ☐ Une liste d'objets
- ☒ Une collection de paires clé-valeur
-  Une map contient un nombre fini de clé, et chaque clé possède une valeur qui lui est associée.


Question 2

Quelle réponse ci-dessous est correcte pour implémenter une map ?

- ☒ `var map = { 2: 'hello', 10: 'bonjour' };`
- ☐ `var map = [2: 'hello', 10: 'bonjour'];`
- ☐ `var map = { 2: 'hello' | 10: 'bonjour' };`
-  Les paires clé-valeur sont contenues entre accolades, séparées par des virgules.


Question 3

Il existe une propriété d'une map qui permet d'en récupérer toutes les clés.

- ☒ Vrai
- ☐ Faux
-  Il s'agit de la propriété *keys*, qui retourne un objet itérable.


Question 4

Comment accéder à une valeur en utilisant une clé ?

- ☐ `map.key`
- ☐ `map(key)`
- ☒ `map[key]`
-  La clé doit être placée entre crochets.

Question 5

Quelle méthode permet de tester si la map contient une valeur en particulier ?

- ☐ `ContainsKey`
- ☒ `ContainsValue`
- ☐ `ForEach`
-  La méthode *containsValue* teste si la map contient une valeur donnée.

Exercice p. Solution n°3

```
// On implémente la liste d'entiers avec les nombres 103, 45 et 68.  
var a = [103, 45, 68];  
// On utilise la propriété length pour récupérer la longueur de la liste.  
print(a.length);  
// Cela affiche dans la console : 3
```

Exercice p. Solution n°4

```
// On utilise la propriété last pour récupérer le dernier élément de la liste.  
print(a.last);  
// Il est également possible d'utiliser la longueur pour cela.  
print(a[a.length - 1]);  
// Cela affiche dans la console : 68
```

Exercice p. Solution n°5

```
// On utilise la méthode insert pour insérer l'entier 74 à l'index 2.  
a.insert(2, 74);  
print(a);  
// Cela affiche dans la console : [103, 45, 74, 68]
```

Exercice p. Solution n°6

```
// On utilise la méthode sort pour trier la liste par ordre croissant.  
a.sort();  
print(a);  
// Cela affiche dans la console : [45, 68, 74, 103]
```

Exercice p. Solution n°7

```
// On implémente la map avec les paires clé-valeur : "Hello" 3 et "Bonjour" 5.  
var b = {"Hello" : 3, "Bonjour" : 5};  
// On utilise la propriété length pour récupérer la longueur de la map.  
print(b.length);  
// Cela affiche dans la console : 2
```

Exercice p. Solution n°8

// On utilise la méthode *addAll* pour insérer la paire clé-valeur : "Hi" 1.

```
b.addAll({"Hi" : 1});
```

```
print(b);
```

// Cela affiche dans la console : {Hello: 3, Bonjour: 5, Hi: 1}

Exercice p. Solution n°9

// On utilise la propriété *keys* pour récupérer les clés de la map.

```
print(b.keys);
```

// Cela affiche dans la console : (Hello, Bonjour, Hi)

Exercice p. Solution n°10

// On utilise la méthode *clear* pour effacer toutes les paires clé-valeur.

```
b.clear();
```

```
print(b);
```

// Cela affiche dans la console : {}

Exemple de programme final :

```
void main() {  
    var a = [103, 45, 68];  
  
    print(a.length);  
  
    print(a.last);  
    // print(a[a.length - 1]);  
  
    a.insert(2, 74);  
  
    print(a);  
  
    a.sort();  
  
    print(a);  
  
    var b = {"Hello" : 3, "Bonjour" : 5};  
  
    print(b.length);  
  
    b.addAll({"Hi" : 1});  
  
    print(b);  
  
    print(b.keys);  
  
    b.clear();  
  
    print(b);  
}
```

Résultat affiché dans la console :


Console

```
3
68
[103, 45, 74, 68]
[45, 68, 74, 103]
2
{Hello: 3, Bonjour: 5, Hi: 1}
(Hello, Bonjour, Hi)
{}
```

Exercice p. 13 Solution n°11


Question 1

Comment sont définies les listes et les maps en programmation ?

- ☒ Ce sont des structures de données
- ☐ Ce sont des fonctions complexes
-  Les structures de données sont des ensembles de données structurées qui permettent de manipuler de façon ordonnée des données, comme c'est le cas avec les maps et les listes.


Question 2

À quelle structure de données correspond la notion clé-valeur ?

- ☐ Les listes
- ☒ Les maps
-  Une map est une collection de paires clé-valeur.

Question 3

Quelle structure de données peut-on trier avec la méthode *sort* ?


- ☒ Les listes
- ☐ Les maps
-  La méthode *sort* permet effectivement de trier des listes.

Question 4

Dans une map, les clés et les valeurs doivent être du même type.

☐ Vrai

☒ Faux

 Les clés peuvent par exemple être de type *int* tandis qu'elles sont associées à des valeurs de type *String* (tout est possible au niveau des types).


Question 5

Dans quel cas a-t-on besoin de faire appel à des listes ou des maps dans notre code ?

☒ Lorsque la manipulation des données se complexifie

☐ Lorsque l'on a besoin de fonctions mathématiques avancées

☐ Pour générer des nombres aléatoires

 Les listes et les maps sont nécessaires dès lors qu'il faut manipuler de façon ordonnée des ensembles d'objets.