Programmation orienté objet

Dr. Ines Ben Tekaya

04 Décembre 2018

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Chapitre 4 : Array List et les Hashmap

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Objectifs

- Maîtriser la création de collections : ArrayList et HashMap.
- Utilisez des ArrayList et HashMap pour stocker et manipuler des données.

Compter des mots différents

- Compter le nombre de mots différents ou d'adresses IP ou des éléments de données de n'importe quel type.
  - Nous avons compté 'c', 'g', 't' et 'a'.
  - Nous avons compté 'A', 'B', ... 'Z'.
  - Première étape dans un texte contenant : "le", "chat", "albatros" :
    - Compter le nombre de mots différents

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Utilisation de StorageResource

- Utiliser StorageResource, c'est facile.

```
public class CountWords {

    StorageResource myWords;

    public CountWords() {
        myWords = new StorageResource();
    }

    public int getCount(){
        return myWords.size();
    }

    public void readWords(String source){
        myWords.clear();
        if (source.startsWith("http")){
            URLResource resource = new URLResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
        else {
            FileResource resource = new FileResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
    }
```

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Utilisation de StorageResource

- Utiliser StorageResource, c'est facile.
  - Compter tous les mots d'un fichier ou d'une URL

```
public class CountWords {

    StorageResource myWords;

    public CountWords() {
        myWords = new StorageResource();
    }

    public int getCount(){
        return myWords.size();
    }

    public void readWords(String source){
        myWords.clear();
        for(String word : resource.words()){
            myWords.add(word.toLowerCase());
        }
    }
    else {
        FileResource resource = new FileResource(source);
        for(String word : resource.words()){
            myWords.add(word.toLowerCase());
        }
```

Utilisation de StorageResource

- Utiliser StorageResource, c'est facile.
  - Compter tous les mots d'un fichier ou d'une URL
  - Ajouter chacun à StorageResource

```
public class CountWords {

    StorageResource myWords;

    public CountWords() {
        myWords = new StorageResource();
    }

    public int getCount(){
        return myWords.size();
    }

    public void readWords(String source){
        myWords.clear();
        if (source.startsWith("http")){
            myWords.add(word.toLowerCase());
            myWords.add(word.toLowerCase());
        }
        }
        else {
            FileResource resource = new FileResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            myWords.add(word.toLowerCase());
        }
    }
```

Utilisation de StorageResource

- Utiliser StorageResource, c'est facile.
  - Compter tous les mots d'un fichier ou d'une URL
  - Ajouter chacun à StorageResource
    - Utilisez .size ()

```
public class CountWords {

    StorageResource myWords;

    public CountWords() {
        myWords = new StorageResource();
    }

public int getCount(){
    return myWords.size();
}

    public void readWords(String source){
        myWords.clear();
        if (source.startsWith("http")){
            URLResource resource = new URLResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
        else {
            FileResource resource = new FileResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
```

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Utilisation de StorageResource

- Utiliser StorageResource, c'est facile.
  - Compter tous les mots d'un fichier ou d'une URL
  - Ajouter chacun à StorageResource
    - Utilisez .size ()
  - Mots différents ?

```
public class CountWords {

    StorageResource myWords;

    public CountWords() {
        myWords = new StorageResource();
    }

    public int getCount(){
        return myWords.size();
    }

    public void readWords(String source){
        myWords.clear();
        if (source.startsWith("http")){
            URLResource resource = new URLResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
        else {
            FileResource resource = new FileResource(source);
            for(String word : resource.words()){
                myWords.add(word.toLowerCase());
            }
        }
    }
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Modification du code pour des mots uniques

- Champ : StorageResource myWords.
  - Stocker tous les mots lus dans un fichier.

```
FileResource resource = new FileResource(source);
for(String word : resource.words()){
    myWords.add(word.toLowerCase());
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Modification du code pour des mots uniques

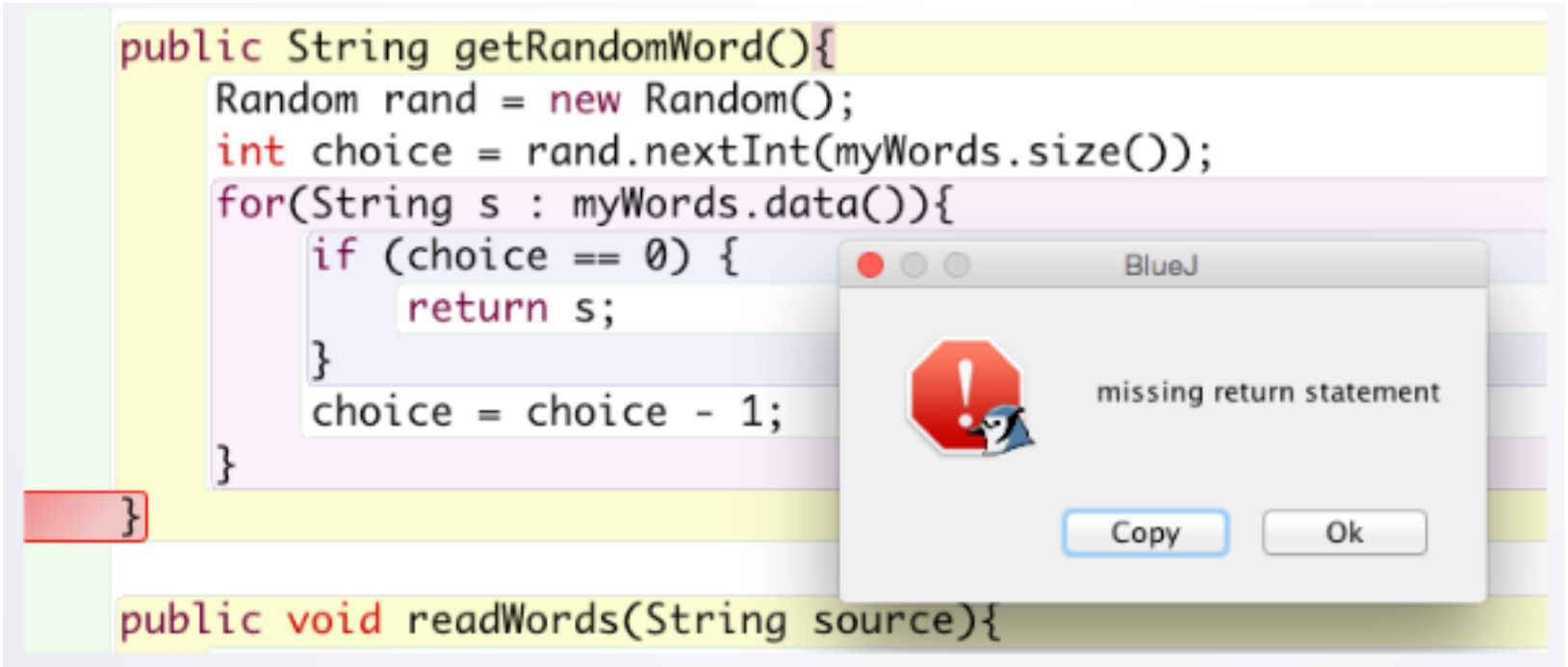
- Champ : StorageResource myWords.
  - Stocker tous les mots lus dans un fichier.
  - seulement des mots uniques / différents.

```
FileResource resource = new FileResource(source);
for(String word : resource.words()){
    myWords.add(word.toLowerCase());
}
```

```
FileResource resource = new FileResource(source);
for(String word : resource.words()){
    word = word.toLowerCase();
    if (! myWords.contains(word)){
        myWords.add(word);
    }
}
```

Choix aléatoire de StorageResource

- StorageResource accédé en tant qu'iterable.
  - Doit utiliser la boucle pour obtenir tous les éléments.
  - Même si nous nous arrêtons tôt, les problèmes de codage.



Choix aléatoire de StorageResource

- StorageResource accédé en tant qu'iterable.
  - Doit utiliser la boucle pour obtenir tous les éléments.
  - Même si nous nous arrêtons tôt, les problèmes de codage.
- Serait plus rapide et plus simple avec String[]
  - Mais ne connais pas la capacité avant de lire !

```
public String getRandomWord(String[] words) {
    Random rand = new Random();
    int index = rand.nextInt(words.length);
    return words[index];
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayList comme solution

- Classe ArrayList dans le package java.util
  - S'agrandit au besoin en utilisant la méthode .add.
  - Fournit un accès via index à n'importe quel élément de la liste.
  - Indispensable pour implémenter StorageResource !
- Syntaxe de base, nous verrons l'utilisation dans le code

```
ArrayList<String> words = new ArrayList<String>();
words.add("hello");
words.add("world");
String s = words.get(1);
words.set(0,"goodbye");
```

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

ArrayList comme solution

- Classe ArrayList dans le package java.util
  - S'agrandit au besoin en utilisant la méthode .add.
  - Fournit un accès via index à n'importe quel élément de la liste.
  - Indispensable pour implémenter StorageResource !
- Syntaxe de base, nous verrons l'utilisation dans le code

```
ArrayList<String> words = new ArrayList<String>();
words.add("hello");
words.add("world");
String s = words.get(1);
words.set(0,"goodbye");
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayList comme solution

- Classe ArrayList dans le package java.util
  - S'agrandit au besoin en utilisant la méthode .add.
  - Fournit un accès via index à n'importe quel élément de la liste.
  - Indispensable pour implémenter StorageResource !
- Syntaxe de base, nous verrons l'utilisation dans le code

```
ArrayList<String> words = new ArrayList<String>();
words.add("hello");
words.add("world");
String s = words.get(1);
words.set(0,"goodbye");
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayList comme solution

- Classe ArrayList dans le package java.util
  - S'agrandit au besoin en utilisant la méthode .add.
  - Fournit un accès via index à n'importe quel élément de la liste.
  - Indispensable pour implémenter StorageResource !
- Syntaxe de base, nous verrons l'utilisation dans le code

```
ArrayList<String> words = new ArrayList<String>();
words.add("hello");
words.add("world");
String s = words.get(1);
words.set(0,"goodbye");
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayList comme solution

- Classe ArrayList dans le package java.util
  - S'agrandit au besoin en utilisant la méthode .add.
  - Fournit un accès via index à n'importe quel élément de la liste.
  - Indispensable pour implémenter StorageResource !
- Syntaxe de base, nous verrons l'utilisation dans le code

```
ArrayList<String> words = new ArrayList<String>();
words.add("hello");
words.add("world");
String s = words.get(1);
words.set(0,"goodbye");
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Arrays et ArrayList

- String[]a et ArrayList<String> b
  - a[k] comparé b.get() et b.set() .
- int[]ac et ArrayList<Integer> bc
  - Problèmes liés aux conversions int/Integer
  - ac[index]++ fonctionne
  - bc.get(index)++ Ne Fonctionne Pas !
- Mais, les tableaux ne grandissent pas de taille, c'est une préoccupation !

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Collection indexable, comme un tableau, mais pouvant être développée !
  - Accès via index entier commençant par zéro.
  - import java.util.ArrayList ou java.util. * ;

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Collection indexable, comme un tableau, mais pouvant être développée !
  - Accès via index entier commençant par zéro.
  - import java.util.ArrayList ou java.util. * ;
  - Créer avec générique : ArrayList<String>

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Collection indexable, comme un tableau, mais pouvant être développée !
  - Accès via index entier commençant par zéro.
  - import java.util.ArrayList ou java.util. * ;
  - Créer avec générique : ArrayList<Integer>

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Collection indexable, comme un tableau, mais pouvant être développée !
  - Accès via index entier commençant par zéro.
  - import java.util.ArrayList ou java.util. * ;
  - Créer avec générique : ArrayList<Integer>
- Méthodes courantes pour ArrayList
  - .add(elt) : ajouté à la fin d'ArrayList
  - .size() : renvoie le nombre d'éléments dans ArrayList
  - .get(index) : retourne les éléments à l'index
  - .set(index,elt) : assigner elt à l'emplacement d'index

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Accéder aux éléments via l'indexation
  - Commencez avec zéro, boucle à moins de size().
  - Accès via get (index)
  - Ne pas appeler .remove() pendant l'itération

```
ArrayList<String> a = new ArrayList<String>();
// add elements

for(int k=0; k < a.size(); k++){
    String s = a.get(k);
    // process s
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Résumé sur ArrayList

- Accéder aux éléments via l'indexation
  - Commencez avec zéro, boucle à moins de size().
  - Accès via get (index)
  - Ne pas appeler .remove() pendant l'itération

```
ArrayList<String> a = new ArrayList<String>();
// add elements

for(String s : a){
    // process s
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Live coding !

Nouvelle structure

- Compter les fréquences : 'CGAT' à 'AB . . . YZ'
  - Étendu aux mots avec deux ArrayLists.
  - De 4 à 26 à . . .
- De ArrayLists parallèles à HashMap
  - Beaucoup plus rapide pour compter les fréquences de mots

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayLists parallèles

- Vu le code pour compter les occurrences de mots, grâce à l'appel de .indexOf (s)

```
public void findUnique(){
        FileResource resource = new FileResource();
        for(String s : resource.words()){
            s = s.toLowerCase();
            int index = myWords.indexOf(s);
            if (index == -1){
                myWords.add(s);
                myFreqs.add(1);
            }
            else {
                int freq = myFreqs.get(index);
                myFreqs.set(index,freq+1);
            }
        }
}
```

ArrayLists parallèles

- Vu le code pour compter les occurrences de mots, grâce à l'appel de .indexOf (s)

```
public void findUnique(){
        FileResource resource = new FileResource();
        for(String s : resource.words()){
            s = s.toLowerCase();
            int index = myWords.indexOf(s);
            if (index == -1){
                myWords.add(s);
                myFreqs.add(1);
            }
            else {
                int freq = myFreqs.get(index);
                myFreqs.set(index,freq+1);
            }
        }
}
```

ArrayLists parallèles

- Vu le code pour compter les occurrences de mots, grâce à l'appel de .indexOf (s)

```
public void findUnique(){
        FileResource resource = new FileResource();
        for(String s : resource.words()){
            s = s.toLowerCase();
            int index = myWords.indexOf(s);
            if (index == -1){
                myWords.add(s);
                myFreqs.add(1);
            }
            else {
                int freq = myFreqs.get(index);
                myFreqs.set(index,freq+1);
            }
        }
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ArrayLists parallèles

- Vu le code pour compter les occurrences de mots, grâce à l'appel de .indexOf (s)

```
public void findUnique(){
        FileResource resource = new FileResource();
        for(String s : resource.words()){
            s = s.toLowerCase();
            int index = myWords.indexOf(s);
            if (index == -1){
                myWords.add(s);
                myFreqs.add(1);
            }
            else {
                int freq = myFreqs.get(index);
                myFreqs.set(index,freq+1);
            }
        }
 }
```

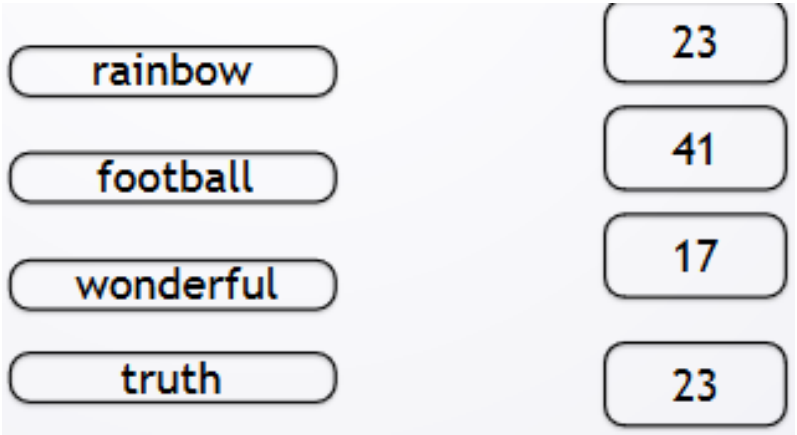
Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.

Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.
  - Plus mathématique que géographique
  - La clé est un élément du domaine, la valeur est ce que la clé corrspond à l'intervalle
- Rechercher la clé, pour obtenir la valeur associée.

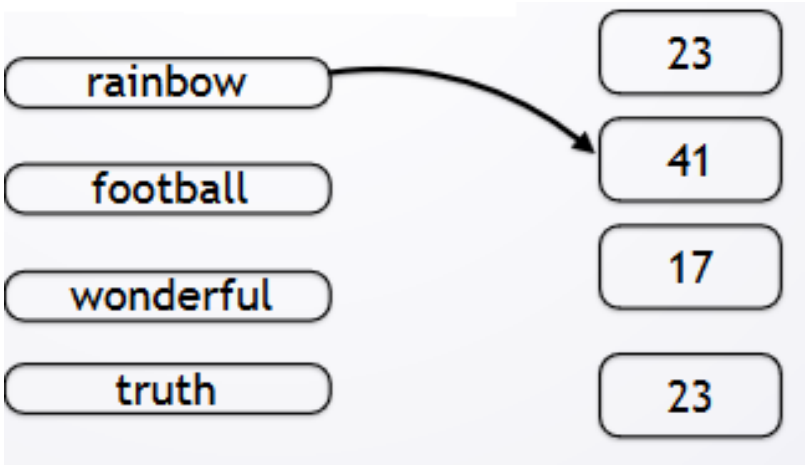
map.get("rainbow")



Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.
  - Plus mathématique que géographique
  - La clé est un élément du domaine, la valeur est ce que la clé corrspond à l'intervalle
- Rechercher la clé, pour obtenir la valeur associée.

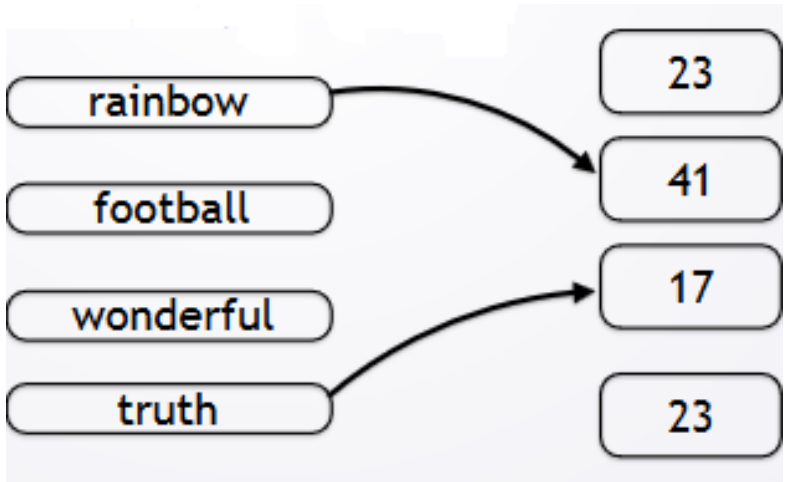
map.get("rainbow")



Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.
  - Plus mathématique que géographique
  - La clé est un élément du domaine, la valeur est ce que la clé corrspond à l'intervalle
- Rechercher la clé, pour obtenir la valeur associée.

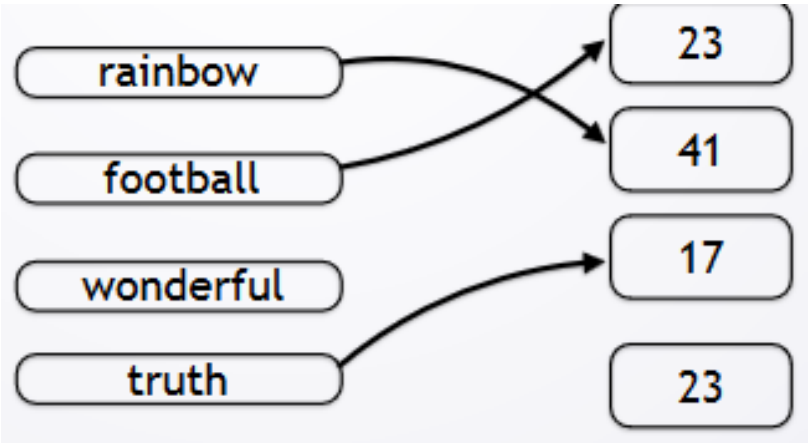
```
map.get("rainbow")
map.get("truth")
```



Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.
  - Plus mathématique que géographique
  - La clé est un élément du domaine, la valeur est ce que la clé corrspond à l'intervalle
- Rechercher la clé, pour obtenir la valeur associée.

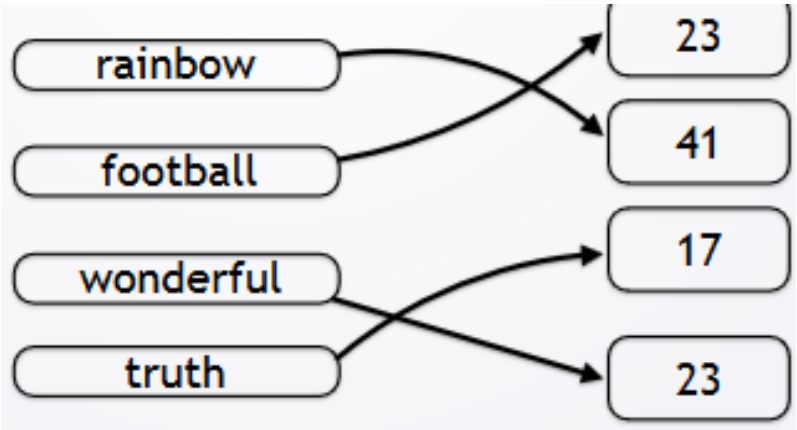
```
map.get("rainbow")
map.get("truth")
map.get("football")
```



Motivation de HashMap

- Un HashMap est une classe qui associe des clés avec des valeurs, généralement appelé map.
  - Plus mathématique que géographique
  - La clé est un élément du domaine, la valeur est ce que la clé corrspond à l'intervalle
- Rechercher la clé, pour obtenir la valeur associée.

```
map.get("rainbow")
map.get("truth")
map.get("football")
map.get("wonderful")
```



UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.

```
public void findUnique(){
        FileResource resource = new FileResource();
        for(String s : resource.words()){
            s = s.toLowerCase();
            int index = myWords.indexOf(s);
            if (index == -1){
                myWords.add(s);
                myFreqs.add(1);
            }
            else {
                int freq = myFreqs.get(index);
                myFreqs.set(index,freq+1);
            }
        }
}
```

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

PANTHEON SORBONNE

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

PANTHEON SORBONNE

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.
  - La clé est-elle nouvelle, non vue précédemment ? mettre la valeur 1, sinon mettre à jour.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.
  - La clé est-elle nouvelle, non vue précédemment ? mettre la valeur 1, sinon mettre à jour.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

Mise à jour des valeurs dans HashMap

- Un HashMap remplace deux ArrayLists.
  - Key est String, la valeur associée est Integer.
  - La clé est-elle nouvelle, non vue précédemment ? mettre la valeur 1, sinon mettre à jour.

```
public void countWordsMap(){
    FileResource resource = new FileResource();
    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for(String w : resource.words()){
      w = w.toLowerCase();
      if (!map.containsKey(w)){
        map.put(w,1);
      }
      else {
        map.put(w,map.get(w)+1);
      }
}
```

Accéder à toutes les valeurs de map

- Affichage de toutes les valeurs dans des ArrayLists parallèles, est utilisée pour la boucle
- Un index est utilisé pour word et freq.

```
public void printWords(){
    for(int k=0; k < myFreqs.size(); k++){
      System.out.println(myFreqs.get(k)+"\t"+myWords.get(k));
}
```

Accéder à toutes les valeurs de map

- Affichage de toutes les valeurs dans des ArrayLists parallèles, est utilisée pour la boucle.
- Un index est utilisé pour word et freq.
- Affichage de toutes les valeurs dans map nécessite une boucle sur les clés.
- La valeur de get est associée à la clé.

```
public void printWords(){
    for(String s : myMap.keySet()){
      System.out.println(myMaps.get(s)+"\t"+s);
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Accéder à toutes les valeurs de map

- Affichage de toutes les valeurs dans des ArrayLists parallèles, est utilisée pour la boucle.
- Un index est utilisé pour word et freq.
- Affichage de toutes les valeurs dans map nécessite une boucle sur les clés.
- La valeur de get est associée à la clé.
  - Iterable .keySet(), est similaire à .words() ou .lines() ou .data()

```
public void printWords(){
    for(String s : myMap.keySet()){
      System.out.println(myMaps.get(s)+"\t"+s);
}
```

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Accéder à toutes les valeurs de map

- Affichage de toutes les valeurs dans des ArrayLists parallèles, est utilisée pour la boucle.
- Un index est utilisé pour word et freq.
- Affichage de toutes les valeurs dans map nécessite une boucle sur les clés.
- La valeur de get est associée à la clé.
  - Iterable .keySet(), est similaire à .words() ou .lines() ou .data()

```
public void printWords(){
    for(String s : myMap.keySet()){
      System.out.println(myMaps.get(s)+"\t"+s);
}
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Maps sont très efficaces !

- Lorsque les fichiers sont volumineux, l'efficacité est primordiale.

	Total Words	Different Words	Time ArrayList	Time HashMap
Julius Caesar	20,869	4,443	0.25	0.03
Confucius	34,582	6,558	0.4	0.05
Scarlet Letter	85,754	13,543	1.3	0.1
King James Bible	823,135	32,675	20.8	0.48

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Maps sont très efficaces !

- Lorsque les fichiers sont volumineux, l'efficacité est primordiale.
- Rechercher dans map est indépendant du nombre de clés ! ArrayList nécessite d'examiner tous les éléments.

	Total Words	Different Words	Time ArrayList	Time HashMap
Julius Caesar	20,869	4,443	0.25	0.03
Confucius	34,582	6,558	0.4	0.05
Scarlet Letter	85,754	13,543	1.3	0.1
King James Bible	823,135	32,675	20.8	0.48

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Live coding !

Maps sont très efficaces !

- peuvent utiliser une seule clé et plusieurs valeurs.
- Nous pouvons utiliser un array list pour ces valeurs.
- Exemple

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE