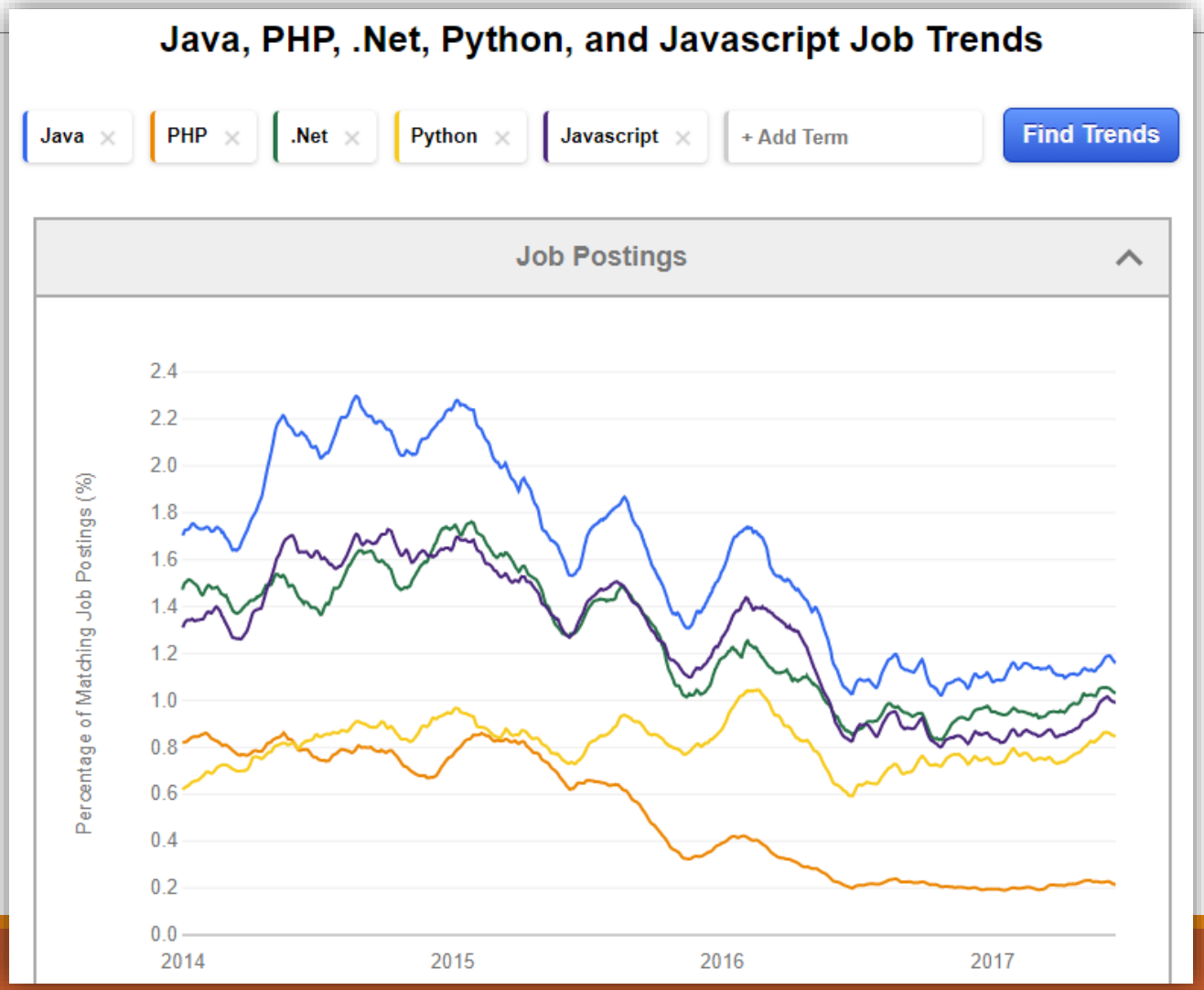
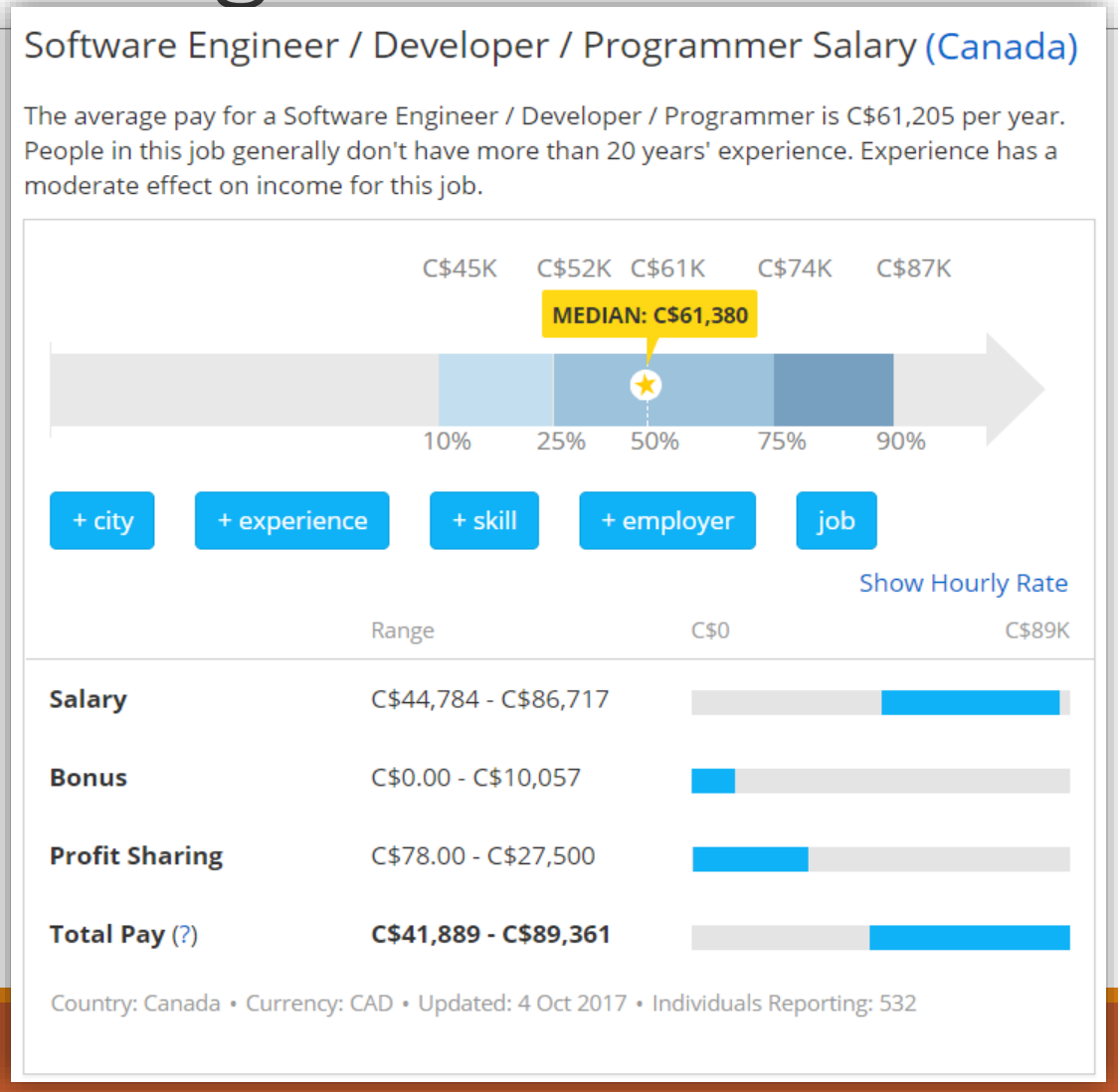
Motivation


Java, PHP, .Net, Python, and Javascript Job Trends
Java
PHP
.Net
Python
Javascript
+ Add Term
Find Trends
Job Postings
Percentage of Matching Job Postings (%)
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2014
2015
2016
2017

Motivation: Programmeur



Software Engineer / Developer / Programmer Salary (Canada)

The average pay for a Software Engineer / Developer / Programmer is C$61,205 per year. People in this job generally don't have more than 20 years' experience. Experience has a moderate effect on income for this job.

C$45K C$52K C$61K C$74K C$87K

MEDIAN: C$61,380

10% 25% 50% 75% 90%

+ city + experience + skill + employer job

Show Hourly Rate

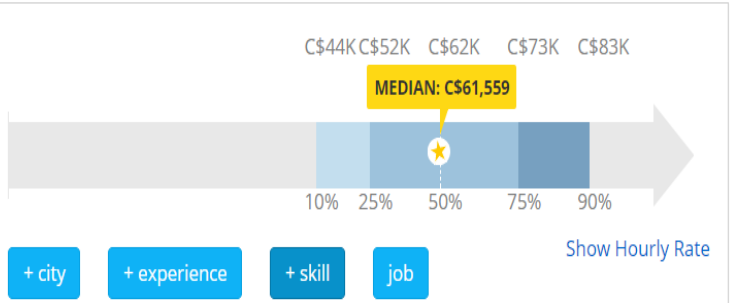
	Range	C$0 C$89K
Salary	C$44,784 - C$86,717	
Bonus	C$0.00 - C$10,057	
Profit Sharing	C$78.00 - C$27,500	
Total Pay (?)	**C$41,889 - C$89,361**	

Country: Canada • Currency: CAD • Updated: 4 Oct 2017 • Individuals Reporting: 532

Motivation: .NET / Java / PHP

.NET Software Developer / Programmer Salary (Canada)

A . NET Software Developer / Programmer earns an average salary of C$60,983 per year. Pay for this job rises steadily for more experienced workers, but goes down noticeably for the few employees with more than 20 years' experience. Most people with this job move on to other positions after 20 years in this field.

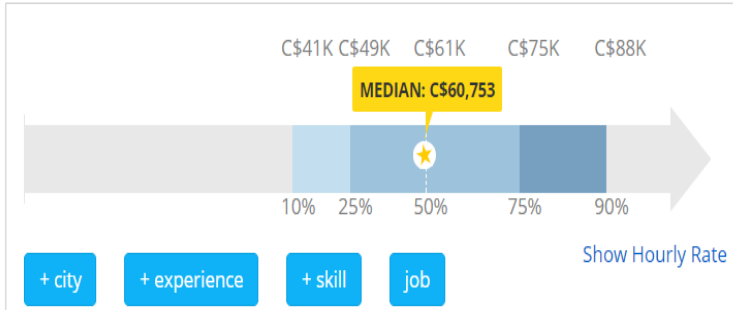


	Range	C$0	C$85K
Salary	C$43,892 - C$82,876		
Bonus	C$0.00 - C$6,937		
Profit Sharing	C$100.67 - C$6,953		
Total Pay (?)	**C$43,607 - C$85,675**		

Country: Canada • Currency: CAD • Updated: 11 Oct 2017 • Individuals Reporting: 447

Java Software Developer / Programmer Salary (Canada)

A Java Software Developer / Programmer earns an average salary of C$61,034 per year. Most people with this job move on to other positions after 20 years in this field. The highest paying skills associated with this job are . NET, C# Programming Language, and Oracle.

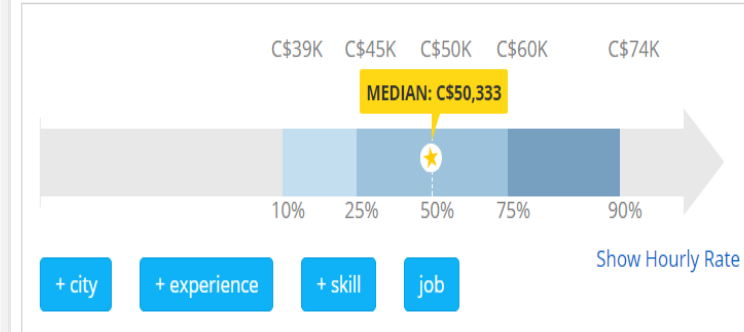


	Range	C$0	C$91K
Salary	C$40,689 - C$88,153		
Bonus	C$972 - C$7,946		
Profit Sharing	C$2,550		
Total Pay (?)	**C$39,543 - C$91,117**		

Country: Canada • Currency: CAD • Updated: 4 Oct 2017 • Individuals Reporting: 102

PHP Web Developer Salary (Canada)

A PHP Web Developer earns an average salary of C$50,207 per year. Experience strongly influences pay for this job. Most people move on to other jobs if they have more than 10 years' experience in this field.



	Range	C$0	C$78K
Salary	C$39,453 - C$74,497		
Bonus	C$0.00 - C$4,932		
Profit Sharing	C$25,000		
Total Pay (?)	**C$39,324 - C$78,192**		

Country: Canada • Currency: CAD • Updated: 6 Oct 2017 • Individuals Reporting: 66

01) Organisation de données en Java

420-109-GG

LOTFI DERBALI

L'écosystème Java

Java existe maintenant depuis plus de vingt ans. Il a été créé en 1995 par James Gosling et Patrick Naughton de la société Sun Microsystems. Cette société a été rachetée par Oracle en 2009. Java appartient donc maintenant à Oracle.

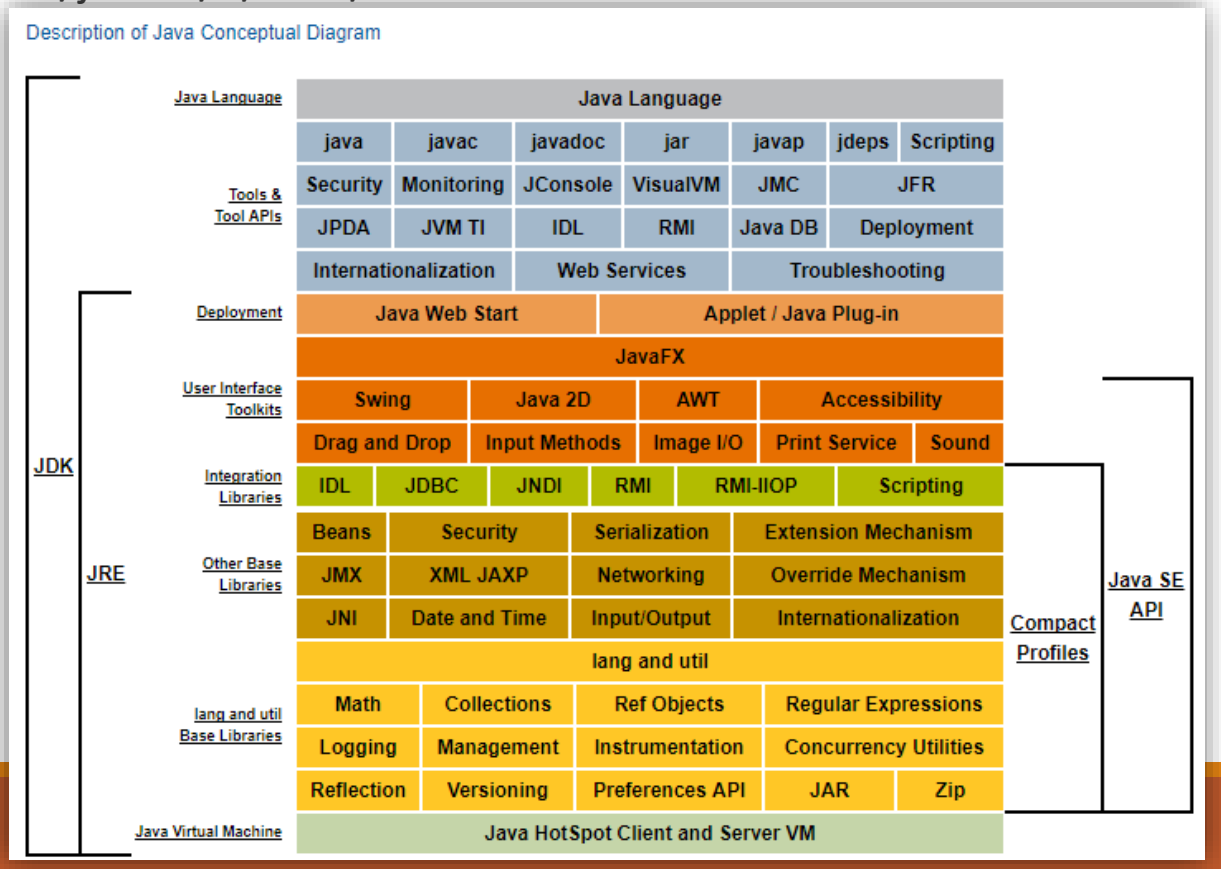
Qu'est-ce que Java ?

- C'est tout d'abord un langage orienté objet
- C'est ensuite une plateforme de développement. Elle est composée d'un ensemble de librairies formant le JDK (*Java SE Developement Kit*). Il y a plusieurs plateformes:
  - La plateforme de base est nommée Java SE (*Java Standard Edition*). Elle est constituée de l'ensemble des librairies du JDK et de certaines spécifications. Elle répond à la plupart des besoins.
  - La seconde plateforme est la plateforme Java EE. Son objectif est de permettre la création d'applications distribuées et notamment d'applications web. Cette plateforme s'appuie sur la plateforme Java SE, mais aussi sur des logiciels tiers, les serveurs d'applications.
- C'est, pour finir, un environnement d'exécution (JRE - *Java Runtime Environment*). Il permet d'exécuter des programmes au travers de la machine virtuelle Java (JVM - *Java Virtual Machine*).

Les librairies disponibles

Pour connaître les librairies mises à disposition par Java, la meilleure manière est de lire la documentation.

- http://docs.oracle.com/javase/8/docs/index.html



Les librairies disponibles

Un survol rapide permet de visualiser certaines API essentielles :

- **lang and util** : les packages java.lang et java.util mettent à disposition des fonctionnalités fondamentales nécessaires dans toutes les applications.
- **Collections** : le framework de collections met à disposition un large panel de classes et d'interfaces pour manipuler des ensembles d'objets.
- **Reflection** : l'API de Reflection présente dans le package java.lang.reflect propose un ensemble de classes et d'interfaces pour découvrir et manipuler les métadonnées des classes d'un programme. Un grand nombre de frameworks se basent sur cette API car elle permet notamment d'exploiter les annotations.
- **Date and Time** : les packages java.time et java.time.* permettent de manipuler efficacement les dates. Ce package est apparu avec le JDK 8 et propose une refonte globale de la gestion des dates.
- **Input/Output** : les packages java.io et java.nio permettent de gérer les flux d'entrées et de sorties.
- **JDBC** : l'API JDBC (Java DataBase Connectivity) permet d'uniformiser l'accès aux bases de données. Cette API fait partie de la plateforme Java SE. Cependant, c'est une API incontournable dans le développement d'une application web.
- **JNDI** : l'API JNDI (Java Naming and Directory Interface) permet d'accéder à des ressources nommées (système de fichiers, annuaire LDAP, objets nommés…).

Les types primitifs

Ce sont des types de donnée non objet qu'on retrouve dans de nombreux autres langages de programmation.

- Boolean
- Char
- byte : (1 octet) entiers compris entre -128 et +127 (-27 et 27-1)
- short : (2 octets) entiers compris entre -32768 et +32767 (-215 et 215-1)
- int : (4 octets) entiers compris entre -2147483648 et +2147483647 (-231 et 231-1)
- long : (8 octets) entiers compris entre -9223372036854775808 et +9223372036854775807 (-2^{63} et 2^{63}-1)
- float : (4 octets) dans ce cas max vaut 255 : ensemble des nombres [-3.40282347E38 .. -1.40239846E-45], 0, [1.40239846E-45 .. 3.40282347E38]
- double : (8 octets) dans ce cas max vaut 2047 : ensemble des nombres [-1.79769313486231570E308 .. -4.94065645841246544E-324], 0, [4.94065645841246544E-324 .. 1.79769313486231570E308]

Classes de types primitifs (Wrappers)

Classes qui encapsulent des types primitifs

- Byte, Short, Integer et Long
- Float et Double
- Character
- Boolean

Possibilités de conversion

- Les classes wrappers proposent également des méthodes de classe, parmi lesquelles on retrouve deux méthodes qui permettent de convertir une chaine de caractères vers un type primitif.

- Ces méthodes sont parseXxx et valueOf qui prennent toutes deux un objet String en paramètre et renvoient un type primitif correspondant Xxx pour parseXxx et un objet wrapper pour valueOf.

```
String s = "456";

int i = Integer.parseInt (s);   // convertit la chaine s en int
Double d = Double.valueOf (s);  // convertit la chaine s en Double
```

Pourquoi une structure de données?

Structure de données sont des conteneurs de données:

- Une organisation des informations
- Pour faciliter le traitement efficace (tableau, liste, …)
- Pour mieux regrouper les informations pour le même concept (classe, …)

Exemple:

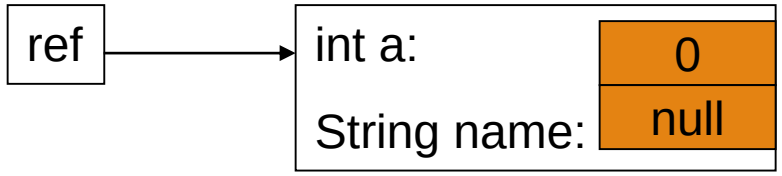
- Tableau: pour organiser une séquence de données
  - *En Java:* int [ ] a = new int [10];

Pourquoi une structure de données?

Autre Exemple:

- Classe: pour regrouper les attributs du même concept
- *En Java:* public class C {

```
        int a;
        String name;
}
C ref = new C();
```



Structures de données en Java

Deux grandes catégories :

- ***Collection***, ou groupe d'objets.
  - Type Set, ne pouvant contenir 2 fois le même élément.
  - Type List, éléments indicés par des entiers positifs.

- ***Map***, table d'association de couples clé-valeur

Différents types de conteneurs, selon l'interface et la structure de donnée.

		Implantations			
		Table de hachage	Tableau à taille variable	Arbre équilibré	Liste chaînée
Interfaces	Set	HashSet		TreeSet	
	List		ArrayList		LinkedList
	Map	HashMap		TreeMap	

Structures de données en Java

Toutes les structures de données de Java sont importables à partir du package ***java.util***

- BitSet
  - La classe BitSet représente de jeux de bits, utile pour représenter un groupe de flags booléens.
- Vector
  - C'est sûrement la structure de données que vous utiliserez le plus souvent. Un Vector est simplement une matrice « dynamique », c'est à dire que l'on peut lui ajouter ou lui enlever des éléments après son initialisation.
- Stack
  - Stack fonctionne avec le principe « premier entré, dernier sorti » (FIFO).
- Dictionary
  - Une structure de données Dictionary se comporte comme une structure Vector, exception faite de l'accès aux données qui ne se fait pas par un indice mais par une "clé".
- Hashtable
  - Hashtable est, comme Vector, une des structures de données les plus utilisées. C'est une classe abstraite dérivée de la classe Dictionary. En français, table de hachage.
- L'interface Enumeration (qui n'est pas une réelle structure de données)
  - Comme l'indique le titre du paragraphe, enumeration n'est pas une classe instanciable main une interface qui définit seulement 2 méthodes:
    - public abstract boolean hasMoreElements();
    - public abstract Object nextElement();

Les Collections

Différents méthodes:

- boolean add(Object o)
  - ajoute l'élément spécifié à la collection. renvoie true si la collection a été modifiée par l'opération (un Set ne peut contenir 2 fois le même élément).
- boolean contains(Object o)
  - teste si la collection contient o
- boolean equals(Object o)
  - teste l'égalité de contenu de la collection avec o.
- Iterator iterator()
  - renvoie un itérateur sur les éléments de la collection.
- boolean remove(Object o)
  - enlève une instance de o de la collection.
- int size()
  - renvoie le nombre d'éléments de la collection.
- Object[] toArray()
  - renvoie un tableau contenant tous les éléments de la collection.

Map

Différents méthodes:

- boolean containsKey(Object key)
  - teste si la table contient une entrée avec la clé spécifiée.
- boolean containsValue(Object value)
  - teste si la table contient une entrée avec la valeur spécifiée.
- boolean equals(Object o)
  - teste l'égalité de contenu de la table avec o.
- Object get(Object key)
  - renvoie la valeur de la table correspondant à la clé key.
- Object put(Object key, Object value)
  - associe la valeur value à la clé key dans la table. Si une valeur était déjà associée, la nouvelle remplace l'ancienne et une référence vers la nouvelle est renvoyée, sinon null est renvoyé.
- Object remove(Object key)
  - enlève l'entrée associée à key de la table. Renvoie une référence sur la valeur retirée ou null si elle n'est pas présente.
- int size()
  - renvoie le nombre d'entrées (paires clé-valeur) de la table.

Classe ArrayList

ArrayList est, grossièrement parlant, un tableau à longueur variable de références à des objets.

On dispose de 3 constructeurs :

- ArrayList()
  - créé une liste de taille initiale 10 références.
- ArrayList(int size)
  - créé une liste de taille initiale size références.
- ArrayList(Collection c)
  - créé une liste avec les éléments de c.
  - la capacité initiale de la liste est de 110% celle de c.

Classe ArrayList

Différents méthodes:

- void add(int index, Object element)
  - l'objet spécifié par élément est ajouté à l'endroit spécifié de la liste.
- boolean contains(Object element)
  - renvoie true si l'élément est contenu dans la liste et false sinon.
- Object get(int index)
  - renvoie l'élément situé à la position spécifiée de la liste.
- final int indexOf(Object element)
  - renvoie l'indice de la 1ère occurrence de l'élément. S'il n'y est pas, -1 est renvoyé.
- Object remove(Object element)
  - enlève la première occurrence de l'élément trouvée dans la liste. Renvoie une référence sur l'élément enlevé.
- Object set(int index, Object element)
  - remplace l'élément à la position spécifiée par l'élément.

Classe LinkedList

Classe LinkedList, de structure sous-jacente une liste doublement chaînée.

- Ajout/suppression en début de liste en temps constant (O(1)).
- Insertion/suppression d'un élément juste après un élément donné (par exemple par un itérateur) en temps constant (O(1)).
- Accès à l'élément i en O(i).

Méthodes supplémentaires de LinkedList

- void addFirst(Object o)
  - insère l'élément spécifié au début de la liste.
- void addLast(Object o)
  - ajoute l'élémentspécifié à la fin de la liste.
- Object getFirst()
  - renvoie le 1er élément de la liste.
- Object getLast()
  - renvoie le dernier élément de la liste.
- Object removeFirst()
  - enlève et renvoie le 1er élément de la liste.
- Object removeLast()
  - enlève et renvoie le dernier l'élément de la liste.

Classe Stack

Stack implante une pile (file LIFO, Last In/First Out) standard.

Stack est une sous classe de Vector.

- Elle hérite donc de toutes les méthodes de Vector, et en définit certaines qui lui sont propres.

Méthodes propres de Stack:

- boolean empty()
  - renvoie true si la pile est vide et false sinon.
- Object peek()
  - renvoie l'élément du dessus de la pile, mais ne l'enlève pas.
- Object pop()
  - renvoie l'élément du dessus de la pile, en l'enlèvant.
- Object push(Object element)
  - pousse élément sur la pile. Élément est également renvoyé.
- int search(Object element)
  - cherche élément dans la pile. S'il est trouvé, son offset par rapport au dessus de la pile est renvoyé. Sinon, -1 est renvoyé.

Classe HashMap

HashMap est la plus utilisée des Map en pratique.

Table de hachage

- une représentation d'une clé est utilisée pour déterminer une valeur autant que possible unique, nommée code de hachage

Le code hachage est alors utilisé comme indice auquel les données associées à la clé sont stockées.

- Pour utiliser une table de hachage :
  1. On fournit un objet utilisé comme clé et des données que l'on souhaite voir liées à cette clé.
  2. La clé est hachée (calculée).
  3. Le code de hachage résultant est utilisé comme indice auquel les données sont stockées dans la table.

Classe HashMap

Méthodes de HashMap:

- boolean containsKey(Object key)
  - Renvoie true s'il existe une clé égale à key.
- boolean containsValue(Object value)
  - Renvoie true s'il existe une valeur égale à value.
- Object get(Object key)
  - Renvoie une référence sur l'objet contenant la valeur associée à la clé key ou null.
- Object put(Object key, Object value)
  - Insère une clé et sa valeur dans la table.
  - Renvoie null si key n'est pas déja dedans, ou la valeur précédente associée à key sinon.
- Object remove(Object key)
  - Enlève la clé key et sa valeur.
  - Renvoie la valeur associée à key ou null.

Transitions entre conteneurs

Dans Collection:

- toArray() renvoie un tableau contenant tous les objets de la collection.

Dans Map:

- values() renvoie une Collection des valeurs de la table
- keySet() renvoie un Set des clés de la table

Dans la classe utilitaire Arrays

- Classe de manipulation de tableaux.
- Méthode *static List asList(Object[] a)* renvoie une vue de type List de a
- toArray() renvoie un tableau contenant tous les objets de la collection

Itération de conteneurs

Par le biais de l'interface Iterator.

Iterator définit des méthodes par lesquelles on peut énumérer des éléments d'une collection.

Iterator spécifie 3 méthodes :

- boolean hasNext()
  - renvoie true s'il y a encore des éléments dans la collection.
- Object next()
  - renvoie une référence sur l'instance suivante de la collection.
- remove()
  - enlève l'élément renvoyé dernièrement par l'itérateur.

```
static void filter(Collection c) {
  for (Iterator i = c.iterator(); i.hasNext();)
      if (!cond(i.next()))
          i.remove();
}
```