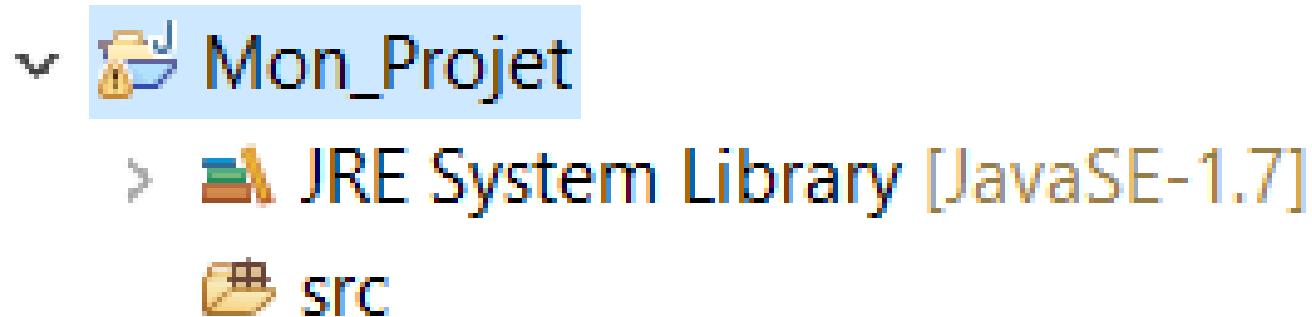


JAVA

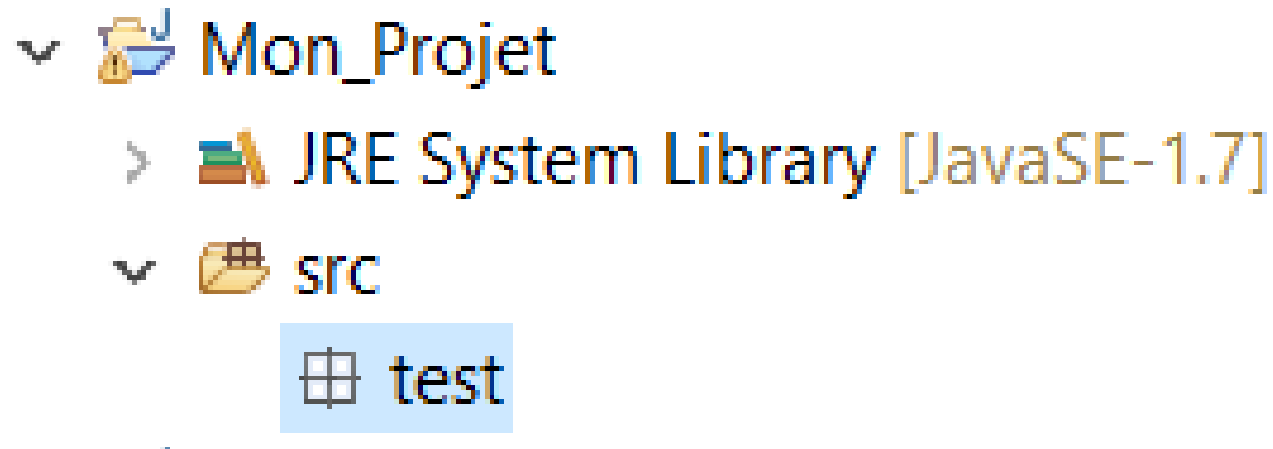
ARCHITECTURE

- Le dossier **src** contiendra tout notre code source et dans lequel nous passerons 99%de notre temps
- Le dossier **JRE** contient tout le JDK c'est-à-dire toutes les packages dépendance de notre application.



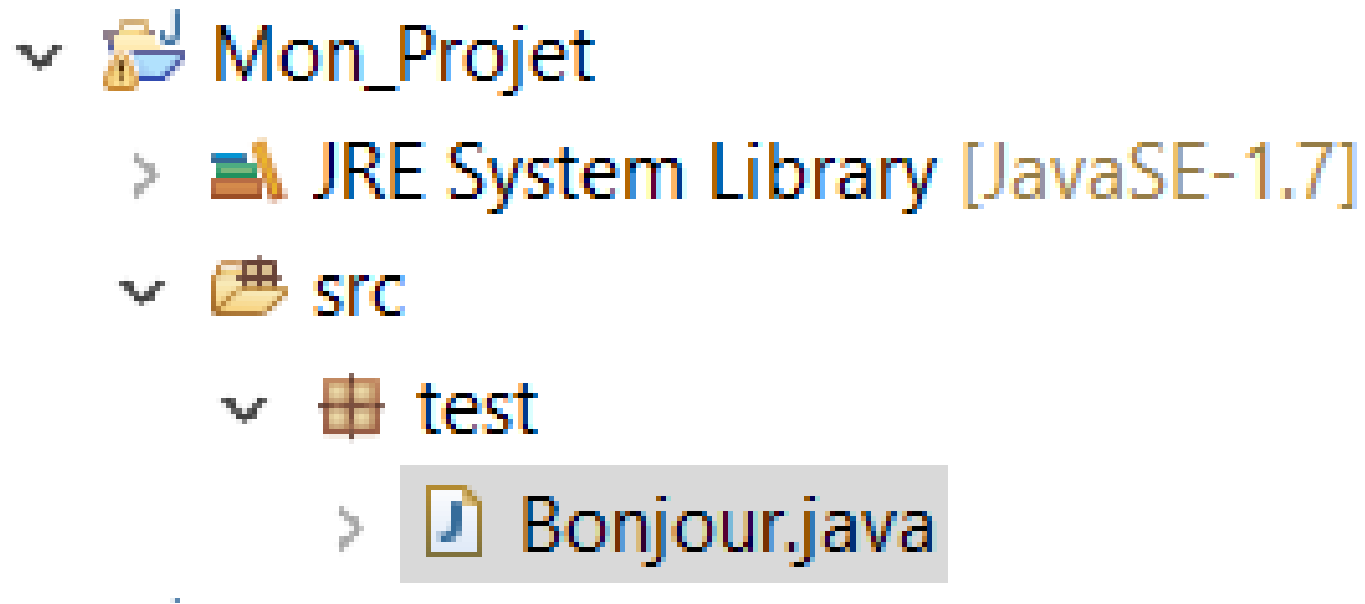
PREMIÈRE APPLICATION

- Maintenant que nous avons tout installer, nous allons créer notre première application web.
- Commençons par le package: Un package permet de regrouper les classes en ensemble pour faciliter la modularité.



CLASSE BONJOUR

Créons une classe Bonjour



CLASSE BONJOUR

- La méthode main constitue la partie principale du programme, permettant l'exécution d'une application Java.
- public indique que la méthode peut être appelée par n'importe quelle classe. Lorsque la méthode main est trouvée dans une classe, elle devient la méthode à partir de laquelle démarre automatiquement le programme.

```
1 package test;  
2  
3 public class Bonjour {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub  
7     }  
8 }  
9  
10 }
```

SYSTEM.OUT.PRINTLN()

- **System** : C'est le nom de la classe standard qui contient les objets qui encapsule la norme I/O dispositifs de votre système.
- **out** : l'objet out représente le flux de sortie
- **println** : Le println() est méthode de out objet prend la chaîne de texte comme argument et l'affiche sur l'écran du moniteur .

```
package test;

public class Bonjour {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello");
    }

}
```

LES VARIABLES

Concernant le nom de nos variables, nous avons une grande liberté dans le nommage de celles-ci mais il y a quand même quelques règles à respecter :

- Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre ;
- Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores mais pas de caractères spéciaux ;
- Le nom d'une variable ne doit pas contenir d'espace.
- Le nom ne doit pas être un mot réservé du langage.

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    String prenom;  
    int age;  
    prenom = "Zak";  
    age=30;  
    System.out.println("Bonjour "+prenom);  
    System.out.println("Vous avez "+age+" ans");  
}
```

```
Bonjour Zack  
Vous avez 30 ans
```

LES TYPES

- **boolean** : un booléen (d'ordre 2 en réalité) qui ne pourra prendre que les valeurs **true** ou **false**.
- **byte** : un entier relatif très court (entre -128 et 127).
- **short** : un entier relatif court (entre -32 768 et 32 767).
- **int** : un entier relatif (entre -2 147 483 648 et 2 147 483 647).
- **long** : un entier relatif long (entre -9 223 372 036 854 776 000 et 9 223 372 036 854 776 000).
- **float** : un nombre décimal (entre $-3,4 \cdot 10^{38}$ et $3,4 \cdot 10^{38}$).
- **double** : un nombre décimal à double précision (entre $-1,7 \cdot 10^{308}$ et $1,7 \cdot 10^{308}$).
- **char** : un caractère (entre '\u0000' et '\uffff').
- **String** : une chaîne de caractère. NB : ce n'est pas un type primitif mais une classe.

CONVERSION DE TYPE DE DONNÉES (CASTING)

Casting : Le cast est le fait de forcer le compilateur à considérer une variable comme étant d'un type qui n'est pas le type déclaré ou le type réel de la variable. Il faut savoir que Java est un langage fortement typé.

Il y a 2 types de casting ;

CONVERSION IMPLICITE

une conversion implicite consiste en une modification du type de donnée effectuée automatiquement par le compilateur. Cela signifie que lorsque l'on va stocker un type de donnée dans une variable déclarée avec un autre type, le compilateur ne retournera pas d'erreur mais effectuera une conversion implicite de la donnée avant de l'affecter à la variable.

(converting a smaller type to a larger type size)

byte -> short -> char -> int -> long -> float -> double

```
public static void main(String[] args) {  
    int myInt = 9;  
    double myDouble = myInt; // Automatic casting: int to double  
  
    System.out.println(myInt);    // Outputs 9  
    System.out.println(myDouble); // Outputs 9.0  
}
```

CONVERSION EXPLICITE

une conversion explicite (appelée aussi opération de cast) consiste en une modification du type de donnée forcée. Cela signifie que l'on utilise un opérateur dit de cast pour spécifier la conversion. L'opérateur de cast est tout simplement le type de donnée, dans lequel on désire convertir une variable, entre des parenthèses précédant la variable.

(converting a larger type to a smaller size type)

```
public static void main(String[] args) {  
    double myDouble = 9.78;  
    int myInt = (int) myDouble; // Manual casting: double to int  
  
    System.out.println(myDouble); // Outputs 9.78  
    System.out.println(myInt);    // Outputs 9  
}
```

SCANNER

```
package test;

import java.util.Scanner;

public class Bonjour {

    public static void main(String[] args) {
        Scanner clavier = new Scanner(System.in);
        System.out.println("Donnez votre prénom : ");
        String prenom=clavier.next();
    }

}
```

SCANNER

Pour lire une chaîne de caractère (String) et s'arrêter au premier espace	clavier.next()
Pour lire une ligne entière	clavier.nextLine()
Pour lire un entier (int)	clavier.nextInt()
Pour lire un décimal (double)	clavier.nextDouble()

EXEMPLE

```
package test;

import java.util.Scanner;

public class Bonjour {

    public static void main(String[] args) {
        Scanner clavier = new Scanner(System.in);

        String prenom;
        int age;
        System.out.println("Entrez votre prénom : ");
        prenom = clavier.next();
        System.out.println("Entrez votre age : ");
        age = clavier.nextInt();

        System.out.println("Bonjour " + prenom + " votre age est " + age + "
ans");

        clavier.close();
    }
}
```

```
Entrez votre prénom :
Jerome
Entrez votre age :
42
Bonjour Jerome votre age est 42 ans
```

LES CONDITIONS

```
Scanner clavier = new Scanner(System.in);
```

```
int nb;
```

```
System.out.println("Entrer un nombre");
```

```
nb = clavier.nextInt();
```

```
if (nb > 0) {
```

```
    System.out.println("Positif");
```

```
} else if (nb < 0) {
```

```
    System.out.println("Négatif");
```

```
} else {
```

```
    System.out.println("Nul");
```

```
}
```

```
clavier.close();
```

LES BOUCLES (FOR)

```
public class Bonjour {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.print(i);  
        }  
    }  
}
```

12345678910

LES BOUCLES (WHILE)

```
public class Bonjour {  
    public static void main(String[] args) {  
        int n = 1;  
        while (n <= 5) {  
            System.out.print(n);  
            n++;  
        }  
    }  
}
```

12345

LES BOUCLES (DO...WHILE)

```
public class Bonjour {  
    public static void main(String[] args) {  
        int n = 1;  
        do {  
            System.out.print(n);  
            n++;  
        } while (n <= 5);  
    }  
}
```

12345

LES TABLEAUX

Exemple d'un tableau nommé t1, avec des 4 entiers :

```
int[] t1 = { 10, 15, 45, 89 };
```

Pour afficher 15 ;

```
System.out.println(t1[1]);
```

Si l'on veut modifier une valeur dans le tableau

```
t1[0]=111;
```

Contrairement à JavaScript, on ne peut pas rajouter une valeur dans le tableau de cette façon ;

```
t1[4] = 100; //erreur
```

➔ *Un tableau en Java est dimensionné une fois pour toute.*

On peut commencer par déclarer un tableau en indiquant uniquement sa taille. Les valeurs peuvent être précisées dans un 2nd temps

```
int[] t1 = new int[3];  
t1[0]=111;  
t1[1]=55;  
t1[2]=88;
```

LES TABLEAUX : FOR

```
public static void main(String[] args) {  
    int[] t1 = { 10, 15, 45, 89 };  
    for (int i = 0; i < t1.length; i++) {  
        System.out.println(t1[i]);  
    }  
  
    for (int val : t1) {  
        System.out.print(val + " ");  
    }  
}
```

```
10  
15  
45  
89  
10 15 45 89
```

LES CHAÎNES DE CARACTÈRES

charAt()

Description Returns the character at the specified index (position)

Return Type char

```
String prenom = "Zack";  
char c = prenom.charAt(1);  
System.out.println(c);
```

a

LES CHAÎNES DE CARACTÈRES

equals()

Description Compares two strings. Returns true if the strings are equal, and false if not

Return Type boolean

```
String a = "zack", b = "Zack";  
if (a.equals(b)) {  
    System.out.println("identiques");  
} else {  
    System.out.println("différents");  
}
```

différents

LES CHAÎNES DE CARACTÈRES

equalsIgnoreCase()

Description Compares two strings, ignoring case considerations

Return Type boolean

```
String a = "zack", b = "Zack";  
if (a.equalsIgnoreCase(b)) {  
    System.out.println("identiques");  
} else {  
    System.out.println("différents");  
}
```

identiques

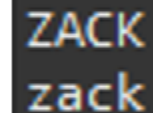
LES CHAÎNES DE CARACTÈRES

toUpperCase() – toLowerCase()

Description Converts a string to upper/lower case letters

Return Type String

```
String a = "Zack";  
String aUpper = a.toUpperCase();  
String aLower = a.toLowerCase();  
System.out.println(aUpper);  
System.out.println(aLower);
```



ZACK
zack

LES CHAÎNES DE CARACTÈRES

contains()

Description Checks whether a string contains a sequence of characters

Return Type boolean

```
String chaine = "Bonjour Zack Hello";  
String test = "Zack";  
if (chaine.contains(test)) {  
    System.out.println("présent");  
} else {  
    System.out.println("non");  
}
```

présent

LES CHAÎNES DE CARACTÈRES

On peut assembler la méthode `toLowerCase()` avec `contains()` pour éviter des problèmes dûs à la casse.

```
String chaine = "Bonjour Zack Hello";  
String test = "zack";  
if (chaine.toLowerCase().contains(test)) {  
    System.out.println("présent");  
} else {  
    System.out.println("non");  
}
```

présent

LES CHAÎNES DE CARACTÈRES

startsWith()

Description Checks whether a string starts with specified characters

Return Type boolean

```
String chaine = "Bonjour Zack Hello";  
if (chaine.toLowerCase().startsWith("bon")) {  
    System.out.println("ok");  
} else {  
    System.out.println("non");  
}
```

ok

- ① La méthode `toLowerCase()` est ajoutée à `startsWith()` pour éviter les problèmes de casse.

LES CHAÎNES DE CARACTÈRES

endsWith()

Description Checks whether a string ends with specified characters

Return Type boolean

```
String chaine = "Bonjour Zack Hello";  
if (chaine.endsWith("lo")) {  
    System.out.println("ok");  
} else {  
    System.out.println("non");  
}
```

ok

LES CHÂÎNES DE CARACTÈRES

length()

Description Returns the length of a specified string

Return Type int

```
String chaine = "Bonjour Zack Hello";  
System.out.println(chaine.length());
```

18

① Affiche la longueur d'une chaîne, en tenant compte des espaces

LES CHAÎNES DE CARACTÈRES

replace()

Description Searches a string for a specified value, and returns a new string where the specified values are replaced

Return Type String

```
String chaine = "Bonjour Zack Hello";  
String chaine2 = chaine.replace("Zack",  
"Jérôme");  
System.out.println(chaine2);
```

Bonjour Jérôme Hello

LES CHAÎNES DE CARACTÈRES

replaceFirst()

Description Replaces the first occurrence of a substring that matches the given regular expression with the given replacement

Return Type String

```
String chaine = "Bonjour Zack Hello Zack";  
String chaine2 = chaine.replaceFirst("Zack", "Jérôme");  
System.out.println(chaine2);
```

Bonjour Jérôme Hello Zack

LES CHÂÎNES DE CARACTÈRES

replaceAll()

Description Replaces the first occurrence of a substring that matches the given regular expression with the given replacement

Return Type String

```
String chaine = "Bonjour Zack Hello Zack";  
String chaine2 = chaine.replaceAll("Zack", "Jérôme");  
System.out.println(chaine2);
```

Bonjour Jérôme Hello Jérôme

LES CHAÎNES DE CARACTÈRES

substring()

Description Extracts the characters from a string, beginning at a specified start position, and through the specified number of characters

Return Type String

```
String chaine = "Bonjour Zack Hello";  
System.out.println(chaine.substring(3, 7));  
System.out.println(chaine.substring(13));
```

```
jour  
Hello
```

① On s'arrête avant l'index 7

LES CHÂÎNES DE CARACTÈRES

indexOf()

Description Returns the position of the first found occurrence of specified characters in a string

Return Type int

```
String chaine = "Bonjour Zack Hello";  
System.out.println(chaine.indexOf("Zack"));
```

8

```
String chaine = "Bonjour Zack Hello";  
System.out.println(chaine.indexOf("o"));
```

1

❶ Il retourne automatiquement le premier occurrent

Mais si on a plusieurs fois Zack, il faut utiliser la syntaxe

```
String chaine = "Bonjour Zack Hello Zack";  
System.out.println(chaine.indexOf("Zack", 9));
```

19

Ou

```
System.out.println(chaine.indexOf("Zack", chaine.indexOf("Zack") + 1));
```

LES CHÂÎNES DE CARACTÈRES

lastIndexOf()

Description Returns the position of the last found occurrence of specified characters in a string

Return Type int

```
String chaine = "Bonjour Zack Hello Zack";  
System.out.println(chaine.lastIndexOf("Zack"));
```

19

LES CHAÎNES DE CARACTÈRES

toCharArray()

Description Converts this string to a new character array

Return Type char[]

```
String chaine = "Bonjour Zack Hello Zack";  
char[] c = chaine.toCharArray();  
System.out.println(c[2]);  
System.out.println(c.length);
```

n
23

Pour transformer un tableau en chaine de caractère

```
char[] c= {'Z','A','C','K'};  
String chaine = new String(c);  
System.out.println(chaine);
```

ZACK

LES CHAÎNES DE CARACTÈRES

repeat()

Repeat permet de répéter une chaîne de caractère n fois → repeat(String str, int times)

```
String str = "Abc";  
System.out.println(str.repeat(3));
```

AbcAbcAbc

LES COLLECTIONS (ARRAYLIST)

La classe **ArrayList** en Java est utilisée pour stocker une collection d'éléments de taille dynamique. Contrairement aux tableaux dont la taille est fixe, un **ArrayList** augmente sa taille automatiquement lorsque de nouveaux éléments lui sont ajoutés.

Pour utiliser un ArrayList, il est nécessaire d'importer java.util.ArrayList

```
import java.util.ArrayList;
```

LES COLLECTIONS (ARRAYLIST)

La classe **ArrayList** en Java est utilisée pour stocker une collection d'éléments de taille dynamique. Contrairement aux tableaux dont la taille est fixe, un **ArrayList** augmente sa taille automatiquement lorsque de nouveaux éléments lui sont ajoutés.

Pour utiliser un ArrayList, il est nécessaire d'importer java.util.ArrayList

```
import java.util.ArrayList;
```

LES COLLECTIONS (ARRAYLIST)

```
package test;  
import java.util.ArrayList;
```

```
La taille la collection : 4  
La valeur de la position 0: 10
```

```
public class Bonjour {  
    public static void main(String[] args) {  
        ArrayList<Integer> col = new ArrayList<>();  
        col.add(10);  
        col.add(20);  
        col.add(30);  
        col.add(40);  
        System.out.println("La taille la collection : " + col.size());  
        System.out.println("La valeur de la position 0: " + col.get(0));  
    }  
}
```


LES COLLECTIONS (ARRAYLIST)

3 méthodes pour afficher toutes les valeurs :

Méthode 1

```
for (int i = 0; i < col.size(); i++) {  
    System.out.print(col.get(i) + " ");  
}
```

10 20 30 40

Méthode 2

```
for (int val : col) {  
    System.out.print(val + " ");  
}
```

10 20 30 40

Méthode 3

```
System.out.print("Afficher la collection " + col);
```

Afficher la collection [10, 20, 30, 40]

LES COLLECTIONS (ARRAYLIST)

`indexOf()`, `lastIndexOf()`

Avec la collection suivante ;

```
col.add(10);  
col.add(20);  
col.add(30);  
col.add(40);  
col.add(20);
```

La position de l'élément 20 est 1

```
System.out.println("La position de l'élément 20 est " + col.indexOf(20));
```

Pour trouver l'index du dernier élément ;

```
System.out.println("La position de l'élément 20 est " + col.lastIndexOf(20));
```

La position de l'élément 20 est 4

LES COLLECTIONS (ARRAYLIST)

isEmpty()

Pour tester si une collection est **vide**,

```
System.out.println("La collection est vide ? " + col.isEmpty());
```

```
La collection est vide ? false
```

contains()

Pour tester si la collection **contient** la valeur 20 ;

```
System.out.println("La collection contient la val 20 ? " + col.contains(20));
```

```
La collection contient la val 20 ? true
```

LES COLLECTIONS (ARRAYLIST)

set()

Pour **modifier** un élément, set()

```
col.set(0, 111);  
System.out.println(col);
```

```
[111, 20, 30, 40, 20]
```

remove()

Pour **supprimer** un élément, on agit sur l'**index** avec un remove()

```
col.remove(2);  
System.out.println(col);
```

```
[10, 20, 40, 20]
```

❗ Attention, cela va décaler tous les index

LES COLLECTIONS (ARRAYLIST)

clear()

Pour vider la collection,

```
col.clear();  
System.out.println(col);
```

```
[]
```

Copier une collection dans une autre ;

```
ArrayList<Integer> col2 = new ArrayList<>(col);  
System.out.println(col2);
```

```
[20, 100, 10, 40, 15]
```

LES COLLECTIONS (ARRAYLIST)

addAll()

Pour copier une collection à la fin d'une autre ;

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(1);  
col.add(2);  
col.add(3);  
  
ArrayList<Integer> col2 = new ArrayList<>();  
col2.add(10);  
col2.add(20);  
  
col.addAll(col2);  
System.out.println(col);
```

```
[1, 2, 3, 10, 20]
```

LES COLLECTIONS (ARRAYLIST)

Insérer une valeur

Dans la collection, on peut insérer une valeur

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(11);  
col.add(22);  
col.add(33);  
  
col.add(1, 44);  
System.out.println(col);
```

Les autres éléments sont décalés.

```
[11, 44, 22, 33]
```

LES COLLECTIONS (ARRAYLIST)

sort()

Avec la collection suivante ;

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(20);  
col.add(100);  
col.add(10);  
col.add(40);  
col.add(15);
```

Pour la trier par **ordre croissant**, sort():

```
Collections.sort(col);  
System.out.println(col);
```

[10, 15, 20, 40, 100]

Nécessaire d'ajouter

```
import java.util.Collections;
```

Pour la trier par **ordre décroissant** :

```
Collections.sort(col, Collections.reverseOrder());  
System.out.println(col);
```

[100, 40, 20, 15, 10]

LES COLLECTIONS (ARRAYLIST)

reverse()

Pour inverser l'ordre de la collection , `reverse()`;

```
Collections.reverse(col);  
System.out.println(col);
```

```
[15, 40, 10, 100, 20]
```

shuffle()

Fonction de mélange

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(11);  
col.add(22);  
col.add(33);
```

```
Collections.shuffle(col);  
System.out.println(col);
```

```
[22, 11, 33]
```

LES COLLECTIONS (ARRAYLIST)

max(), min()

Trouver le max et le min d'une collection

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(11);  
col.add(22);  
col.add(33);  
  
System.out.println("MAX : " + Collections.max(col));  
System.out.println("MIN : " + Collections.min(col));
```

```
MAX : 33  
MIN : 11
```

LES COLLECTIONS (ARRAYLIST)

frequency()

Trouver la fréquence d'un élément

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(10);  
col.add(20);  
col.add(30);  
col.add(40);  
col.add(20);  
  
System.out.println("la fréquence de 20 : " +  
Collections.frequency(col, 20));
```

```
la fréquence de 20 : 2
```

LES COLLECTIONS (ARRAYLIST)

replaceAll()

```
ArrayList<Integer> col = new ArrayList<>();  
col.add(10);  
col.add(20);  
col.add(30);  
col.add(40);  
col.add(10);  
  
Collections.replaceAll(col, 10, 100);  
System.out.println(col);
```

```
[100, 20, 30, 40, 100]
```

LES COLLECTIONS (ARRAYLIST)

swap()

```
ArrayList<String> col = new ArrayList<>();  
col.add("a");  
col.add("b");  
col.add("c");  
  
Collections.swap(col, 0, 2);  
System.out.println(col);
```

[c, b, a]

LES FONCTIONS

```
public class Bonjour {  
    public static void main(String[] args) {  
        Hello();  
    }  
  
    public static void Hello() {  
        System.out.println("Hello");  
    }  
}
```

Hello

Il est nécessaire de déclarer la fonction après la fonction Main et de l'appeler dans la fonction Main.

LES FONCTIONS AVEC PARAMÈTRES

```
public class Bonjour {  
    public static void main(String[] args) {  
        String prenom = "Zak";  
        Hello(prenom);  
    }  
  
    public static void Hello(String test) {  
        System.out.println("Hello " + test);  
    }  
}
```

Hello Zak

LES FONCTIONS : RETURN

```
public class Bonjour {  
    public static void main(String[] args) {  
        int resultat = somme(5, 2);  
        System.out.println("la somme est : " + resultat);  
    }  
  
    public static int somme(int a, int b) {  
        int sm = a + b;  
        return sm;  
    }  
}
```
