Java tutorial

HISTOIRE DE LANGAGE JAVA :

Le langage Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy, cofondateur de Sun Microsystems. Java a été officiellement présenté le 23 mai 1995 au SunWorld. La société Oracle racheta alors la société Sun en 2009, ce qui explique pourquoi ce langage appartient désormais à Oracle. La particularité et l'intérêt de Java réside dans sa portabilité entre les différents systèmes d'exploitations tels que Unix, Windows, ou MacOS. Un programme développé en langage Java, peut ainsi s'exécuter sur toutes les plateformes, grâce à ses frameworks associés visant à garantir cette portabilité.

LES OUTILS NÉCESSAIRES POUR DÉVELOPPER AVEC JAVA :

Pour commencer à programmer avec Java, vous avez besoin de deux choses JDK & IDE

JDK (Java Deverlopment Kit)

IDE (Integrated Development Environment)

JDK est un outil de développe Java, c'est pour les programmeurs Java pour développer des applications. Il se compose de:

- JRE (Java Runtime Environment) est un environnement d'exécution d'applications Java.
- Javac: Un programme pour traduire le code que vous écrivez en bytecode, l'application Java pour l'exécuter traduit bytecode en code machine et exécuté, cela signifie que le bytecode est juste un code intermédiaire.
-
- Archive (jar): est un programme qui compresse les fichiers en un seul fichier se terminant

par un bocal. Souvent utilisé pour les fichiers de classe d'emballage.

- Javadoc: Un outil pour créer des documents en utilisant l'API.
- Et d'autres outils nécessaires pour les développeurs Java.

IDE (Integrated Development Environment) Est un environnement de développement intégré (IDE), essentiellement, il est un programme pour vous d'écrire du code. Ce programme prend en charge de nombreuses fonctionnalités d'automatisation pour les développeurs. Ces suggestions lors de la programmation, le perfectionnement du code, ...

Le premier programme en java

Il est de tradition de commencer tout apprentissage d'un langage par l'écriture d'un petit programme capable d'écrire « Bonjour le monde » sur un écran. Il semble même que ce soit une caractéristique de tout langage de programmation évolué, et de nombreux frameworks. Java tombant dans cette catégorie, il paraît indispensable de sacrifier à cette tradition !

Écrivons un premier programme Java. Cela nous permettra d'une part de clarifier la procédure de compilation et d'exécution, et d'autre part d'examiner la structure générale des programmes Java.

Exemple 1. J'écris Bonjour le monde en Java !

```
public  class Bonjour {
   public  static  void main (String [] arguments) {
    System.out.println("Bonjour le monde") ;
    System.exit(0) ;
  }
}
```

On remarque tout d'abord que le code est contenu dans un bloc : `public class Bonjour { ... }`. Tout comme dans d'autres langages, un code Java se découpe en blocs, délimités par des accolades. Tout code Java doit être déclaré à l'intérieur d'une classe, et la fonction de ce bloc est de délimiter une telle classe. À l'intérieur d'une classe, on peut déclarer trois types de choses : des champs, des méthodes et des blocs.
La première déclaration que l'on trouve dans ce bloc est celle d'une méthode : `public static void main(...)`. Cette méthode main a un statut particulier : c'est elle qui est appelée en premier quand on lance un programme Java.
À l'intérieur de ce bloc se trouvent deux commandes. La première affiche Bonjour le monde sur la console, la deuxième ferme le programme. Nous venons d'écrire notre premier programme Java.

I.Les variables

Une variable est un objet repéré par son nom, pouvant contenir des données, qui pourront être modifiées lors de l'exécution du programme. Les variables en langage Java sont typées, c'est-à-dire que les données contenues dans celles-ci possèdent un type, ainsi elles sont donc stockées à une adresse mémoire et occupent un nombre d'octets dépendant du type de donnée stockée.

1. Créer une variable, et demander à l'utilisateur d'entrer son prénom. La machine répond par un charmant « Bonjour » suivi du prénom

```
package com.company;
import java.util.Scanner; // import the Scanner class
2.  Une société de produit cosmétique souhaitera créer une calculatrice de TVA avec remise. yourName {
    public static void main(String[] args) {
        Scanner firstAt = new Scanner(System.in);
        String userName;

        // Enter username and press Enter
        System.out.println("What's your name ?");
        userName = firstAt.nextLine();

        System.out.println("Hello " + userName);
    }

}
```

```
package com.company;
```

un package est une unité (un fichier) regroupant des classes. Pour créer un tel package, il suffit de commencer le fichier source contenant les classes à regrouper par l'instruction package suivi du nom que l'on désire donner au package. dès lors, toutes les classes contenues dans le fichier feront partie du package...

```
import java.util.Scanner; // import the Scanner class
```

java.util.Scanner est une classe de l'API Java utilisée pour créer un objet Scanner, un
objet extrêmement polyvalent que vous pouvez utiliser pour saisir des caractères alphanumériques à partir de plusieurs entrées
sources et les convertir en données binaires.

2. Une société de produit cosmétique souhaitera créer une calculatrice de TVA avec remise.

```
package com.company;
import java.util.Scanner; // import the Scanner class
public class Main {

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

        double HT=0, TVA=0, TTC=0, R=0, NETC=0;

        System.out.println("Entry Number");
        int n=yourPrix.nextInt();
        for (int i = 0; i < n; i++) {
            System.out.println("Entre Prix");
            double prix=yourPrix.nextDouble();
            HT+=prix;

        }
        if (HT>2000) {
            R = HT*0.1;
            NETC=HT-R;

        }
        TVA=NETC*0.2;
        TTC=NETC+TVA;
        System.out.println("the amount is "+HT+"MAD");
        System.out.println("the discount is"+R+"MAD");
        System.out.println("net commerciel"+NETC+"MAD");
        System.out.println("TVA"+TVA+"MAD");
        System.out.println("TTC"+TTC+"MAD");
    }


}
```

II.Les boucles

Dans la programmation informatique (computer programming), une boucle est régulièrement utilisée et le but est d'exécuter un programme à plusieurs reprises. Java prend en charge 3 types de boucles différentes :

- 1 la boucle **for;**
- 2 La boucle **while;**
- 3. La boucle **do-while.**

1-La boucle for

Syntaxe :

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed}
```

Statement 1: est exécutée (une fois) avant l'exécution du bloc de code.
Statement 2: définit la condition d'exécution du bloc de code.

Statement 3: est exécutée (à chaque fois) après l'exécution du bloc de code.

La boucle while

Syntaxe :

```
while (condition) {
  // code block to be executed}
```

La boucle while est utilisée pour exécuter plusieurs fois un segment de programme, quand une condition est toujours true. La boucle while est généralement utilisée lorsque le nombre d'itérations ne peut être déterminé avant (non fixé).

3-La boucle do-while

Syntaxe :

```
do {
  // code block to be executed
}
while (condition);
```

La boucle do-while est utilisée pour exécuter un segment de programme plusieurs fois. Les caractéristiques de do-while est un bloc de commande qui est toujours exécuté au moins une fois. Après l'itération (iteration), le programme vérifie la condition, si la condition est toujours vraie, le bloc de commande sera encore exécuté.

Maintenant en va voir des exemples dans cette partie.

1-Créer une variable et l'initialiser à 0. Tant que cette variable n'atteint pas 10 : Afficher & Incrémenter de 1

```
package com.company;

class boucle {
    /* début de la fonction main() */
    public static void main(String[] args) {
        /* Déclaration des variables */
        int i;
        /* initialisation des variables */
        i=1;
        /* début de la boucle while : tant que i est inférieur */
        /* ou égal à 10 on l'affiche et on l'incrémente : */
        while(i<=10)
```

```
        {
            System.out.println(i);
            i=i+1;
        } /* fin de la boucle while */
    } /* fin de la fonction main() */
}
```

2. Créer deux variables. Initialiser la première à 1 et la deuxième avec un nombre compris en 2 et 100. Tant que la première variable n'est pas supérieure à 30 : Multiplier la première variable avec la deuxième & Afficher le résultat & Incrémenter la première variables:

```
package com.Atelier.lesboucles;
import java.util.Scanner;
public class Multiplacation {

    public static void main(String[] args) {
        Scanner scanner = new Scanner( System.in );
        int a=1;
        System.out.println("Saisir le nombre");
        Float b=scanner.nextFloat();
        if((b>2) && (b<100))
        {
            while (a < 30) {
                  System.out.println(a*b);
                  a++;
                }

        }
        else {
             System.out.println("entre le nombre compris 2 et 100");
        }
    }
}
```

3. Créer une variable et l'initialiser à 1. Tant que cette variable n'atteint pas 10 : Afficher & Incrémenter de la moitié de sa valeur

```
int i = 1;
        while (i < 10) {
            double j=i/2;
          System.out.println("la moité est "+j);
          i++;
        }
```

4. En allant de 1 à 15 avec un pas de 1, afficher le message On y arrive presque... En allant de 20 à 0 avec un pas de 1, afficher le message 'C'est presque bon...' En allant de 1 à 100 avec un pas de 15, afficher le message 'On tient le bon bout...' En allant de 200 à 0 avec un pas de 12, afficher le message 'Enfin ! ! !'

III.Les fonctions

On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instruction par simple appel de la fonction dans le corps du programme principal. Les fonctions permettent d'exécuter dans plusieurs parties du programme une série d'instructions, cela permet une simplicité du code et donc une taille de programme minimale. D'autre part, une fonction peut faire appel à elle-même, on parle alors de fonction récursive (il ne faut pas oublier de mettre une condition de sortie au risque sinon de ne pas pouvoir arrêter le programme...).

```
package com.company;

public class function {

        // exircice 1
        public boolean bool() {
            return true;
        }

        // return string
        // exercice 2
        public String return_string(String a) {
            return a;
        }

        // contactenate string
        // exercice 3
        public String concatnate_a_b(String a, String b) {
            return a + b;
        }

        // exercice 4
        public String whoBorSm(int a, int b) {
            if (a > b)
                return "Le premier nombre est plus grand";
            else if (a < b)
                return "Le premier nombre est plus petit";
            else
                return "Les deux nombres sont identiques";
        }

        // exeercice 5
        public String contatenate_str_int(int a, String b) {
            return a + b;
        }

        // exercice 6
        public String who_are_you(String nom, String prenom, int age) {
            return "Bonjour " + nom + " " + prenom + ", tu as " + age + " ans.";
        }
        // exercice 7
        public String is_majour(int age, String genre) {
            genre = genre.toLowerCase();
            // if genre does not match homme or femme
            if (!genre.equals("homme") && !genre.equals("femme")) return "NaH";
            // age > 18 ans
            if (age >= 18) {
                return "Vous êtes un " + genre + " et vous êtes majeur";
            }
```

```
            // age > 0
            else if (age > 0) return "Vous êtes un " + genre + " et vous êtes mineru";
                // if age < 0;
            else return "Nah";
        }

        // exercice 8
        public int sum_of_3(int a, int b, int c) {
            return a + b + c;
        }


        public static void main(String[] args) {
            // TODO Auto-generated method stub
            function test = new function();
            System.out.println(test.return_string("hi"));
            System.out.println(test.concatnate_a_b("Hi", "Lamyaa"));
            System.out.println(test. contatenate_str_int(18, "ops test"));
            System.out.println(test.who_are_you("lamyaa", "lamyita", 24));

            System.out.println(test.sum_of_3(18, 20, 22));
            System.out.println(test.whoBorSm(23 ,23));
            System.out.println(test.bool());
            System.out.println(test.is_majour( 23, "femme"));




        }

    }
```

IV.Les collections

Une collection gère un groupe d'un ensemble d'objets d'un type donné ; ou bien c'est un objet qui sert à stocker d'autres objets. Dans les premières versions de Java, les collections étaient représentées par les "Array","Vector","Stack" etc. Puis avec Java 1.2 (Java 2), est apparu le framWork de collections qui tout en gardant les principes de bases, il a apporté des modifications dans la manière avec laquelle ces collections ont été réalisées et hiérarchisées.

Tout en collaborant entre elles, ces collections permettent de réaliser dans des catégories de logiciels des conceptions réutilisables.

ET maintenant on va voir des petites exemples:

```
    package lescollections;

    import java.util.ArrayList;

     import java.util.*;
      public class collection{
      public static void main(String args[]){

    List<String> list=new ArrayList<String>();
```

```
    List<String> listl=new LinkedList<String>();
    Set<String> myhashset = new HashSet<>();
    HashMap<String,String> myhashmap1 = new HashMap<>() , myhashmap2 = new HashMap<>();  //Ecrire une hashmap
    list.add("janvier");
    list.add("février");
    list.add("mars");
    list.add("avril");
    list.add("mai");
    list.add("juin");
    list.add("juillet");
    list.add("aout");
    list.add("septembre");
    list.add("octobre");
    list.add("novembre");
    list.add("décembre");
    list.set(7,new String("août"));          //Modifier le mois aout
    listl.add("Im the first elm");                 //Insérer un élément dans la liste à la première position
    list.remove(2);                           //Supprimer le troisième élément de cette liste.
    list.sort( Comparator.comparing( String::toString ));  //Trier cette liste
    listl.addAll(list);         //Copier cette liste dans une autre
    listl.addAll(list);         //clone la liste dans une autre??????????????????????????????

    myhashset.add("janvier");
    myhashset.add("février");
    myhashset.add("mars");
    myhashset.add("avril");
    myhashset.add("mai");
    myhashset.add("juin");
    myhashset.add("juillet");
    myhashset.add("aout");
    myhashset.add("septembre");
    myhashset.add("octobre");
    myhashset.add("novembre");
    myhashset.add("décembre");

    myhashmap1.put("car","Mercedesbenz");
    myhashmap1.put("motor","Docati");
    myhashmap1.put("chihaja","chi haja mn chihaja");


   for(String mois:list){    //afficher list des mois
        System.out.println(mois);
         System.out.println("HashSet: " + list);   //or
}
    System.out.println(list.get(2));  //afficher la troisième ligne
    System.out.println(list.get(4));  //afficher le 5eme ligne

    System.out.println(list.indexOf("octobre"));  //Rechercher un élément dans cette liste

    System.out.println(listl.isEmpty());          //  Tester cette liste est vide ou non

    for(String mois:myhashset){    //afficher hashset des mois
        System.out.println(mois);
         System.out.println("HashSet: " + myhashset);    //or
    }



    String last = null;
    for(String mois:myhashset){    //afficher last des mois
        last = mois;
    }
    System.out.println(last);

    System.out.println(myhashset.size());       //le nombre d'éléments dans un ensemble de HashSet

    myhashset.clear();                          //Vider cette HashSet
```

```
    System.out.println(myhashset.isEmpty());    //Tester que cette HashSet est vide ou non


    System.out.println("Hashmap : "+myhashmap1);
    System.out.println("Hashmap : "+myhashmap1.size());
    myhashmap2.putAll(myhashmap1);
    System.out.println("Hashmap : "+myhashmap2);

    myhashmap1.remove("car");
    System.out.println("Hashmap : "+myhashmap1);
}
}
```

Dans ce moment on a passer au brief ou plutôt l'application de gestion.

voilà la problèmatique:

Ma tante m'envoie une enveloppe, pour m'aider à financer mes études. Je gère cette somme de la façon suivante :

Les trois quarts de cette somme seront dédiés à l'achat de livres et fournitures. Le reste sera équitablement réparti entre les rubriques :

- Cafés,
- Carte de recharge prépayer,
- Billets de TRAME

Je décide également de gérer cette somme en MAD tout rond. Le reste de l'argent sera destiné à acheter un bouquet de fleur à ma maman pour la prochaine fête des mères. En supposant qu'un café vaut 10 MAD, qu'un numéro de la Carte de recharge prépayer en vaut 10 MAD et qu'un billet de TRAME vaut 8 MAD.

```
 package problématique;

import java.util.Scanner;

public class gestion {

public static void main(String[] args) {
   Scanner input = new Scanner(System.in);
        System.out.println("entrer le prix");
        int prix = input.nextInt();
        int qrt = prix/4;
        int rest1=prix%4;
        int achat = qrt*3;
        int total= qrt/3;
        int rest2= qrt%3;
        int cafe= total/10;
        int rest3= total%10;
        int carte= total/10;
        int rest4= total%10;
        int billet= total/8;
        int rest5= total%8;
        int fleur = rest1 +rest2+rest3+rest4+rest5;
        System.out.println(" le prix l'achat de livres et fournitures est : "+ achat+"DH");
        System.out.println("Vous pouvez ensuite acheter :");
```

```
        System.out.println("-"+cafe+" cafés.");
        System.out.println("-"+carte+" Carte de recharge prépayer.");
        System.out.println("-"+billet+" billets de TRAME.");
        System.out.println("il vous restera "+ fleur+" MAD pour les roses blanches");
}
}
```