ROYAUME DU MAROC
OFFICE DE LA FORMATION PROFESSIONNELLE ET DE
PROMOTION DU TRAVAIL
INSTITUT SPECIALISE DE TECHNOLOGIE APPLIQUEE-BEN M'SIK

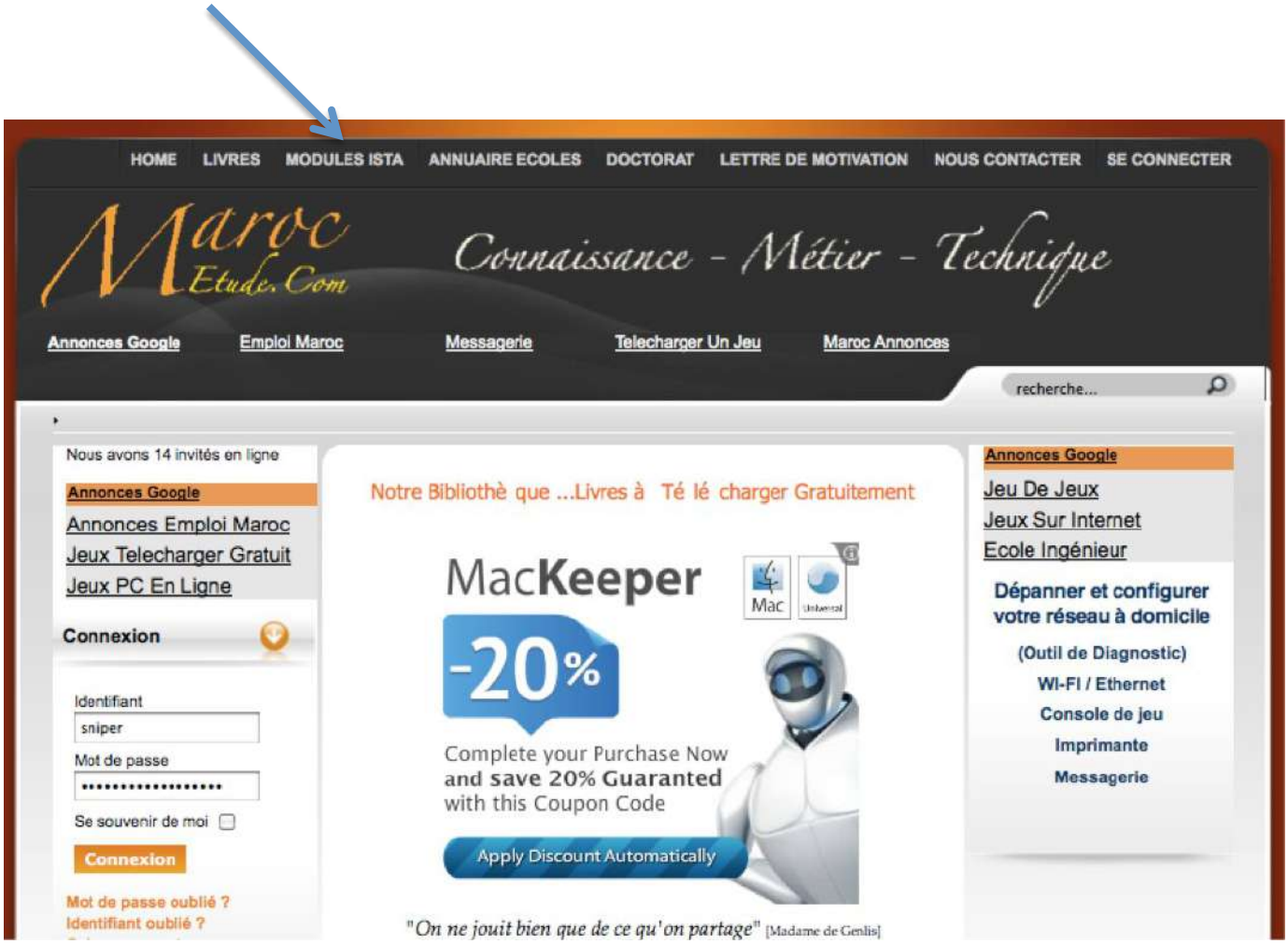
Programmation Orienté Objet

PORTAIL DE LA FORMATION PROFESSIONNELLE AU MAROC

Télécharger tous les modules de toutes les filières de l'OFPPT sur le site dédié à la formation professionnelle au Maroc : **www.marocetude.com**

Pour cela visiter notre site www.marocetude.com et choisissez la rubrique :

MODULES ISTA



I-introduction :

1-C'est quoi le langage Java :

Le langage JAVA est un langage de programmation puissant et à la fois facile à comprendre par le débutant.

2-Comment JAVA

Il répond simultanément au besoin des objets
1- la programmation structuré tel que : C, Pascal, Cobol…
2-la programmation orienté objet : la méthodologie de programmation de l'avenir.

3-Definition

Les langages de programmation sont directement compréhensible par la machine.les étapes de traduction on peut deviser les langages existant en trois grandes catégories :

1-les langages machines.

Un ordinateur ne comprend pas que sa propre langage machine, constitue d'une suite de langue nombre 0 et 1 ce qu'ont appelle le binaire

2-les langages d'assemblage.

Développé aussi des programmes de traduction appelé assembleur pour convertir en langage machine les programmes.

3-les langages de haut niveau.

Sont beaucoup plus attrayant de point de vue des programmeurs.
Tel que C,C++,JAVA…

II-les instructions d'entré/sortie :

1-Affichage :

L'affichage est prémondial au niveau de programmation, si on écrit un programme on a besoin qu'il nous retourne le résultat et comment ce résultat.

*Affichage.java

```
//début de programme
public class Affichage {
    public static void main(String args[]) {
        // l'exécution de programme
        System.out.println("BienVenue a la programmation en JAVA !!");
    }
}
```

Run...

```
BienVenue a la programmation en JAVA !!
```

***les commentaires :**
// : Commentaire en seul ligne
/*Multi lignes*/

Public class+nom { :

La class Affichage est définie par l'utilisateur .le nom commence par une lettre majuscule suivie des lettres ou des chiffres et caractères de soulignement ne doit pas commencer par des chiffres ou des espaces.

Les applications JAVA débutant automatiquement leur exécution à **main**

Exemple:

```
public static void main(String args[]) {
    // l'exécution de programme
}
```

Les parenthèses indiquent que **main** est un bloc de construction de programme appelé méthode.
Les méthodes ont la possibilité d'effectuer des taches et de renvoyer des informations lorsqu'elles ont fini leurs taches.
`void` Indique que la méthode effectue une tache
{et} :qui délimite le corps de la définition d'une class ou une méthode.
`System.out.println` Est connue sous le nom d'objet de sortie standard, permet à l'application JAVA,d'afficher les chaînes et d'autre types d'information.

Exemple:

```
System.out.println("BienVenue a la programmation en JAVA !!");
```

Le **;** appelé une instruction, chaque instruction doit être terminer par **;** on dit aussi fin d'instruction.

2-Afficher sur plusieurs lignes :

Une seule instruction peut afficher sur plusieurs lignes on utilisant des caractères de nouvel ligne. Ceci sont des caractères spéciaux qui indiquent au méthode *print* et *println* de System.out ; à quel moment ils doivent placer le curseur de sortie au début de la ligne suivante la fenêtre de commande.

Exemple:

*Affichage.java

```
//début de programme
public class Affichage {
    public static void main(String args[]) {
        // l'exécution de programme
System.out.println("BienVenue \n à la programmation \n  en JAVA !!");
    }
}
```

Run...

```
BienVenue
 à la programmation
  en JAVA !!
```

3-Caractères de contrôle :

Peuvent être intégré dans une chaîne de caractère.

4-Affichage d'une boite de dialogue .

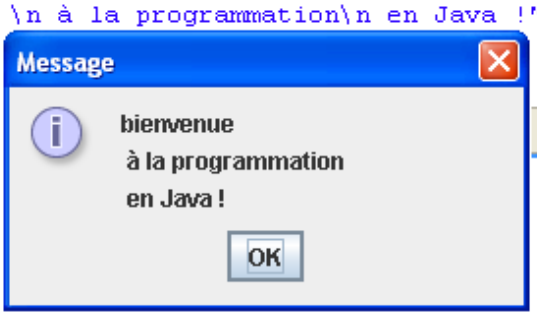
Le langage JAVA permet au utilisateur d'afficher une ou des fenêtres pour rédiger et de lire des messages. Les boites de dialogue sont normalement des fenêtres dans lesquelles les programmes affiche des messages. grâce à la classe ***JOptionPane*** de JAVA fournie des boites de dialogue .

Exemple:

*Affichage.java

```
import javax.swing.JOptionPane;
public class Affichage {
    public static void main(String args[]) {
        JOptionPane.showMessageDialog(null,
                "bienvenue \n à la programmation\n en Java !");
        System.exit(0);
    }
}
```

Run...



- **JOptionPane :**

est défini dans le package `import javax.swing.JOptionPane;`

- **le package :**

c'est le noyau du java ou son stocké,ont doit utilisé l'instruction

import pour l'importation du class.
Le compilateur charge la class ***JOptionPane*** du package javax.swing le swing contient des nombreuse class qui permettais au programmeur java de définir des interfaces utilisateur graphique (GUI-Graphical user interface
La ligne **JOptionPane.showMessageDialog(null,"Bienvenue a la programmation\n java")**

-La méthode ***showMessageDialog*** affiche la boite de message de la class JOptionPane, la méthode a deux argument séparer par, la premier c'est le mot clé null permet a l'application java de déterminer l'emplacement de la boite de Dialogue, c'est l'argument est null la boite de dialogue au centre de l'écran ,la deuxième argument représente la chaîne a affiché à l'exécution de la ligne l'instruction affiche la boite de **Dialog** avec la barre de titre de dialogue contient la chaîne MESSAGE par contre le bouton OK est incorporé automatiquement **System.exit(0)** la ligne est nécessaire dans toute application qui affiche une interface utilisateur graphique pour forcé la fin du l'application .la class System suivi d'un point et de la méthode exit l'argument 0 de exit indique que l'application c'été achevé avec un succès une valeur différente de 0 indique qu'une erreur se serai produite .a class System fait parti du package java.lang.*; par contre la Class n'est pas importé par le package par ce que ce package est importé par dans tout programme en java par défaut

REMARQUE:

L'oublie de l'appelé System .exit dans une application qui affiche une interface utilisateur graphique empêche l'achèvement du programme le résultats se manifeste généralement par impossibilité de taper quoi se soit dans la fenêtre de commande

Type de boite de message	Fenêtre	Icône
`JOptionPane.showMessageDialog(null,"\____","Info", JOptionPane.INFORMATION_MESSAGE);`		
`JOptionPane.showMessageDialog(null," ","Info", JOptionPane.QUESTION_MESSAGE);`		
`JOptionPane.showMessageDialog(null," ","Info", JOptionPane.WARNING_MESSAGE);`		

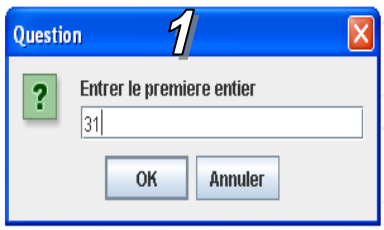
`JOptionPane.showMessageDialog(null," ","Info", 0 );`		
`JOptionPane.showConfirmDialog(null," ");`		
`JOptionPane.showInputDialog(null," ");`		
`JOptionPane.showMessageDialog(null," ","Info", JOptionPane.PLAINE_MESSAGE);`		Aucune

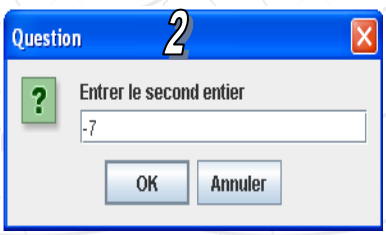
Exemple:

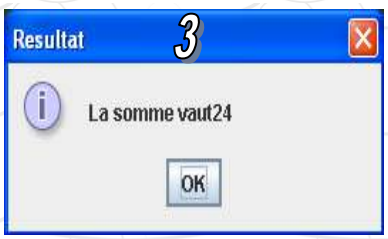
*Addition.java

```
import javax.swing.JOptionPane;

public class Addition {
    public static void main(String args[]) {
        String pNombre, sNombre;
        int Nombre1, Nombre2, Somme;
        pNombre = JOptionPane.showInputDialog(null,
                "Entrer le premiere entier", "Question",
                JOptionPane.QUESTION_MESSAGE);
        sNombre = JOptionPane.showInputDialog(null, "Entrer le second entier",
                "Question", JOptionPane.QUESTION_MESSAGE);
        Nombre1 = Integer.parseInt(pNombre);
        Nombre2 = Integer.parseInt(sNombre);
        Somme = Nombre1 + Nombre2;
        JOptionPane.showMessageDialog(null, "La somme vaut" + Somme,
                "Resultat", JOptionPane.INFORMATION_MESSAGE);
        System.exit(0);
    }

}
```

A l'exécution le programme affiche 1->2->3->







III-LES VARIABLES

Les variables est une case mémoire identifier par un identificateur.
Tous les variables ayant un type avant leur utilisation dans un programme.

1-les types des variables:

Le langage java est considéré un langage fortement typés les types prédéfini doivent apparaître en lettre minuscules (ex:int, char, double).
Les types de langage java sont

Type	Taille (bits)	Valeurs possibles	Désignation
boolean	1	true *ou* false	valeur logique
char	16	0 à 0xFFFF (65535)	caractère unicode (non signé)
byte	8	-128 à 127	entier signé sur 1 octet
short	16	-32 768 à 32 767	entier signé sur 2 octets
int	32	-2 147 483 648 à 2 147 483647	entier signé sur 4 octets
long	64	de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807	entier signé sur 8 octets
float	32	de +/-1.40239846e-45 à +/-3.40282347e+38	réel 4
double	64	de +/- 4.94065645841246544e-324 à +/- 1.79769313486231570e+308	réel 8

2)Les opérateurs d'affectations

Le langage java propose plusieurs opérateurs d'affectation .

Exemple:

C=a+b **C+=5**
C-=2 **C/=4** **G%=9**

3.3)Les opérateurs d'incrémentation et de décrémentation

Le langage java propose l'opérateurs d'incrémentation ++ est de décrémentation -- au lieu de passer par l'expression suivant .

Exemple:

I=I+1 **I=I-1** **a++** **++a** **b--** **--b**

4)Les opérateurs relationnel et d'égalité

Les opérateurs relationnel et d'égalité Ont tous le même niveau de présence.

Exemple:

x= =a , x!=y {opérateurs d'égalité

x>y , x<y , x>=y , x<=y {les opérateurs relationnel

5)Les opérateurs arithmétiques

Toutes les langage effectuent des calcul mathématique java aussi est un langage qui a évolue la méthode mathématique .ainsi il a intégrer des fonction mathématique qui n'existe pas dans d'autre langages.

Exemple:

Opération	Opérateur	Expression en Java
multiplication	*	P*X
division entière	/	résultat entier si diviseur et dividende entier : X/Y
modulo	%	le reste de la division entière :X%Y
addition	+	F+7
soustraction	-	P-X

5-1Les priorités

Java applique les opérateurs selon une séquence précise déterminer par les règles de présence des opérateurs.

1)Les opérateurs dans les expressions continue entre pair de parenthèse sont évaluer en premier lieu par centre l'utilisateurs de java utilise les parentés pour l'ordre d'évaluation dans la séquence de son choix.

Exemple:

y=(a+x+x)+(b*x)+c

2)Les opérateurs de multiplication,de division et de modulo s'applique en suite ,si une expression contient plusieurs opérations de multiplications de division et de modulo,les opérateurs s'applique de gauche a droit par ce que ont le même niveau de présence.

3)Les opérateurs d'addition soustraction
S'applique on derniers lieu ont le même niveau de présence s'il y a plusieurs opérations plus et moins les opérateurs s'applique de gauche a droite.

Exemple:

x=a+b-c+d-y+n

5-2Les méthodes mathématique

Les méthodes de la class Math permettait au programmeur d'effectué des calcules mathématique .les appel des méthodes commence par le nom de la méthodes suivi d'une parenthèse ouvrante suivi de l'argument ou la liste des arguments séparé par des virgule puis de parenthèse fermante .

Exemple:

Math.sqrt (900) , Math.pow(x, 2) ;

6) Les opérateurs logiques :

Le langage java propose de opérateurs logique qui permettais de former des conditions complexe, n combinant des conditions simple . une variable de types boolean ne peut prendre que l'une de deux valeur soit vrai/faux ,true/false , 0/1 ,O/N ,les opérateurs logique java sont prédéfini le langage .

Exemple:

!	non
&&	Et logique /And
&	E boolean logique
^	ou exclusif
|| ou |	ou

Exemple:

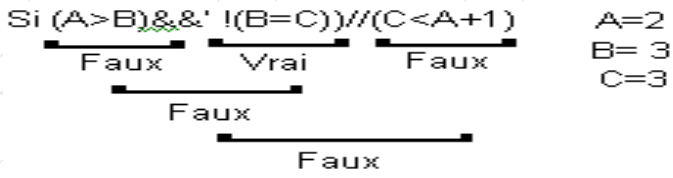
1&&1=1
1&&0=0
0&&1=0
0&&0=1

1 ||1=1
1 ||0=1
0 ||1=1
0 ||0=0

```
si(x>y&& age>=65)
si(moyen>=90 ||  ex final>=90)alors
            écrire ("le grade d'étudiant est A")
```

REMARQUE :

L'opérateur **&&** a une présence supérieur a celle de l'opérateur **||** c'est une expression qui contient les opérateur **&&** ou **||** L'évaluation est de la façon suivante



Exemple: **Table de verite (! Non logique) :**

! 1 = 0
! 0 = 1

si (!B=C)

6-1 Les opérateurs logiques :

Les nombres entier sont limités en grandeur, elle c'est dépend de type de machine, la mémoire centrale est décomposé en zone identique comportant chaque de 6 à 64 bits selon les machines plus bits de parité permettant le contrôle des information chaque de mémoire constituent l'adresse de zone.

6-1 Les nombres positif:

Les nombres entiers sont enregistrés sous les plupart des machines par un moy et un système de numération.
Le système de numération est une méthode de présentation des nombres les plus utilisé en informatique sont : Binaire, Octal, Décimal et Hexadécimal.

Pour toutes les bases est $N=a_{n-1}*b^{n-1}+a_{n-2}*b^{n-2}\ldots\ldots a_1*b^1+a_0*b^0$

Le système Binaire

Le système Binaire ou système de base 2 on a deux symboles

0 et 1

Le système binaire est le système de base utilisé pour l'organisation interne des ordinateurs.

Ex : 10110010

Chaque chiffre est affecté à une puissance de 2. Pour transformer un nombre binaire en décimal, il suffira de multiplier chaque chiffre par 2^n, n étant le rang du chiffre.

Traduisons le nombre binaire 10110010 en décimal :

$10110010 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 128 + 0 + 32 + 16 + 0 + 0 + 2 + 0$

$= 178$

$10110010_2 = 178_{10}$

Traduisons le nombre binaire 10110010 en décimal :

$10110010 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 128 + 0 + 32 + 16 + 0 + 0 + 2 + 0$

$= 178$

$10110010_2 = 178_{10}$

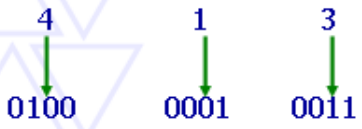
En binaire pur un nombre décimal est traduit globalement ; en D.C.B on traduira chaque chiffre décimal en binaire.

Suivant le rang occupé, chaque chiffre aura une valeur ou un poids déterminé **(1, 2, 4 ou 8)**

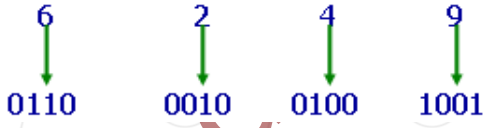
Ex :

$\mathbf{9_{10} = 1001_2}$

Le nombre décimal 413 sera représenté en DCB par le nombre :



De même :



Du système binaire au système octal

1ère méthode :

Cette méthode consiste à transformer le nombre binaire en nombre décimal, puis le convertir du décimal en octal.

Ex :

$10111 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 16 + 0 + 4 + 2 + 1$

$= 23$

$10111_2 = 23_{10}$

23	8	
7	2	8
	2	0

$23_{10} = 27_8$

$10111_2 = 27_8$

Du système binaire au système octal

2ème méthode :

Les nombres binaires de un à trois chiffres ont un correspondant octal d'un seul chiffre.

Binaire	Octal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7

2ème méthode :

Pour convertir du binaire en octal, on décompose le nombre binaire en séries de trois chiffres à partir de la droite. Chaque série donne le chiffre octal correspondant.

Ex :

10111 se décompose en deux séries 010 et 111

les valeurs respectives de ces deux séries sont

2 et 7.

$10111_2 = 27_8$

$1\ 101\ 001\ 111 = 1517_8$

$1\ \ 5\ \ 1\ \ 7$

Du Binaire à l'Hexadécimal

Il est possible de procéder de la sorte :

Binaire ---------- Décimal

Décimal ----------Hexadécimal

Mais on constate que les nombres binaires de 1 à 4 chiffres ont un équivalent Hexadécimal d'1 seul symbole de (1 à F)

Ex :

$10\ 1111\ 0111 = 1517_8$

$2\ \ F\ \ 7$

$1011110111_2 = 2F7_{16}$

$1110\ 0110\ 1111\ 0101 = E6F5_{16}$

$E\ \ 6\ \ F\ \ 5$

Le système décimal

Soit la suite suivante :
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
10, 11, 12, ...20, 21, ...99
100, 101, 102,.....200, ..., 600, ...999
1000, 1001, 1002.....5000, ...9999

On constate que :
-L'écriture de chaque nombre n'utilise que les dix caractères définis dans la première ligne. Ces caractères sont appelés des chiffres.
-A part la première, chaque catégorie commence par un nombre formé par un 1 et des 0
Ex : 10, 100, 1000 .....
-Les premiers nombres de chaque catégorie sont liés entre eux par une relation mathématique.

$100 = 10 \times 10 = 10^2$
$1000 = 100 \times 10 = 10^3$
$10000 = 1000 \times 10 = 10^4$

le système décimal a pour base 10.

Les chiffres formant un nombre ont un sens défini selon leur position (ou selon leur rang)

Ex : 924

On dira que :

4 est placé dans le rang 0

2 est placé dans le rang 1

9 est placé dans le rang 2

et $924 = 900 + 20 + 4$

c'est à dire $924 = 9 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$

on peut dire qu'un chiffre **a** de rang **n** à pour valeur : $\mathbf{a \times 10^n}$

Le système octal

Dans le système décimal nous avions 10 symboles à notre disposition par contre en octal nous n'avons que 8 symboles disponibles.

0, 1, 2, 3, 4, 5, 6, 7

Si on écrit 345 on ne sait pas s'il est en décimal ou en octal. Pour cela on écrit :

715 en base 10 s'écrira 715_{10}

715 en base 8 s'écrira 715_8

De l'octal au décimal

La conversion se réduit à une addition de puissances de 8

Ex : 635_8

$635 = 6 \times 8^2 + 3 \times 8^1 + 5 \times 8^0$

$= 6 \times 64 + 3 \times 8 + 5 \times 1$

$= 384 + 24 + 5 = 413$

Résultat : $635_8 = 413_{10}$

De l'octal au Binaire

On convertit chaque symbole octal en son équivalent binaire. Sur trois chiffres :

$5\ \ 1\ \ 6_{\ 8} = 101\ 001\ 110_2$

101 001 110

$7\ \ 7\ \ 7_{\ 8} = 111\ 111\ 111_2$

111 111 111

De l'octal à l'hexadécimal et réciproquement

La méthode la plus pratique et la plus rapide est de passer par le système binaire :

672_8

$6\ \ 7\ \ 2_{\ 8} = 110\ 111\ 010_2$

110 111 010

$1\ \ 1011\ \ 1010 = 1BA_{16}$

1 B A

$F3A_{16}$

$F\ \ 3\ \ A_{\ 16} = 1110\ 0001\ 0011_2$

1110 0001 0011

$111\ \ 100\ \ 111\ \ 010_{\ 2} = 7472_8$

7 4 7 2

$F3A_{16} = 7472_8$

Le système Hexadécimal

Dans le système hexadécimal ou système de base 16, nous utiliserons 16 symboles qui seront :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Correspondance décimale :

A ---------- 10
B ---------- 11
C ---------- 12
D ---------- 13
E ---------- 14
F ---------- 15

De l'hexadécimal au décimal

Cherchons la correspondance décimale du nombre hexadécimale 1A

$1A = 1 \times 16^1 + 10 \times 16^0 = 16 + 10 = 26$

$1A_{16} = 26_{10}$

$81CF = 8 \times 16^3 + 1 \times 16^2 + 12 \times 16^1 + 15 \times 16^0$

$= 8 \times 4096 + 1 \times 256 + 12 \times 16 + 15 \times 1$

$= 32768 + 256 + 192 + 15$

$= 33231$

$81CF_{16} = 33231_{10}$

De l'Hexadécimal au binaire

Le principe consiste à convertir chaque symbole en son équivalent binaire :

E 1 3 $_{16} = 1110\ 0001\ 0011_2$

1110 0001 0011

Les opérations en code binaire

Addition

Table d'addition :

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (1est le retenu)

ex : 101 + 11

$^{1}1^{1}01$
11
1000

100100 + 10110

$10^{1}0100$
10110
111010

Multiplication

Table de multiplication :

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

ex :

10
x 11
10
10.
1 1 0

1011
x 101
1011
^{1}0000.
1011.
1 10111

Remarque : Il faut faire attention lors de l'addition finale.

Division

```
10110 | 10
0011  | 1011
 010
  0
```

```
111011 | 110
0010   | 1001
  101
  1011
   101
```

Le quotient entier est 1001
Le reste est 101

Soustraction

1) étude directe :

ex :

```
 10101
-  10
-----
 10011
```

```
 100110
-  11010
-------
 011000
```

2) Soustraction par complémentation :

Exemple dans le système décimal :
64 – 31
on prend le complément de 31 qui est 10^2 - 31 = 69
on additionne 64 et 69 et on abandonne le report de gauche.

```
 164
+ 69
----
 133
```

33 est bien égal à 64 - 31

Soustraction

Cette méthode est peu utilisée en décimal mais elle sera pratiquée en binaire, ce sera de plus la méthode utilisée dans les ordinateurs.
Comment trouver le complément à deux d'un nombre binaire ?

- Il faut remplacer tous les zéros par des uns et les uns par des zéros, on obtiendra ainsi le complément à un.
- Ajouter 1 au nombre trouvé précédemment.

Exemple de complémentation à deux:
Supposons le nombre binaire suivant :

1010011

le complément à 1 est : 0101100

\+ 1

on obtient ainsi le complément à deux : 0101101

Soustraction

Exemple de soustraction de deux nombres binaires :

1ère méthode :

```
  1100111101
- 0001010011
------------
  1011011010
```

2ème méthode :

Le complément à deux de 0001010011 est 1110101101

```
  1100111101
+ 1110101101
------------
 11011011010
```

IV-Structure de contrôle

4-1-Introduction :

Ecrire un programme pour résoudre un problème il est essentiel de bien comprendre le problème et bien planifier l'approche de sa résolution et les types de blocs et les principe de construction .les structures de contrôle permettais de construire et de manu piler les objets.

4-2 Définition :

Les instructions d'un programme s'exécute les une après les autre selon l'ordre dans lequel elles ont été écrite : le processus d'exécution séquentiel. Java permettais au programmeurs de spécifier l'instruction à exécuté c'est le transfert de contrôle, la notion de programmation structurés est devenue un synonyme de illimitation de l'instruction "go to". Java ne possède pas "go to" par contre "go to" est un mot réservé de langage il n'est pas utilisé.

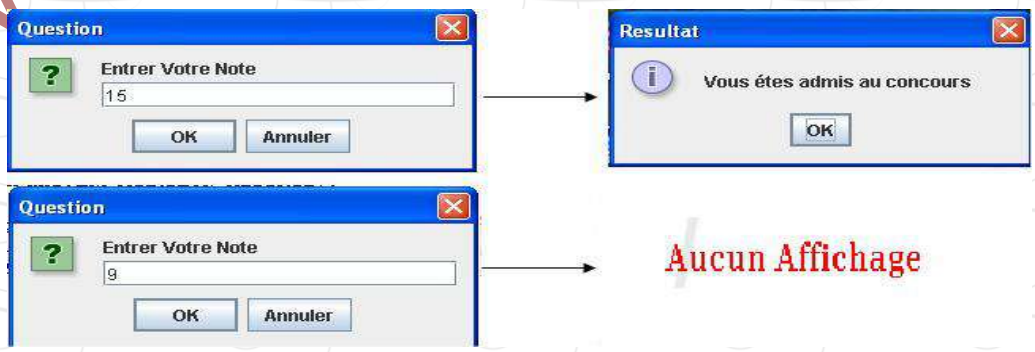
4-3 l'instruction de sélection if :

Une structure de sélection autorise le choix d'activité dans un programme. if teste une condition s'il est vrais exécute l'action , s'il est fausse il ignore.

Exemple:

Addition.java

```
import javax.swing.JOptionPane;

public class Addition {
    public static void main(String args[]) {
        String pNombre, sNombre;
        String X;
        int Moy;
        X = JOptionPane.showInputDialog(null, "Entrer Votre Note", "Question",
                JOptionPane.QUESTION_MESSAGE);
        Moy = Integer.parseInt(X);
        if (Moy >= 10) {
            JOptionPane.showMessageDialog(null, "Vous êtes admis au concours",
                    "Resultat", JOptionPane.INFORMATION_MESSAGE);
        } System.exit(0);
    }
}
```



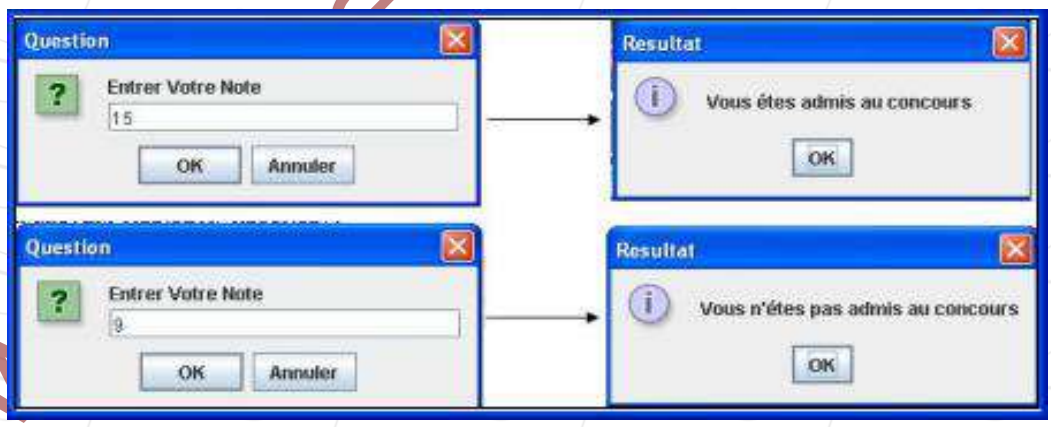
4-4 la structure de sélection if/else :

La structure permet de tester la condition s'il est vrai exécute l'action 1 si non exécute l'action 2

Exemple:

*Addition.java

```
import javax.swing.JOptionPane;

public class Addition {
    public static void main(String args[]) {
        String pNombre, sNombre;
        String X;
        int Moy;
        X = JOptionPane.showInputDialog(null, "Entrer Votre Note", "Question",
                JOptionPane.QUESTION_MESSAGE);
        Moy = Integer.parseInt(X);
        if (Moy >= 10) {
            JOptionPane.showMessageDialog(null, "Vous étes admis au concours"
                    , "Resultat", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null,
                    "Vous n'étes pas admis au concours" , "Resultat",
                    JOptionPane.INFORMATION_MESSAGE);
        }
        System.exit(0);
    }
}
```



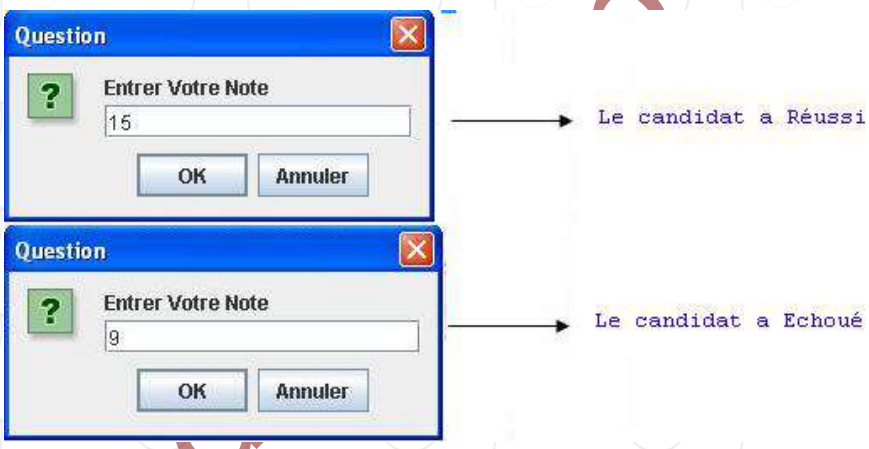
Remarque :

Il existe en Java une opérateur conditionnel (**? :**) est lié à la structure if/else et le seul opérateur tératisme de Java il forme une expression conditionnel

Exemple:

Addition.java

```
import javax.swing.JOptionPane;

public class Addition {
    public static void main(String args[]) {
        String pNombre, sNombre;
        String X;
        int Moy;
        X = JOptionPane.showInputDialog(null, "Entrer Votre Note", "Question",
                JOptionPane.QUESTION_MESSAGE);
        Moy = Integer.parseInt(X);
        System.out.print(Moy >=10?"Le candidat a Réussi":"Le candidat a Echoué");
    }
}
```



a- la structure imbriquée

Il y a des structure imbriqué test plusieurs cas on placent les structure au sein d'autre structure.

Exemple:

```
if (Moy >= 90)
    System.out.print("A");
else if (Moy >= 80)
    System.out.print("B");
else if (Moy >= 70)
    System.out.print("C");
else if (Moy >= 60)
    System.out.print("D");
else
    System.out.print("E");
```

4-5la structure de répétition while :

La structure *while* permet au programmeur de spécifier la répétition d'une action tant qu'une condition est vraie.

Exemple:

```
String note,Comptnote;
int produit=2;
while(produit<=100) {

    //Programme de Moyenne d'une class


    while (cap<=10)
         note = JOptionPane.showInputDialog(null,
                "Entrer un nombre entier ou reel", "Question",
                JOptionPane.QUESTION_MESSAGE);
    valnote=Integr.parseInt(note);
    Total+=valnote ;
```

*MoyClasse.java

```
import java.text.DecimalFormat;
import javax.swing.*;

public class MoyClasse {
    public static void main(String args[]) {
        int compteur, ValNote, total;
        double moy;
        String note;
        total = 0;
        compteur = 0;
        note = JOptionPane.showInputDialog("entrer une not entier-1 pour exit");
        ValNote = Integer.parseInt(note);
        while (ValNote != -1) {
            total += ValNote;
            compteur += 1;
            note = JOptionPane
                    .showInputDialog("Entrer une note ou -1 Pour exit");
            ValNote = Integer.parseInt(note);
            DecimalFormat deuxchiffre = new DecimalFormat("0.00");
            if (compteur != 0) {
                moy = (double) total / compteur;
                JOptionPane
                        .showMessageDialog(null, "La moyenne de la class est:"
                                + deuxchiffre.format(moy),
                                "Moyenne de la class",
                                JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null,
                        "Aucune note n'a été entrée", "Moyenne de la class",
                        JOptionPane.INFORMATION_MESSAGE);
            }
            System.exit(0);
        }
    }
}
```

La classe décimal format sert à mettre des nombres en forme dans notre exemple la dernière fois on a comme valeur deux chiffres après la virgule donc on a déclaré deux chiffres comme une référence à un objet de la classe décimal format qu'est une package au point décimaux et 2chiffres après le point décimal

4-6La structure de répétition for :

La structure for gère tous les détails de la répétition contrôlée par un compteur.

Exemple:

for (i=1 ;i<=10 ;i++) x=2 ;y=10 ;

A éviter for (int y=x; j<=4*2*10; j*=y/x)

a)évolution de la variable de contrôle 1à100 par incrémentation de 1

for (int i=1; i<=100 ;i++)

b) for (int i=100; i>=1 ;i--)

c) for (int i=7;i<=77;ii+=7)

d) for (int i=2;1>=2;i-=2)

e) for (int y=2;j<=20;j+=3)

f) for (int j=99;j>=0;j-=11)

Exercice:

Ecrire le programme qui permet de calculer la somme de n valeur de 1à100 et l'incrémentation de pas par 2

4-7 la structure de répétition (do/while) :

La structure (do/while) test la condition de la boucle après exécution du corps de la boucle .dont le corps de la boucle s'exécute au moins une fois c'est la structure se termine l'exécution se poursuit a l'instruction placé après la close while.

Exemple:

```
do {
    Instruction

    }
while (condition)
```

```
i=0;
do{
    s+=i ;
    i+=1;
    }
while (i>=10)
```

Remarque :

Il existe en Java une opérateur conditionnel (**? :**) est lié à la structure if/else et le seul opérateur tératisme de Java il forme une expression conditionnel

Exercice:

Ecrire le programme qui permet de calculer la somme de n valeurs de 1à100à le pas d'incrémentation est de 4.

4-8) la structure de sélection multi avec switch :

La structure de sélection du types pour gérer une prise de décision afin d'exécuté différentes action en fonction de ses valeurs

Exemple:

```
String entrer ;
Int sortie
Entre=JOptionPane.showInputDialog (" Entrer le 1 pour le teste 1 \n
```

```
                                    '' Entrer le 2 pour le teste 2\n''+
                                    '' Entrer le 3 pour le teste 3\n'') ;
Sortie=Integer.parseInt (enter) ;
Switch (sortie){
Case1 :System.out.print (''le nombre vaut 1'') ;
        break ;
Case2 :System.out.print (''le nombre vaut 2'') ;
        break ;
Case3 :System.out.print (''le nombre vaut 3'') ;
        break ;
Defaut =System.out.print (''le nombre ne vaut ni 1 ni 2 ni 3'') ;
}
```

Exemple:

*Appreciation_Switch.java

```
import java.io.*;
import javax.swing.JOptionPane;
public class Appreciation_Switch {
public static void main (String args[])throws Exception {
BufferedReader IN=null;
      IN=new BufferedReader(new InputStreamReader(System.in));

      String S=JOptionPane.showInputDialog("Quelle est votre note(entier)?");
       int note=Integer .parseInt(S);
           switch (note) {
               case 0:case 1:  case 2: System.out.println("Très insuffisant."); break;
               case 3:  case 4:  System.out.println("Insuffisant."); break;
               case 5: case 6:    System.out.println("Moyen.");  break;
               case 7: case 8:  System.out.println("Bien.");
                   break;
               case 9: case 10: System.out.println("Très bien."); break;
              default :
               System.out.println("Ce n'est pas une note entre 0 et 10.");
              }
   }
}
```

Exercices:

Ecrire le programme qui permet de calculer le carré et le cube du nombre compris entre 0 et 10.

La structure *switch* est constitue du suite de étiquette case, le programme se

compare la valeur d'expression de control qui doit produire obligatoirement une valeur entier de types byte char 'short' ou 'int'
La structure suite s'interrompe immédiatement à l'instruction break.

a)l'instruction break :

L'instruction break provoque la sortie immédiate de toutes les structures

Exemple:

```
for (i=1 ;i<=10 ;i++)
  {
  if (i= =5)
  break;
  Sortie+=i+"/";
   }
```

b) l'instruction continue :

L'instruction continue s'exécute, le contrôle du programme poursuit à l'incrémentation de la variable de contrôle dans la structure for

Exemple:

```
for (i=1 ;i<=10 ;i++)
  {
  if (i= =5)
  continue;
  Sortie+=i+"/";
  }
```

Remarque :

Les programmes considèrent que break et continu violent les règles de la programmation structurée les effets de ces instructions sont réalisable par les techniques de la programmation structurée, il y a des programmeurs n'utilisent jamais ni break ni continue.

b) l'instruction break et continue avec une étiquette :

L'instruction break avec une étiquette prend le rôle d'être exécuter dans tous les structures et provoque la sortie immédiate dans chaque structure de répétition intermédiaire qui l'entour. Par contre, l'exécution du programme reprend à la 1ère instruction après le bloc d'étiquette. Une étiquette est une identification suivie de deux points **stop :**

Exemple:

```
stop :{
        for (i=1 ;i<=10 ;i++)
        {
         for (j=1;j<=5;j++)
        {
         if (i= = 5)
         break stop;
        Sortie+="*";
        }
        }
        Sortie+="\n boucle terminer normalement");  }
```


Remarque:

Si i=5 le bloc est enferme entre { et } de stop, le programme enchaîne immédiatement a la ligne qui suit }.

V-Les tableaux :

5-1-Introduction :

La notion des tableaux c'est très important pour les structures des données, ce dernier c'est la structure des données dynamique tel que liste.

5-2-Définition :

Un tableau est un groupe d'emplacement en mémoire qui porte tous le même nom et le même type pour faire référence à un emplacement donné, c'est-à-dire à un élément du tableau .le programmeur doit spécifié le nom du tableau et le numéro du position de l'élément particulier du tableau placer entre croché, les tableaux en java commence toujours par 0.

5-3-Déclaration et allocation des tableaux :

Les tableaux sont des objets qui occupe le place en mémoire tous les objets de java doivent recevoir une allocation de mémoire .l'opérateur new permet d'allouer de l'espace en mémoire pour les tableaux et d'autres objet.

Exemple:

```
int c []=new int [20] ;
int c [];
c=new int [20];
```

Remarque :

Si vous déclarez un tableau de la façon suivante in C[20] ;
Provoque une erreur de syntaxe. On peut faire les déclaration suivante est valable aussi pour les deux tableaux suivantes
String B[ ]=new string[20],x[ ]=new string[40];

Exemple:

```
1/ String b[]= new String [20], x[]=new String [40];

2/ double [ ]=tableau1,tableau2;
   double [ ]tableau1=new double[10],tableau2=new double[20] ;
```

Définition :

- **Classe JTextArea :**

Propose une zone qui permet de manu piler plusieurs lignes de texte dans la quel on affiche le résultats au sien d'une boite de message.

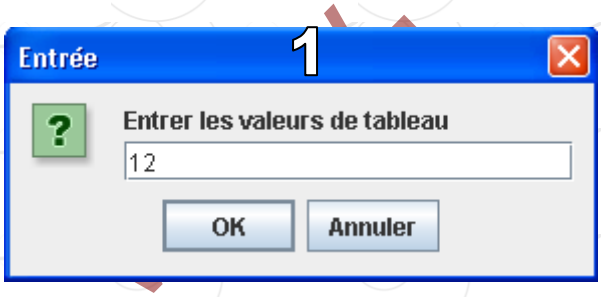
- **t.length :**

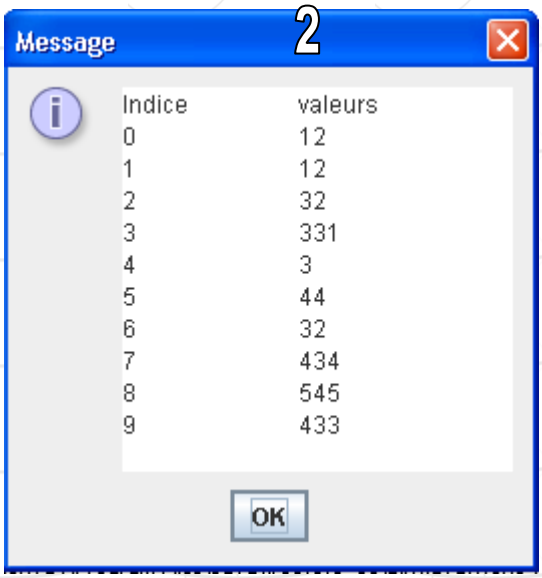
Détermine la longueur du tableau la boucle se poursuit l'exécution tant que la valeur de la variable de contrôle est inférieur à t.length.

INTERTABL.JAVA

```
import javax.swing.*;
public class InterTabl{
        public static void main(String args[])
        {
                int t[];
                int i;
                t = new int [10];
                for(i=0;i < t.length; i++)
                {
String x=JOptionPane.showInputDialog("Entrer les valeurs de tableau");
                t[i]=Integer.parseInt(x);
                }
                String sortie="Indice\tvaleurs\n";
                for( i=0;i < t.length; i++)
                        {
                        sortie += i + "\t" + t[i] +"\n";
                        JTextArea zoneSortie= new JTextArea ();
                        zoneSortie.setText (sortie);
                        }
        JOptionPane.showMessageDialog(null, zoneSortie);
                }
        }
```

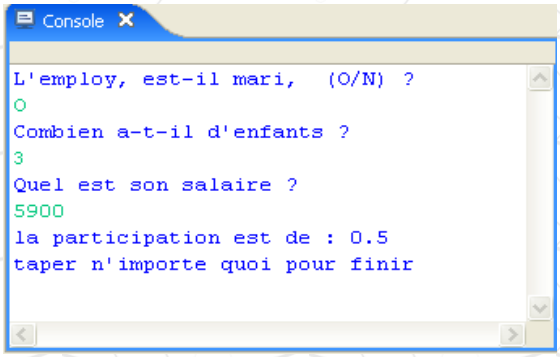
Remplissage:
10 éléments du tableau

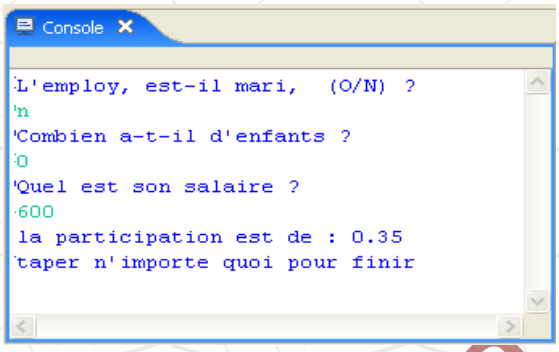




Alternative.java

```
import java.lang.*;
public class Alternative {
    public static String lire() throws java.io.IOException {
        String strSaisie = "";
        char C;
        while ((C = (char) System.in.read()) != '\r') // ou \n
        {
            if (C != '\n') {
                strSaisie = strSaisie + C;
            }
        }
        return strSaisie;
    }
    public static int convertStringInt(String strEntier) {
        Integer intConv = new Integer(strEntier);
        int nConv = intConv.intValue();
        return nConv;
    }
    public static float convertStringFloat(String strEntier) {
        Integer intConv = new Integer(strEntier);
        float nConv = intConv.floatValue();
        return nConv;
    }
    //Point d'entrée principal de l'application.
    public static void main(String[] args) throws java.io.IOException {
        String strRepMarie;
        int nEnfants;
        double nParticipation;
        float nSalaire;
        System.out.println("L'employ\u201A est-il mari\u201A  (O/N) ?");
        strRepMarie = lire();
        System.out.println("Combien a-t-il d'enfants ?");
        String strEnfants = lire();
        nEnfants = convertStringInt(strEnfants);
        System.out.println("Ouel est son salaire ?");
        String strSalaire = lire();
        nSalaire = convertStringFloat(strSalaire);
        if (strRepMarie.equals("N") == true) {
            nParticipation = 0.2;
        } else
            nParticipation = 0.25;
        nParticipation = nParticipation + (nEnfants * 0.1);
        if (nSalaire <= 6000)
            nParticipation = nParticipation + 0.1;
        if (nParticipation > 0.5) {
            nParticipation = 0.5;
        }
        System.out.println("la participation est de : " + nParticipation);
        System.out.println("taper n'importe quoi pour finir");
        String strBidon = lire();
    }
}
```





- **final :**

Permet de déclarer une variable constante s'initialise avant tout utilisation et ne peut s'obéir la moindre modification.

Exemple:

```
Public.....
{final N=10 ;
Int [] ;
T=new int[] ;
```

Remarque :

Lors de toute modification d'une variable final après la déclaration le compilateur affiche une message d'erreur .

- **setText:**

La méthode setText remplace le texte d'un **JTextArea** il permet de remplacer le texte avec une chaîne de sortie.

Exemple:

```
zone sortie (sortie) ;
```

5-4) l'utilisation d'une liste pour initialiser les éléments d'un tableau :

Une liste d'initialiseur, inscrite a l'intérieur des {} est précédé par l'opérateur d'affectation, permet d'allouer et d'initialiser un tableau lors de ça déclaration, le nombre d'éléments de la liste d'initialisation détermine la taille du tableau.

Exemple:

```
int t[ ]={10,20,30,40,50} ;
```

5-5) Tableaux multiples [i] [j] :

Les tableaux a deux indice représente des nombre d'exposé en ligne et en colonne, pour identifier un élément bien déterminer il faut indiquer les deux indice t[i][j] ;
La déclaration et l'allocation d'un tableau multiples se fait de la façon suivante :

Exemple:

```
int T[][]=new int[12][4]
```

Ou bien

```
int T[][]={{1,2},{3,4}}
```

Remarque :

Un tableau a un indice multiples dans chaque ranger possède un nombre d'efférente de colonne peut s'allouer de la manier suivante :

Exemple:

```
int T[ ][ ] ;
T=new int[2][ ] ;     => allocation des lignes
T[0]=new int[5] ;    =>  0 0 0 0 0
T[1]=new int[3] ;    =>  1 1 1
```

VI-Les procédures et les fonctions (méthodes) :

6-1) Introduction :

Tous problèmes peux se décomposer en sous problème c'est adire en groupe d'action dans certain se répéteront plusieurs fois, tant dit que d'autre ne seront exécuté qu'une fois.

6-2) Définition :

Lorsque l'algorithme devient trop complique ont aura envi de le découpé de manière a se que chaque partie soit plus simple et plus lisible la division du programme en plusieurs modules assure une meilleur qualité de programmation .de même lorsque une partie de code doit être exécute plusieurs fois a des endroit différent ou réutiliser ultérieurement en pourra ne décrire qu'une fois lui donner un nom en fessons fonction ou une procédure

6-3) Procédure :

Le procédure et une suite d'instructions servant a réaliser une tache précise en fonction d'un certain nombre du paramètre.

Exemple:

```
Public static void chargement (.....)
{
}
Public static void main (.....)
{
Chargement () ;
}
 }
```

6-4) Fonction :

Une fonction est une procédure dans le but et déterminer une valeur et de retourner au programme appelant.

Exemple:

```
Public static void main(.....)
{
SU=Surf(Z)
}
Public static double surf (double r)
{
double SUR ;
SUR= (Math.pow(r, 2)*Math. PI) ;
Return SUR.
 }
}
```

Remarque :

Par contre, le nom de la fonction est utiliser comme identificateur de tous les types () pour transmettre le résultat de calcul au nom du fonction

Exercice:

Ecrire le programme de fact à l'aide de fonction.

6-5) les paramètre par valeur et par référence :

Il existe deux fonctions de passer des arguments à des méthode ou à des fonctions dans plusieurs langages de programmation, se sont le passage pour valeur et par référence dénommé appelé par valeur ou par référence.

1-par valeur

Le passage par valeur effectue une copie de la valeur de l'argument avant de la passer à la méthode appeler la variable par valeur contient normalement une valeur

Exemple:

```
long x ,y ;
X=0 ; on affecte la valeur  0 à x
X=y ; on affecte la valeur  y à x
```

Y=4 ; on affecte la valeur 4 à y
Donc On a comme résultat les types des variable x=0 et y=4,la modification de y n'a pas modifie x.

Remarque :

Par contre tous les types des variables sont des variables par valeur

2-par référence

Le passage par référence l'appelons permet a la méthode d'applet d'accéder directement à ces donnes et de les modifier au besoin.
L'appel par retourne améliore les performances on évite la copie des donnés surtout lorsque leur taille est importante mais aux des trimes de la sécurité et l'intégrité de ses donnés.

Remarque :

Référence : c'est l'adresse lieu physique on se trouve la variable cette modification sera visible dans la procédure modifié appelante après le retour

6-6) surcharge d'une méthode :

Java permet l'usage de même nom pour plus méthode ou fonction si leur liste de paramètre différant on nombre type par ordre des paramètre (la surcharge des méthodes)
L'appel d'une méthode surcharge, le compilateur java choisi la méthode approprié on examinant le nombre, les types et l'ordre des argument de l'appel .permet aussi de crée plusieurs méthode de même nom et effectue des taches similaire mais avec des types des données différent.

VII-Manipulation des objets des chaine de caractères (class String)

7-1) Introduction

Les fonctionnalités de traitement de la chaîne de caractère de java, permet la validation des saisie à l'affichage des informations à l'instruction de l'utilisateur de texte sorte de manipulation de texte sa deux définition.

7-2) Définition

Caractères constituent les blocs fondamentaux des programmes source en java tant programme est composé d'une séquence de caractère,il sont groupés de manière sensé .permettant à la machine de les interpréter en une suite d'instruction et d'accomplir une tache ,les caractère qui fournissent toute une fondation pour la manipulation des chaînes et des caractères en java sont :

- **String, StringBuffer et character**=Package java.lang

Par contre, la class

- **StringToknize**=Package java.util

7-3) Constructions de chaînes (String)

La class String propose pas moins de neuf constructeur qui initialise des objets String des manières divergées.

Exemple:

```
S=new String (''Bonjour'') ;
S=new String(S) ;
```

7-4) Méthode utilise avec String

- **length** : Permet de donner la dimension d'une chaîne de caractère.

Exemple:

```
String S, Sortie;
 S=new String (''Salut tout le monde'');
 Sortie="S vaut :"+S;
Sortie+="\n la longueur de S vaut "+S.length ();
```

- **charAt** : La méthode "charAt" reçoit un argument entier qui forme le manière de position ou l'index et retourne le caractère situé à cette endroit.

Exemple:

```
Sortie+="\n la chaîne à retours est: '' ;
```

```
For(''int i ;i=S.length ( )-1;i>=0;i++)
Sortie+=S.charAt (i)+'' '';
```

Remarque :

Comme pour les tableaux .le premier élément d'un String est considéré comme à l'emplacement0

- **getChars :** Permet de copier les caractères d'une String dans un tableau du caractère.

Exemple:

```
String S1,Sortie;
 Char tableau car[ ];
 Tableau car=new char[5];
 S1=new String(''Salut tout le monde'');
 S1.getChars(0,1,tableau car.0);
```

***1er argument :** Donne l'indice du tableau ou la copie des caractères de la String commence.
***2eme argument :** Indique l'indice du dernier caractère de la chaîne à copier plus un.
***3eme argument :** mentionne le tableau de caractère vers lequel se père la copie.
***4eme argument :** Indique l'indice de l'élément du tableau à partir de quel le caractère copier sont déposé.

7-5) Comparaison plusieurs chaînes (String)

JAVA fourni une variété d'une méthode de comparaison d'objet String lorsque la machine compare deux chaîne il compare en réalité les codes numérique les caractère de ses chaînes

- **equals :**

Permet de tester l'égalité il es t identique a l'opérateur d'égalité equals (==)

Exemple:

```
S1=new String(''Bonjour'');
 If(S1.equals("Bonjour");
 If(S1=="Bonjour")
```

equalsIgnoreCase

L'égalité il es t identique Ignore la case du lettre de chaque String, la comparaison ne tiens plus conte de la case.

Exemple:

```
S1=new String (''Bon anniverssaire'');
S2=new String (''bon annivairssaire'');
If (S1.equalsIgnoreCase (S2))
Sortie +=''S1 est equal à S2\n'';
else
Sortie +=''S1 est différent de S2\n'' ;
```

compareTo :

La méthode compareTo return 0 si deux String sont égale et return -1 si la chaîne à comparer est inférieur à la chaîne passer a l'argument.
Return 1 si la chaîne a comparer est supérieur a s'elle passer en l'argument.
Il compare les valeurs numériques des caractères correspondant de deux chaînes.

Exemple:

```
S1=new String (''Bonjour'');
S2=new String (''bonjour'');
Sortie +=''S1 compareTo (S2) +
          S2 compareTo (S1) +
          S1 compareTo (S1);
```

regionsMatches :

La méthode compare les parties des deux objets String en vue d'égalité.

Exemple:

```
 if(S1.regionsMatches(0,S2,0,5)
Sortie +=''Les 5 premiers caractères de S1 et S2  correspond'' ;
      else
Sortie +=''Les 5 premiers caractères de S1 et S2 ne  correspond
pas'' ;
```

Remarque :

Teste en tenant compte de la case

Exemple:

```
if(S1.regionsMatches(True,52,0,5)
        Sortie +=''Les 5 premier caractère de S1 et S2  correspond'' ;
else
        Sortie +=''Les 5 premier caractère de S1 et S2 ne correspond
        pas'' ;
```

Remarque :

Teste sans tenir compte de la case

1ere argument : est l'indice de départ de la String qui appel la méthode.
2eme argument : est la String de comparaison
3eme argument : est l'index de départ dans la chaîne de comparaison.
4eme argument : est le nombre de caractère a comparer.

True :la méthode renvoi vrai seulement si le nombre spirifer de caractère sont identique sur le plan lexicographique si le premier argument est true la méthode ignore la case des caractère a comparer le argument restant sont identique a se décrit dans le cas de la méthode précédente .

-Les methods startwith et endwith.

startwith:

Donne le début de la chaîne c'est-à-dire la chaine commence par le ou les caractères suivant

endwith:

Donne la fin de la chaîne c'est-à-dire la chaîne se termine par le ou les caractères suivant.

Exemple:

```
String chaine []= {‘’démarré’’, démarrage’’,’’terminé’’,’’terminer’’} ;
String Sortie=’’ ‘’;
for (int i=0 ;i<chaînes.length ;i++)
if (chaîne [i]=startWith(‘’dé’’))
Sortie+=”\””+chaine[i]+ ”\””commence par \”de \ ‘’\n” ;
Sortie+= ‘’\n ‘’ ;
```

Remarque :

La méthode startWith teste seulement dans l’exemple mais elle a d’autres fonctions.

Exemple:

```
if (chaîne [i].startWith(‘’mar’’,2))
        Sortie+=”\””+chaine[i]+ ”\””commence par \”mar \ ‘’en
        position 2‘’\n” ;
Sortie+= ‘’\n ‘’ ;
```

Remarque :

La méthode startWith teste à partir de l’emplacement 2 de la chaîne

Exemple:

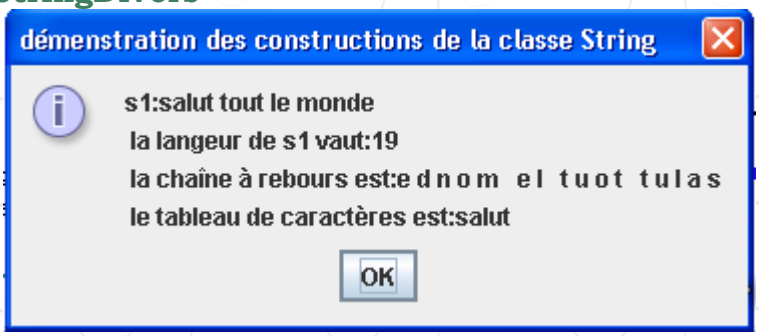
```
If(chaîne [i].endWith(‘’é’’))
Sortie+=”\””+chaine[i]+ ”\””se termine par \”é \ ‘’ \n” ;
Sortie+= ‘’\n ‘’ ;
```

Remarque :

La méthode endWith test simplement la fin de la chaîne

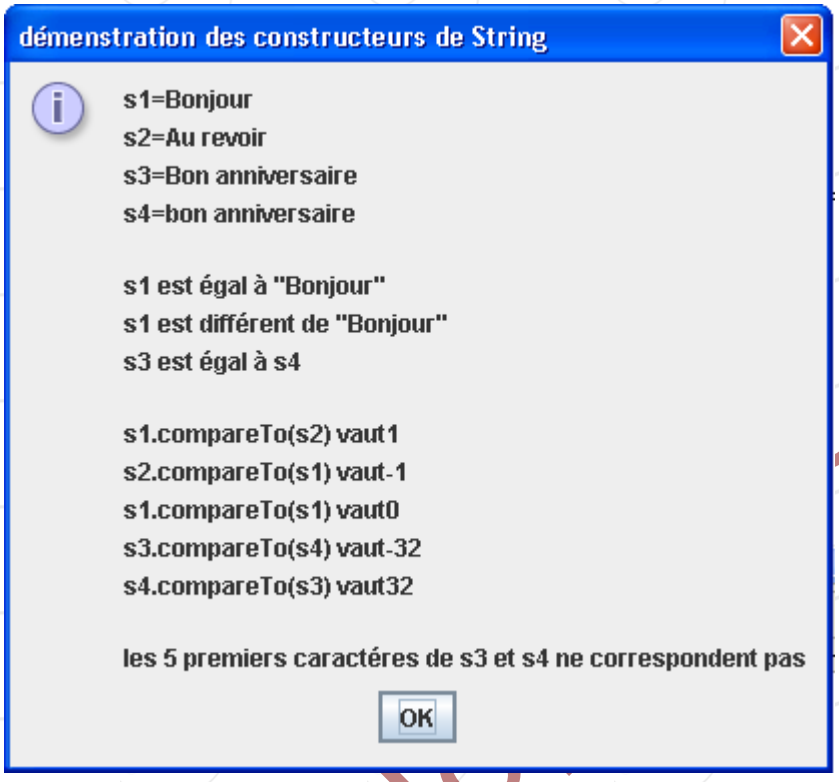
StringDivers.java

```
//démonstration des méthodes length, charAt et getChars de la classe String.
//note: la méthode getChars requiert un point de départ et un point final dans la String. Le point de départ est l'indice réel à partir duquel la copie commence. Le point final est d'une unité après l'indice de fin de copie.
//Packages d'extension Java.
import javax.swing.*;
public class StringDivers {
        //Tester diverses méthodes de String.
        public static void main (String args[]){
        String s1,sortie;
        char tableauCar[];
        s1=new String("salut tout le monde");
        tableauCar=new char[5];
        //préparer la chaine de sortie.
        sortie="s1:"+s1;
        //tester la méthode lenght.
        sortie+="\n la langueur de s1 vaut:"+s1.length();
        //boucler parmi les caractères de s1 et afficher à rebours.
        sortie+="\n la chaîne à rebours est:";
        for(int compteur=s1.length()-1;compteur>=0;compteur--)
                sortie+=s1.charAt (compteur)+" ";
        //copier des caractères d'une chaîne dans un tableau de char.
        s1.getChars(0,5,tableauCar,0);
        sortie+="\n le tableau de caractères est:";
        for(int compteur=0;compteur<tableauCar.length;compteur++)
                sortie+=tableauCar[compteur];
JOptionPane.showMessageDialog(null, sortie,"démonstration des constructions de la classe String",JOptionPane.INFORMATION_MESSAGE);
System.exit(0);
        }
}//fin classe StringDivers
```



comparerstring.java

```
//ce programme présente les méthodes equals, equalsIgnoreCase,
compareTo et regionMatches de la classe String.
import javax.swing.JOptionPane;
public class comparerstring {
	//Tester les méthodes de comparaison de la classe String.
	public static void main(String args[]){
	String s1,s2,s3,s4,sortie;
	s1=new String("Bonjour");
	s2=new String("Au revoir");
	s3=new String("Bon anniverssaire");
	s4=new String("bon anniverssaire");
	sortie="s1="+s1+"\ns2="+s2+"\ns3="+s3+"\ns4="+s4+"\n\n";
	//Tester l'égalité.
	if(s1.equals("Bonjour"))
		sortie+="s1 est égal à \"Bonjour\"\n";
	else
		sortie+="s1 est différent de \"Bonjour\"\n";
	//Tester l'égalité avec==.
	if(s1=="Bonjour")
		sortie+="s1 est égal à \"Bonjour\"\n";
	else
		sortie+="s1 est différent de \"Bonjour\"\n";
	//Tester l'égalité(sans tenant compte de la casse).
	if(s3.equalsIgnoreCase(s4))
		sortie+="s3 est égal à s4\n";
	else
		sortie+="s3 est différent de s4\n";
	//Tester compareTo.
	sortie+="\ns1.compareTo(s2) vaut"+s1.compareTo(s2)+
			"\ns2.compareTo(s1) vaut"+s2.compareTo(s1)+
			"\ns1.compareTo(s1) vaut"+s1.compareTo(s1)+
			"\ns3.compareTo(s4) vaut"+s3.compareTo(s4)+
			"\ns4.compareTo(s3) vaut"+s4.compareTo(s3)+
			"\n\n";
	//Tester regionMatches(en tenant compte de la casse).
	if(s3.regionMatches(0,s4,0,5))
	sortie+="les 5 premiers caractères de s3 et s4 correspondent";
	else
	sortie+="les 5 premiers caractères de s3 et s4 ne correspondent
pas\n";
	JOptionPane.showMessageDialog(null,sortie,"démenstration des
constructeurs de String",JOptionPane.INFORMATION_MESSAGE);
	System.exit(0);
} }
```



7-6) Recherche des caractères et des chaines dans des Strings

indexOf / lastIndexOf:

La méthode indexOf=index de et lastIndexOf=Dernier de
Permet la recherche d'un caractère particulier ou une sous chaine donner dans une String

Exemple:

```
String lettre=''abcdefjhijklmnopqrstuvwxyz '' ;
Sortie+=''\n 'c' se situe à l'index ''+lettres.indexOf('c') ;
Sortie+=''\n 'a' se situe à l'index ''+lettres.indexOf('a',1) ;
Sortie+=''\n '8' se situe à l'index ''+lettres.indexOf('8') ;
```

Exemple:

```
Sortie+=''\n\n le dernier 'c' se situe à l'index ''+lettres.lastIndexOf('def') ;
Sortie+=''\n le dernier 'a' se situe à l'index ''+lettres.lastindexOf('a',25) ;
Sortie+=''\n le dernier '8' se situe à l'index ''+lettres.lastindexOf('8') ;
```

Exemple:

```
String lettre=''abcdefjhijklmnopqrstuvwxyz '' ;
Sortie+=''\n 'def' se situe à l'index
''+lettres.indexOf('def') ;
Sortie+=''\n 'def' se situe à l'index
''+lettres.indexOf('def',7) ;
Sortie+=''\n 'bonjour' se situe à l'index
''+lettres.indexOf('bonjour') ;
```

Exemple:

```
Sortie+=''\n\n le dernier 'def' se situe à l'index
''+lettres.lastindexOf('def') ;
Sortie+=''\n le dernier 'def' se situe à l'index
''+lettres.lastindexOf('def',25) ;
Sortie+=''\n le dernier 'bonjour' se situe à l'index
''+lettres.lastindexOf('bonjour') ;
```

SubString :

La méthode SubString permet la création d'un nouvelle objet String par copie d'une partie d'un objet String existant (veux dire extraire ou soustraire).

Exemple:

```
String lettre=''abcdefjhijklmnopqrstuvwxyz '' ;
Sortie+=''\n le sous chaine de l'index 20 à la fin
''+\+''+lettres.subString(20) ;
Sortie+=les sous Chaine de l'index 0jusqu'a 6 vaut
''+\''''+lettres.subString() ;
```

Remarque :

Si l'index ou bien les arguments sont hors de limites de la chaine le programme génère une syntaxe erreur.

Concat:

est renvoi un nouvel objet String qui contient deux objet String

Exemple:

```
String S=new String(''Heureux'') ;
S2=new String(''Annivairsaire'');
String Sortie="S1+"+S1+"S2="+S2;
Sortie Sortie="\n\n Resultat de S1.concat(S2)=S1.concat(S2);
```

remplace:

Permet de remplacer un caractère par un autre caractère si aucune caractère de 1er argument n'exit dans la chaîne l'original est renvoyé.

Exemple:

```
String S1=new String(''Bonjour'') ;
String S2=new String(''Au revoir'') ;
String S3=new String(''Espaces'') ;
String Sortie="S1="+S1+"S2="+S2+"S3="+S3;
Sortie+="\n\nRemplacement de "o" par "O" dans
S1:"+S1.remplace("o","O");
```

toUpperCase :

Permet de convertir les caractères minuscule en majuscule s'il n y a aucun caractère à convertir en majuscule, la chaîne initiale est renvoyé tell qu'il est.

toLawerCase :

Permet d'effectuer la conversion inverse (**Maj** → **Min**).

Exemple:

```
Sortie+="\nS1.toUpperCase()= "+S1. toUpperCase+
  "S2.toLawerCase()="+S2.toLawerCase() ;
```

trim :

La méthode trim permet de supprimer les espace au début et a la fin du chaîne de caractère

Exemple:

```
Sortie+="\n S3 après trim"+S3.trim() ;
```

toString :

La méthode toString sert a exprimer le contenu d'un objet sous forme du texte la méthode fourni dans la class String garantie que la valeur correcte de la chaine est renvoyé .

Exemple:

```
S1="Bonjour"
Sortie+= ''\n S1 ="+S1.toString();
```

toCharArray :

Permet de crée un nouveau tableau de caractère qui contiens une copie du caractère de la chaine S1 elle affecte au tableau

Exemple:

```
Char tabCare []=S1.toCharArray() ;
```

valueOf :

La class String fourni une série de la méthode statique de class qui prenne des arguments de type divers, convertisse ces argument en chaîne et le retourne se forme d'objet String .les arguments de la méthode valueOf est de type : boolean, char, int, float, lang, double et object
Tous les object sont susceptible d'une conversion en String grâce a la méthode ToString.

Exemple:

```
char tabCar[]={'a','b','c','d','e','f'}
Boolean b=true ;
Char c='Z' ;
Int i=7 ;
Long e=100000000 ;
Float f=
```

intern :

Permet d'obtenir une référence a une chaîne de même contenue que l'objet X est en affecte la référence à Y
La chaîne à la quel Y fait référence est conservé en mémoire par la class String.

Exemple:

```
String S1, S2, S3, S4;
S3 = S1.intern();
S4 = S2.intern();
if (S3 == S4)
 {
Sortie += "\n S3 et S4 ont le même objet en mémoire";
 }
else
{
Sortie += "\n S3 et S4 n'ont pas le même objet en mémoire";
}
```

Remarque :

La méthode intern de String pour obtenir la méthode unique de S1 et S2 est référencé tant par S3 que S4 référencé tant par S3 que S4.

7-7) La class StringBuffer

La class StringBuffer Of nombreuse fonctionnalité une fois q'un objet String crée ne plus question de le modifier ,la class StringBuffer permet non seulement de crée mais également de manu piler de fonction dynamique les informations sous forme d'une chaîne et de modifier des String créé. StringBuffer peuvent stocké un certain nombre de caractère indique par sa capacité,Si la capacité d'un StringBuffer est insuffisante il est automatiquement ajouter pour accepter les caractère supplémentaire il permet aussi d'implanté les operateur + et +=qui assure la concaténation des chaîne.

a)Constructeur StringBuffer

La class String buffer fourni 3 constructeurs :

1-le constructeur par défaut

Pour crée un tempo chaine pour sont aucun caractère et avec une capacité initial de 16carractere

Exemple:

```
Tompon1=new StringBuffer () ;
```

2-le constructeur qui prend un argument "un entier"

De pour crée un StringBuffer sans caractère et dont la capacité initial est spécifié par l'argument entier.

Exemple:

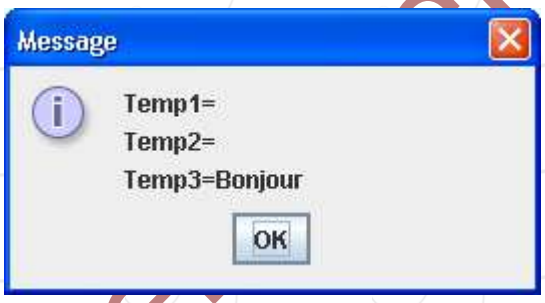
```
Tompon2=new StrigBuffer(10) ;
```

3- Le constructeur qui prend un argument "une chaîne"

Pour crée un StringBuffer contenant des caractères de la chaine donne en argument, la capacité initial est le nombre de caractère de la chaîne.

Exemple:

```
import javax.swing.JOptionPane;
public class POO_exemple {
public static void main(String args[]) {
StringBuffer T1, T2, T3;
T1 = new StringBuffer();
T2 = new StringBuffer(10);
T3 = new StringBuffer("Bonjour");
String Sortie;
Sortie = "Temp1=" + T1.toString() + "\nTemp2=" +
T2.toString()+ "\nTemp3=" + T3.toString();
JOptionPane.showMessageDialog(null, Sortie);
    }
}
```



La méthode ToString pour convertir des StringBuffer on objet chaîne

Remarque :

Les objets String représente les chaînes constantes et les objets StringBuffer sont des chaine modifiable.

Java effectue cette destination entre chaîne constante ou modifiable pour des raisons d'optimisation

Avantage :

Le choix d'utilisé action entre un Objet String et un StringBuffer pour représenté une chaîne .préfère l'objet String si vous étés certain que cette chaîne ne changera pas, se choix améliore considérablement les performances.

Inconvénient :

L'appel sur les objets String des méthodes de StringBuffer alors que ces méthode n'existe pas dans la class String constitue une erreur de syntaxe.

b) Méthodes : length, Capacity, ensureCapacity, setLength :

****length :** Permet de retourner le nombre de caractère présent actuellement dans un StringBuffer .

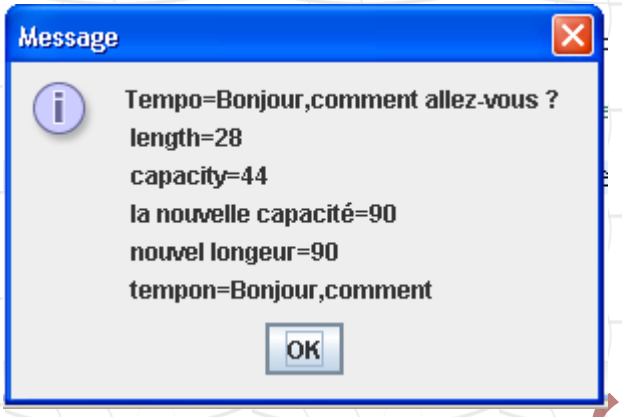
****Capacity:** Permet de retourner le nombre de caractère mémorisable dans un StringBuffer sans n'hésiter d'allocation d'une mémoire supplémentaire

****ensureCapacity:** Permet de garantir au programmeur la capacité minimal d'un StringBuffer .

****setLength :** Permet de augmenter ou de démunie la longueur de StringBuffer .

Exemple:

```
import javax.swing.JOptionPane;
import javax.swing.JTextArea;

public class String_Test {
	public static void main(String args[]) {
		StringBuffer Temp = new StringBuffer("Bonjour,comment allez-
vous ?");
		String Sortie = "Tempo=" + Temp.toString() + "   \n length="
				+ Temp.length() + " \n capacity=" + Temp.capacity();
		Temp.ensureCapacity(75);
		Sortie += "\n la nouvelle capacité=" + Temp.capacity();
		Temp.setLength(16);
		Sortie += "\n nouvel longueur=" + Temp.capacity() + "\n tempon="
				+ Temp.toString();
		JOptionPane.showMessageDialog (null, Sortie);

	}
}
```



Remarque :

La capacité de StringBuffer étendue à une

c) Méthodes charAt,setCharAt,getchars,reverse

charAt:

reçois un argument entier qui forme le numéro de l'emplacement du caractère et retourne le caractère situé à cette endroit

Exemple:

```
S1.charAt(0) ;
S1.charAt(4) ;
```

setcharAt:

permet de définir le caractère a l'emplacement donné ,demande en argument un entier et un caractère pour imposé le caractère situé à l'emplacement spécifié .

Remarque :

L'index donné aux méthode "charAt" et "setcharAt" doit être supérieur ou égale à 0 est inférieur à la longueur de StringBuffer sinon génère syntaxe erreur

Exemple:

```
tempo.setcharAt(0)
Tempon.setcharAt(9 ;'v')
```

getcharAt:

Retourne un tableau de caractère qui contient une copie de caractère présent dans la StringBuffer.
La méthode n'hésite 4 arguments :
*l'index a partir le quel a lieu la copie du caractère du chaîne.
*l'index du dernier caractère a copié a partir de la chaîne +1.
*le tableau des caractère dans le quel les caractère seront copié.
*l'emplacement de départ dans le tableau du caractère ou le premier caractère sera placé.

Exemple:

```
char tableau []=new char [tompon.length()] ;
Tompon .getchar(0,length,tableau,0);
Sortie+="\n \n Voici les  caractère  de tableau :");
For(int i=0 ;i<=tompon.length ;i++)
Sortie+=tabchar[i];
```

reverse: permet de reversé une chaîne de n à 1 (ou bien extraire le contenue de StringBuffer à rebours).

d)Méthodes apprend :

Grâce à StringBuffer et à la méthode append que le compilateur peut mettre en place les opérateurs "+" et "+=" de concaténation de chaîne String.

Exemple:

```
String chaîne 1=''Bonjour'';
String chaîne2=''BC'';
Int valeur =22
String S=chain1+chaine2+chaine3
StringBuffer S=new
StringBuffer().append(''bonjour'').append(''BC'').append(22).toString();
```

e)Méthodes d'insertion et de suppression

La class StringBuffer fourni 9 méthode insert surcharger qui permettais d'insère des valeur de divers type des donné a n'importe quel emplacement dans un String

Exemple:

```
objet O=''Bonjour'';
String S=''Au revoir'';
Char tabCar[la 'a','b','c','d','c','e','j'} ;
boolean b=true;
Char c='15';
Int i=7;
Lang l=10000000;
float f=2,5f;
Double d=33,33333;
StringBuffer tompon=StringBuffer();
Tompon.insert(0,O);
Tompon.insert(0, "  ");
 Tompon.insert(0,s);
Tompon.insert(0, "  ");
Tompon.insert(0,tabCar);
Tompon.insert(0, "  ");
Tompon.insert(0,b);

String Sortie="tampon après insertion:\n''+tampon.toString();
```

.....

- **delete/deleteCharAt:**

Permet de supprimer un ou des caractères à n'importe quelle emplacement dans StringBuffer

- **delete :**

Demande deux argument l'indice de départ et celui de la fin +1 de la suite de caractère a supprimer .tous les caractère compris entre l'indice de départ mais nom compris l'indice de fin sont supprimer.

- **deleteCharAt :**

Ne prend qu'un argument, l'indice de caractère à supprimer.

Remarque :

Les indices non valables provoquent le lancement par les deux méthodes d'une syntaxe erreur (StringOutOfBorndsException)

Exemple:

```
tompon.deleteCharAt(20);
Tompon.delete(2,6);
Sortie+="\n tompon après suppression:\n" +tampon.toString();
```

.....

7-8) La class StringBuffer:

Java fourni un certain nombre de classes qui permettent de traiter des variables de type primitive comme si elles était des objets. Les classes sont boolean, character, double, float, byte, chort, integer, lang. Elle dérive toutes les nombres sauf boolean et character, ses 8 classes on les appels des emballages de type et font partie de package **java.lang** les objet de ses classes sont utilisables n'importe ou dans un programme.

- **isdefined :**

Détermine si le caractère est défini dans la table de code ASCCI si oui retourne vrai si non retourne faux

- **isdegit :**

Détermine si le caractère est un chiffre si oui retourne vrai si non retourne faux

- **isJavaIdentifierStart :** détermine si le caractère est utilisable comme premier caractère d'un identificateur java ,une lettre un caractère de soulignement ou un signe l'art si oui retourne vrai si non retourne faux .

Exemple:

```
character. isdefined(c) ;
character. isletter (c) ;
character. isJavaIdentifierStart(c) ;
```

- **isJavaIdentifierPart :**

Détermine Si le caractère est une lettre d'un identificateur java

- **isLetter :**

Détermine si le caractère est une lettre

Remarque :

La plupart des méthode de la class character sont statique, il prenne au moins un argument pour effectué un test ou une manipulation du caractère, cette class comporte aussi un constructeur qui reçoit un argument char pour initialiser un objet character .il y a aussi les méthode non statique

- **isLetterOrDigit :**

Détermine si le caractère est une lettre ou une chiffre

- **isLowerCase :**

Détermine si le caractère est miniscule

- **isUpperCase :**

Détermine si le caractère est majuscule

Exemple:

```
character.isJavaIdentifierPart(c);
character.isLetter(c);
character.isLetterOrDigit(c);
```

```
"\n le caractère est en miniscule"+character.isLower(c);
"\n le caractère est un lettre capital :"+character.isUpperCase(c);
```

- **toUpperCase :**

Permet de convertir le caractère à son équivalent en capital la méthode renvoi le caractère convertis si le caractère à un équivalent en capital sinon renvoi son argument original.

Exemple:

```
"\n le caractère est miniscule en  majuscule"+character.toUpperCase(c);
```

- **ToLowerCase :**

De convertir le caractère à son équivalent en miniscule la méthode renvoi le caractère convertis si le caractère à un équivalent en miniscule sinon renvoi son argument original.

Exemple:

```
"\n le caractère est majuscule en  miniscule  "+character.toLowerCase(c);
```

****Les méthodes digit et forDigit :**

Effectue les conversion entre les caractères et des chiffres en les différent système de numération

- **digit:**

Converti le caractère en entier dans le système de numération indique par un entier

Exemple:

```
char c ;
character.digit(c,entier) ;
```

- **ForDigit:**converti le chiffre en caractère dans le système de numération indiqué par un entier.

Exemple:

```
Int chiffre ;
Character.forDigit(chiffre,entier) ;
```

Les méthodes non statique de la class non statique de la class character sont les constructeurs **: charValue, toString, hashcode et equals.**

charValue :

Permet de retourner la valeur char dans la méthode character qui est C1 par contre l'objet C2 est renvoyé a String a l'aide de toString.

Exemple:

```
character C1,C2;
  C1=new character('A');
  C2=new character('a');
  String Sortie="C1 ="+C1.charValue()+"\n C2="+C2.toString();
```

hashcode :

La méthode effectue des calcules des codes des découpage pour des objets String est character les valeur des codes de découpage servent a stocké des objet dans des table de découpage on vue de leur récupération rapidement. La table de découpage mémorise des information à l'aide d'un calcul spécial sur les objet a stocké il permet de choisir l'emplacement de la table dans le quel il faut stocké l'objet pour retourne l'information.

Exemple:

```
character C1,C2;
Sortie="\n code de découpage de:"+C1.hashcode() ;
Sortie+="\n code de découpage de:"+ C2.hashcode() ;
```

7-9) La class String Tokenizer:

La lecture d'une phrase, notre esprit découpe cette phrase en mot individuel et signe de ponctuation dans chacun à une signification dans nos yeux.
Le compélateur opèrent un certain découpage en jetons également suivant le même principe. Il décompose des instructions en des petits éléments individuelles, c'est la classe **StringTokenizer** de package **java.util** qui décompose une chaîne en mot, on destingue les mots les uns les autres par délimiteur soit le caractère d'espace, tabulation, retour à la ligne, on retour chariot il y a d'autre caractère peuvent aussi servir de délimiter et séparer les mots.

Exemple:

```
StringTokenizer jeton=new StringTokenizer(chaîne a découpé);
jeton.countTokens()+"\n les jetons sont:\n;
While (jeton.hasMoreTokens())
ZoneSortie.apprend (jeton.nextToken()+"\n");
```

VIII-Traitement des Exceptions:

8-1-Introduction

Une exception indique un problèmes se manifeste pendant l'exécution d'un programme, la faculté de java augmente le nombre et les types d'erreurs peuvent apparaître.

8-2-Définition

Il existe de nombreuse moyens de traité les erreurs, le codes de gestion des erreur se voie d'es percé tous au long du code de logiciel, il sont gérer au endroit même au risque de se produire la meilleur moyens offrir au programmeur qui relie des codes de voir le code de gestion des erreurs et de déterminer si la vérification adéquate mis en place.

8-3-La gestion des exceptions

La mise en place des traitements des exceptions permet au programmeur d'intercepter et de gérer les erreurs au lieu de les laisser se produire et d'ont subir les conséquence quelque exemples des exception :

-d'espacement d'un limites du tableau par l'indice.

- d'espacement arithmétique (lorsqu'une valeur se retrouve hors de l'échelle du valeur représentable)

-devisions par zéro

-déclaration des méthodes non valide

-épuisement de la mémoire

Les traitement des exceptions et destiner a gérer les condition de fonctionnement exceptionnelle qui n'apparaisse pas fréquemment mais qui peuvent se produire .les exception sont des objets de class qui dérive de même super class ;

8-4-Quand utiliser des exceptions ?

Les exceptions est utilisable pour :

-gérer des situations exceptionnelles ou une méthode est un capable d'achever sa tache pour des raisons qui ne peut contrôler.

- traiter les exceptions provenant composons logiciels qui ne sont pas équipé pour gérer directement ses exceptions.

-Gérer les exceptions d'une manière uniforme toute au long d'un grand projet.

Remarque :

Java impose au programmeurs de pondre en conte le traitement des exceptions depuis le commencement d'un projet il doit incorporé une stratégie de gestion des exceptions dans cette projet.

8-5-L'interception des exceptions

L'interception des exceptions sur des blocs protéger suivi du code de prise ne charge des exceptions appeler *traité des exceptions* ou *gestionnaire des exceptions*. Le gestionnaire des exceptions permet de capture, intercepté de l'exception, et l'a traiter

Exemple:

```
public class Exemple_Exception {
        public static void main(String args[])throws exception {

            try
            {
  //Code susceptible de lever ou lancer une  exception
               }
     catch(type d'exception et référencé l'identificateur  )
            {
            //code de gestion d'une exception
            }

}
 }
```

Bloc de code protége(unique)

Gestionnaire d'exception(multiples)

*Bloc **try :***

Une exception qui se produire dans le bloc try et normalement intercepter par un gestionnaire indique de=ans un bloc catch.

Le code qui peut générer une exception et tous codes qui ne peut pas s'exécute si une exception se produit ;

*Bloc **Catch :***

Spécifie le type d'exception qui peut intercepter et contient un gestionnaire d'exceptions, ce dernier bloc catch est suivi d'un bloc ***finally***.

*Bloc **finally :***

Est optionnel offre le code qui doit toujours s'exécuté qu'une exception donné se produise on nom.

Exemple:

```
try{
  //code
    }
  Catch(Exception.....)
  {
  //traitement des Exceptions
  }
  Catch(Exception.....)
  }
  Finally
  {
  //introductions de liberation de resources
  }
```

8-6-Lancer une exception:

Un gestionnaire d'exception peut décider de ne pas traiter l'exception qu'il est sensé gérer, soit parce que il estime ne pas être en mesure de le faire, soit il souhaite on laisse le traitement à un autre gestionnaire catch.
La relance de l'exception se fait par l'instruction suivante:
throws new référenceException;

Exemple:

```
public double quotient(int i, int d)throws DivisionparZeroException
{
If(d==0)
Throws new DivisionparZeroException();
Return (double) i/d;
}
```

Remarque :

Si le dénominateur, le corps de if exécute l'instruction throws qui crie et lance un nouvel objet DivisionparZeroException et intercepter par le bloc catch dont l'argument et de type arithmétique exception qui se situer directement après le bloc try.
Le bloc catch spécifie un nom de paramètre arithmétique exception destiner à recevoir l'objet d'exception lancé, le gestionnaire arithmétique exception convertie l'exception on une String par la méthode toString, la chaîne est passé au tant que message à afficher dans une boite de dialogue d"erreur.

Exemple:

```
Catch(ArithméticException  arithméticException)
{
JOptionPane.showMessageDialog(this,arithméticException.toString(),
"Exception arithmétique",JOptionPane.ERROR_MESSAGE);
}
```

this :

Est une référence se refaire implicitement au variable d'Instance et au méthode d'un objet.

Instance :

Thermes indiquant qu'un objet est crié à partir d'une classe.

Throws :

Reprend une liste des exceptions qu'une méthode est susceptible de lancer.

IX-Les fichiers:

9-1-Introduction:

Le stockage de donnée dans des variables et des tableaux est de nature provisoire, les données disparaissent dis que les variables n'est plus aperte ou dis que le programme se termine, par contre les fichiers retiennent indéfiniment de grand volume des données même après la fin du programme qui on est la source.

9-2-Définition:

Le traitement du fichier est l'un de fonctionnalité incontournable qui doit posséder un langage pour réaliser les applications commerciales, qui gère une masse inorme des données permanant.

Le traitement du fichier ni qu'un sous ensemble des fonctionnalités de traitement mise à la disposition du programme pour lire et écrire les données en mémoire.

Les librairies de la plat forme Java offre un grand nombre de classe dédié au les Entré/Sortie **E/S**, ce classe se trouve dans le package **java.io** il y a aussi des sous package a été introduit pour compléter "java.io" avec des nouvelles fonctionnalités.

9-3-Hiérarchie des données:

Un ordinateur traite toutes les données comme combinaison de 0 et 1, si les caractères sont composés de bit, les champs en sont composés de caractère, un champ est un argument de caractère ou d'octet qui possède un signification.

- **Un champ** :
  Désigne le nom de la personne plusieurs champs que java appel variable d'Instance composant un enregistrement que java mise en sous forme d'une classe.

Exemple:

Un logiciel de paie, un enregistrement relatif à un employé donné pourrait être constitué des champs suivants :

- **Numéro d'identification de l'employé = int**
- **Nom = String**
- **Prénom = String**
- **Adresse = String**
- **Salaire horaire = double**
- **Nombre de déduction = int**
- **Cumule des salaires = « int » « double »**
- **Retourne fixe = « int » « double »**

Chacun de ces champs est relié au même employé (on l'appel **enregistrement**), donc un enregistrement de paie pour chaque employé (on l'appel **fichier**).
Un fichier est un ensemble d'enregistrement, pour faciliter la récupération d'enregistrement spécifique dans un fichier, au moins un des champs est choisie comme **clé d'enregistrement**, un clé désigne exclusivement l'enregistrement associé à une personne ou un article parmi tous les enregistrements des fichiers.

Exemple:

Le champ du numéro d'identification de l'employé peut être considérer comme clé de l'enregistrement.

9-4-Organisation séquentielle :

Il existe plusieurs façons d'organiser des enregistrements dans un fichier. L'organisation la plus courante est dite (**l'organisation séquentielle**).
Dans un fichier séquentiel les enregistrements sont classés par ordre de la clé d'enregistrement. Un groupe de fichier associé est appelé **une base de donnée**.
L'ensemble des programmes conçus pour gérer et manipuler des bases de données portent le nom **"Système de gestion de base de donnée".**
Tous fichiers se terminent soit par une marque de fin de fichier, soit par un numéro d'octet précis consigné dans la structure des données administratives maintenu par le système.

Java résume ce concept sur des informations que lui donne le programmeur. L'identification de fin du fichier apparaît sous la forme exception, tant que dans d'autre cas il se présente comme une valeur de retour d'une méthode invoquée sur un objet de traitement de flux. Un programme java ouvre un fichier en créant à priorité 3 objets de flux qu'elle associe aux périphériques au début de son exécution.

Exemple:

System.in
System.out
System.err

Les flux associés à ces objets fournissent les canaux de communication entre un programme et un périphérique bien déterminé.

- **System.in :** l'objet de flux d'entrée standard.
  Permet à un programme d'entrer des données saisies au clavier ou de lire des octets d'une source différente.
- **System.out :** l'objet de flux de sortie standard.
  Permet d'afficher les données à l'écran.
- **System.err :** l'objet des flux d'erreur standard.
  Permet de sortir des messages d'erreur à l'écran.

Par contre les 2 objets «**.out**» et «**.err**» permettent d'envoyer les sorties à une autre désignation comme dans un fichier sur disque.

La classe **"System"** propose aussi des méthodes qui permettent de rédiger les flux d'entrée, de sortie et d'erreur standard comme :
"setIn, setOut, setErr".

Les programmes Java effectuent un traitement de fichier à l'aide de la classe de package **java.io.* ;** comporte des définitions pour les classes des flux tel que **"FileInputStream"** permet l'entrée d'octet à partir d'un fichier ou bien lire dans les fichiers.

- **File Reader :**

Permet l'entrée des caractères à partir d'un fichier.

- **File Writer :**

Permet la sortie des caractères à partir d'un fichier.

Pour effectuer des entrées et sorties de type de donnée. On peut aussi utiliser des objets de la classe **"ObjectInputStream, DataInputStream".**

9-5-Etude d'un fichier :

Pour écrire séquentiellement dans un fichier on utilise la classe **printWriter**.

Exemple:

Ouverture :

```
String  Nfich="Donnes.dat" ;
printWriter  f=new printWritrer(new BufferedWriter
                    (new FileWrite(Nfich)));
```

Ou bien:

```
printWriter  f=new printWriter(new File("Donnes.dat")) ;
```

Ecriture :

```
f.print(nom) ;
```

Ou bien :

```
f.println(nom) ;
```

Fermeture :

```
f.close() ;
```

Par : www.developper.c.la