JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Généralités & Historique**	**n°0**

1. **Qu'est-ce que ce langage ?**
   - **JAVA :** ***langage de Programmation orienté objet.***
   - Il est inspiré du langage " C "**.**
   - Il nécessite un Compilateur **Java.**
   - Les Navigateurs habituels savent exploiter les programmes **Java** compilés (***applets*** *).*
   - Il permet de résoudre les problèmes que **HTML** ne saurait pas soultionner (***animations*** par exemple)**.**
   - Le programme compilé (applet) est très peu encombrant .C'est le Navigateur du client qui l'interprêtera.

1. **Ses Caractéristiques :**
   - On écrit un programme " **source "** à l'aide d'un banal Editeur ***ASCII (*** **Edit** du **DOS** par exemple) **.**
   - Le programme compilé (applet) est **portable** puisqu'il pourra être interprêté par le Navigateur d'un PC, d'un Mac, d'une console Sun, Unix etc..
   - L'applet pourra être insérée au sein d'une page Web (HTML).

1. **Ses Versions..historique :**
   - **JAVA** a été développé par **Sun Microsystems** en 1991..
   - **JAVA** a été intégré dans **HotJava** en 1994 . **HotJava** a été écrit en quelques mois en **JAVA.**
   - Peu après **Sun** mit sur le marché le **JDK** (Java Developer's Kit). Plusieurs versions existent.
   - Vous serez amené à **télécharger le KIT JDK** .
   - Il existe actuellement des outils de Développement **Java** tels que :
     - **VJ++** de Microsoft
     - **Café** de Symantec
     - La société Borland travaille actuellement à l'élaboration d'un tel outil (**Latte**).

1. **Java est totalement indépendant du Système d'exploitation client :**
   - La " chaine " de développement est la suivante :
   - Ecriture d'un **programme source** grâce à un Editeur ASCII. (Edit du Dos ou Bloc-Notes de W95..etc..).
   - Compilation grâce au complilateur **JAVAC** livré avec le **JDK** qui crée un programme (une **applet).**
   - Cette **applet** contient des **bytecodes** qui pourront être **interprêtés** par tout Navigateur compatible **Java.**
   - Ainsi, cette même applet étant insérée dans une page Web (**HTML**) pourra être utilisée sur une plte-forme MAC , PC sous W95 ou NT, UNIX, SUN etc...
   - De plus la capacité (en Koctets) de cette applet est très faible ..ce qui minimise les temps de transferts.
   - Le tribut à payer de cette organisation en **bytecodes** sera la **vitesse d'exécution** des programmes qui doivent être réinterprêtés par les Navigateurs.

1. **Java est orienté Objet (POO)**
   - Ce qui permet de créer des programmes souples, modulaires et capables de réutiliser notre code.
   - Les programmes utiliseront les objets du Système (ex: ceux de Windows du PC).

1. **Java est issu du C, C++**
   - Ceux d'entre-vous qui connaissent déja le langage C ou C++ seront à l'aise avec l'étude de **JAVA.**

JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Le Kit JDK de JAVA** n° **1**

Où se le procurer gratuitement ?

Sur le Réseau..bien sûr ..Par exemple sur le site **FTP Java de Sun :**

ftp://java.sun.com/pub/ ou
ftp://www.blackdown.org/pub/Java/pub/

Ce sera la version la plus récente.
Installez-le dans un Répertoire de Windows 95 par exemple...Il créra ses sous/répertoires lui-même en décompressant l'éxécutable général.
Vous ne devez pas décompresser le fichier **CLASSES.ZIP** . Java travaille avec ce fichier en mode compressé.

Organisation de notre Etude

- Ce cours nous entraine dans des Expérimentations illustrant toutes les séquences d'information.
- Un Exercice termine chaque séquence d'étude.
- Pour réaliser ces travaux, voici quelle sera notre organisation :

    - Nous lançerons une session **MSDOS** sous **Windows95**.
    - Nous créerons un répertoire **\JAVA_SRC**
    - Nous écrirons nos programmes **source** avec l'éditeur **EDIT** du DOS.
    - Nous utiliserons le compilateur du Kit : **javac** pour compiler nos sources.
    - Nous utiliserons **appletviewer** du Kit pour visualiser nos **applets.**
    - Nous utiliserons aussi **Edit** du Dos pour écrire nos fichiers **HTML.**

- Vous n'aurez pas besoin d'être connecté au réseau pour réaliser les travaux de ces " ateliers ".
- A partir d'un certain niveau du cours vous pourrez être amenés à tester vos applets sur les Navigateurs Netscape ou Internet Explorer.
- Je vous proposerai aussi de télécharger un Logiciel de conversion de type de fichiers Audio le moment venu . Afin d'exploiter le **son** dans vos **applets.**
- Plus tard, je vous indiquerai quels sont les outils logiciels et la méthode qui vous permettront de rendre **transparente** une image **GIF** (animations graphiques).

Objectifs du Cours

- Vous devrez être capable de créer un programme JAVA récapitulatif et représentatif des principales séquences d'étude du Cours.
- Vous devrez être capable d'intégrer l'**applet** ainsi construit au sein d'une Page Web .
- Vous matriserez les principes de base du langage **HTML.**

*Le document suivant vous invite à créer votre **premier programme Java** , le compiler et l'essayer..*

JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Premier programme Java** n° **2**

Je vous sens fébrile ..alors commençons modestement pour se donner du courage

1er Programme

- Il doit, tout simplement ecrire votre nom à l'écran (enfin dans une fenêtre graphique).
- N'oublions pas que Java est un langage Objet donc nous devrons créer une classe...
- Mais tranquillisez vous , vous en prendrez l'habitude.
- Ne vous étonnez pas, non plus si nous devons préciser à Java que nous utiliserons sa classe **applet** qui nous permet de créer des **applets** et sa classe **Graphics** car nous sommes toujours en mode graphique.

Enfin voila comment écrire tout ça..!!

Avant d'attaquer ...

- Vous devez savoir que ce langage vous oblige à respecter les **minuscules** et les **majuscules**
- Vous devez savoir aussi qu'un seul espace entre 2 mots est équivalent à 100 ou 200 espaces ou une ou plusieurs lignes vides. Donc n'hésitez pas à éclaircir vos écritures .
- Vous pouvez placer des commentaires pour rendre plus clair (plus compréhensible) votre programme , pour cela vous pouvez utiliser le **double slash (//)** au cours d'une ligne ..alors tout ce qui se trouve à droite est commentaire.
- Attention aux **;** qui terminent toujours les lignes contenant des instructions
- Attention aux **{** qui ouvrent des blocs , des fonctions ou des procédures .
- Attention aux **}** qui les referment.
- Remarquez l'**indentation** des lignes d'écritures qui permettent de rendre le programme plus lisible..donc plus facile à réparer ou modifier ou simplement à comprendre.
- Prenez bien soin à placer à la même position colonne une **parenthèse ouvrante et sa parenthèse fermante**
- Les n° de lignes ci-dessous ne sont pas à recopier ..ils ne servent qu'à mieux nous comprendre !

```
1.   import java.applet.Applet ;              // cette classe est la " mère " de notre applet
2.   import java.awt.Graphics;                // importons aussi la classe graphique pour écrire !
3.   // sautons 1 ligne pour y voir plus clair
4.   public class monNom extends Applet       // notre applet s'appelera donc : monNom.class
5.   {
6.        public void paint( Graphics gr)     // utilisons la classe Graphique importée
7.        {
8.             gr.drawString(" Bonjour Jeannot Lap ", 30 , 40 );
9.             // on écrit dans la fenêtre la phrase placée dans les guillemets à 30 pixels de la gauche
10.            // de la fenêtre et à 40 pixels du haut de  la fenêtre..
11.       } // n'oublions pas de refermer la parenthèse juste en face de sa soeur ouvrante.
12.  } // n'oublions pas de refermer la parenthèse fermante juste en face de sa soeur ouvrante.
```

Le document suivant vous fournit des explications complémentaires sur ce mini-programme...

JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Premier programme Java** n° **3**

Suite..

Explications complémentaires concernant ce 1er programme

- les lignes 1 et 2 permettent d'importer 2 classes appartenant à Java.
- la ligne 4 va créer une nouvelle classe (c'est notre **applet**) elle descendra de la classe Applet ..elle héritera de tous ses " gènes " mais , en plus, nous allons définir ce que nous voulons qu'elle soit capable de faire.!
- Elle contiendra une fonction **paint** qui est déclarée **void** (on dira qu 'elle est vide ..car elle fera ce que nous voulons mais ne retournera pas de résultat à l'appelant).
- Cette fonction **paint** devra dessiner graphiquement nos caractères (phrase placée entre les guillemets), avec la police , la couleur, la dimension etc.. par défaut ..).

Comment écrire ce programme?

- Tout d'abord , nous avons décidé d'appeler notre Classe **monNom** . Donc notre programme source Java s'appelera: **monNom.java** . Remarquez le respect des minuscules et majuscules !!
- La ligne de commande DOS sera :
  C:\java_src\edit monNom.java
- **Ecrivez** sous Edit ce programme et **enregistrez**-le.
- Maintenant vous devez le **compiler** . La ligne de commande Dos sera :
  C:\java_src\javac monNom.java
- Si Dos vous rend la main sans message c'est que vous n'avez fait aucune erreur de syntaxe ..tout va bien ..En revanche, si une ou plusieurs erreurs vous sont signalées, repèrez bien les n° de lignes indiqués et revenez chez EDIT pour retrouver la ou les panne(s) !!
- Supposons que le Dos vous ait rendu la main sans que le compilateur n'ait bronché !...
- ..alors le compilateur a créé un fichier : **monNom.class** dans votre sous/répertoire.
- Pour essayer votre merveilleux programme il faut écrire encore une ou deux lignes dans un nouveau fichier.
- Nouvelle ligne de commande DOS :
  C:\java_src\Edit monNom.html
  Le contenu de ce nouveau fichier (**monNom.html**) en langage HTML est **:**

  <applet code=monNom width=200 height=200></applet>

  Explications :
  En fait vous venez d'écrire en langage **HTML** une " mini " page **Web .** Ce fichier ne comporte qu'une ligne au cours de laquelle on déclare l'insertion d'une **applet Java** dont le nom est **monNom.class** .
  On dit aussi que la fenêtre nécessaire pour afficher le travail de notre programme aura **200 pixels de largeur** et **200 pixels de hauteur.** *Nous reviendrons plus tard sur les notions de base du langage **HTML.**.*

- Maintenant vous pouvez lancez le **Visualisateur d'applet** du Kit **JDK** . La ligne de commande **DOS** est :
  C:\java_src\appletviewer monNom.html

- L'écran bascule en mode **graphique** ..en revenant chez **Win95** et vous admirez , émerveillé, la fenêtre de votre **applet** et son contenu ..

Le document suivant commence par résumer les grandes phases de ce développement...

JAVA...*le must pour le Web !!*

INTERNET ...Construction de Pages WEB.	**Résumé ..synthèse** n° **4**

Suite..

Résumé des différentes phases de ce Développement

- **Ecriture d'un programme " SOURCE "** en langage **JAVA** à l'aide d'un **Editeur ASCII** sain.

  C:\java_src\EDIT prog.java *Si le programme s'appelait:* ***prog.java***

- **Compilation de ce programme grâce à JAVAC .**

  C:\java_src\javac prog.java *Ce compilateur génère le fichier classe :* ***prog.class***

- **Ecriture d'un programme HTML** pour tester cette applet :

  C:\EDIT prog.html *Afin d'insérer l'applet dans une mini-page Web..*

- **Tester le Fonctionnement** de l'applet, grâce à **appletviewer** :

  C:\java_src\appletviewer prog.html *Ce qui nous ramène chez Windows pour voir l'applet..*

Réflexions préalables

Ces différentes **phases du Développement d'un programme** se répèteront pour tous les suivants . C'est la raison pour laquelle vous devez les avoir bien assimilées.
Rassurez vous, elles deviendront petit à petit familières et automatiques.
Ce programme était très simple et son but était justement de vous permettre de bien comprendre cette chronologie des phases de développement et la fonction de chacune d 'elles.

Les programmes suivants vous amèneront à l'étude du Langage **JAVA** proprement dit de manière progressive et pédagogique .

Chaque étude sera illustrée par un **programme expérimental** et vous proposera au moins un **exercice** en fin de séquence.

Pour ceus d'entre vous qui connaissent déja le Langage **C** ou **C++** , certaines études de base pourront paraitre trop faciles ..Ils jugeront eux-mêmes de l'opportunité ou non de les sauter ..!

Le document suivant commence l'étude du Langage ***JAVA.***

JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Instructions et Variables** n° **5**

Instructions et Expressions

C'est une opération élémentaire du Langage.. Par exemple:

- **import java.* ;**
- **Toto = 45 ;**
- **System.out.println(" Bonjour ") ;**
- **Result = x + y - 24 ;**
- **int PosX ;**

Toutes ces lignes sont de simples instructions ..Elles se terminent toutes par un point-virgule (**;**) **.**

Attention à l'oubli de ce **point-virgule** en fin d'instruction qui peut empêcher le compilateur de faire son travail et ainsi de vous signaler des erreurs !!

Souvent une suite d'instructions peut être placée en **bloc .** Les **blocs** d'instructions sont délimités par des **accolades** ..rappelez vous, on ouvre un bloc avec une accolade **ouvrante** et on le ferme avec une **fermante** ex:

```
{
        String chaine = " bravo " ;
        int nbre=12 ;
        x = x + y ;
}
```

Variables et Types de Données

Les Variables sont définies par:
un **nom** (identificateur), un **type** , une **valeur** .

En **JAVA** ***les variables doivent toujours être déclarées avant leur utilisation*** .
Voici quelques Exemples :

- **boolean Drapeau ;** *//la variable de* ***type boolean*** *s'appelle* ***Drapeau***
- **int age ;** *//la variable.....****int*****.....****age****.*
- **String MessageFin ;** *//.....****String*****.....****MessageFin***
- **String MesGeneral , Mes1 , Mes2 , chaine ;** *//plusieurs variables de même* ***type*** *de noms différents.*
- **float Var1, valeur ;** *//idem.....*
- **int A=12 , B=16, C=17 ;** *//il est possible d'affecter des* ***valeurs*** *en les déclarant.*
- **String JourSem = " Lundi " ;** *//idem.....*

Règles et conseils concernant les Identificateurs de Variables

Les noms de variables répondent à quelques contraintes :

- Ils ne doivent pas commencer par un chiffre
- Ils peuvent commencer par une **Lettre**,le signe **$**, le caractère de soulignement (_)
- Essayez de ne pas utiliser : **%** * **@** qui sont réservés à certains opérateurs du Langage..
- Attention les **majuscules** et les **minuscules** sont bien différenciées..Donc :
  La variable **Toto** n'est pas la même que **TOTO** ou **ToTo** ou **toto**
- En général il est intéressant de choisir un nom de variable qui rappelle sa fonction dans le programme . Et si cette fonction fait intervenir plusieurs mots , essayez de choisir un nom utilisant une contraction de ces mots en prenant soin de faire précéder chacun d'eux par une Majuscule(et sans espace !)...Par exemple:
  Une variable de type **int** représentant les valeurs prises par un **index** dans un **tableau de codes:**
  int IndTabCod = 0 ; ou **int Ind_Tab_C = 0 ;**

Ceci est un conseil permettant de rendre plus compréhensible, plus lisible un programme **JAVA.**

JAVA...le must pour le Web !!

INTERNET ...Construction de Pages WEB.	**Types de Variables**	n° **6**

Types Elémentaires des Variables

Java propose les Types élémentaires suivants

- **boolean** *ne contient que 2 valeurs (**vrai** ou **faux**) ..soit , in English : **true** ou **false***
- **byte** *octet signé (8 bits) ..Donc de -128 à +127*
- **short** *2 octets signé (16 bits).. Donc de -32768 à +32767*
- **char** *Caractère Unicode 16 bits (non signé)*
- **int** *entier signé (32 bits) ..Donc de -2.147.483.648 à +2.147.483.647*
- **long** *entier signé (64 bits)..Donc de -9223372036854775808 à +9223372036854775807*
- **float** *nombre à virgule flottante simple précision (32 bits)*
- **double** *nombre à virgule flottante double précision (64 bits)..Enorme !*

1er Exercice

Il vous manque encore quelques connaissances pour être capable d'écrire un programme Java seuls .
Donc, je vous propose de recopier le programme source ci-dessous ...mais ***Attention il comporte quelques erreurs de syntaxe que vous devrez corriger afin qu'il fonctionne en tant qu'applet.***

```
1.  import java.awt.* ;
2.  import java.applet.* ;

3.  public class exo1 extends Applet ;
4.  {
5.      int posX=20,posY=20;              //déclaration des variables numériques
6.      String ch="Essai de Texte";       //déclaration de la variable de type String
7.
8.      public void paint(Graphics g) ;
9.      {
10.     g.drawString(ch,posX,posY);       //1ère Ecriture du texte
11.     posX=posX+10;  posY=posY+20;      //incrémentation des varibles numériques.
12.     g.drawString(ch,posX,posY)        //2ème Ecriture... etc....
13.     posX=posX+10;  posY=posY+20;
14.     g.drawString(ch,posx,posy);
15.     posX=posX+10;  posY=posY+20;
16.     g.drawString(ch,posX,posY);
17.     }
18. }
```

Appelez ce programme source : **exo1.java. Créez** un programme **exo1.html** comme vous l'avez appris précédemment.
A nouveau je vous rappelle que les n° de lignes ne sont pas à entrer dans votre source.!
Nous verrons bientôt que la conception de ce programme est ridicule car on retrouve 4 appels successifs à la fonction de dessin du Texte . (en lignes 10,12,14,16) . Or il existe une structure Java permettant de réaliser des boucles beaucoup plus pratique que la répétition des actions à réaliser (si on avait dû écrire 50 fois le texte !!!).
Mais l'objectif de ce petit programme était surtout d'illustrer la manipulation des variables numériques .

Le Document suivant vous fournit le corrigé de cet exercice , donc ne copiez pas ! ..

suite..

Corrigé de exo1.java

```
import java.awt.* ;
import java.applet.* ;

public class exo1 extends Applet          // cette ligne ne peut pas se terminer par ; car l'accolade ouvrante suit
{
   int posX=20,posY=20 ;                  // Ici déclaration de plusieurs variables et affectations de valeurs
   String ch="Essai de Texte";            // Déclaration de Chaine de caractères et affectation d'un texte

   public void paint(Graphics g)          // de même ici ..pas de ;  en fin de ligne
   {
   g.drawString(ch,posX,posY) ;           // en revanche à la fin de toutes ces lignes d'instructions il faut le ;
   posX=posX+10;  posY=posY+20 ;
   g.drawString(ch,posX,posY) ;
   posX=posX+10;  posY=posY+20 ;
   g.drawString(ch,posX,posY) ;
   posX=posX+10;  posY=posY+20 ;
   g.drawString(ch,posX,posY) ;
   }
}
```

Les erreurs n'étaient que des ajouts ou oublis de point-virgule (**;**)
Le Fichier **exo1.html** pouvait comporter le contenu suivant :

<applet code = exo1 width=300 height = 200 ></applet>

*Je profite de cet exercice pour vous conseiller (si vous ne l'avez pas encore fait !) , d'utiliser **DOSKEY** du DOS lorsque vous êtes en session DOS . Pour cela , ajoutez la ligne*

DOSKEY

*à la fin de votre fichier **AUTOEXEC.BAT.***
Vous gagnerez beaucoup de temps et de fatigue digitale en pouvant ainsi rappeler vos commandes DOS.

Les Opérateurs d'affectation de JAVA

Opérateur	*Par exemple*	*Signifie*
=	**x = 12**	**x = 12**
+=	**x += 20**	**x = x + 20**
-=	**x -= 5**	**x = x - 5**
***=**	**x *= 4**	**x = x * 4**
/=	**x /= 3**	**x = x / 3**
%=	**Y %= 2**	**reste de la division entière de Y/2** (le Modulo)
^=	**Z ^= 3**	**Z = Z ^ 3** (ou binaire exclusif de Z avec 3)
<<=	**Val <<= 3**	**Val = Val * 8** (contenu de Val glisse de 3 bits à gauche)
>>=	**Xyz >>= 4**	**Xyz = Xyz /16** (gliss. de 4 bits à droite .. 16= 2 puis.4)

Avec le Document suivant vous connaîtrez les Opérateurs de comparaison.. ! ..

suite...

Les Opérateurs de Comparaison de JAVA

Opérateur	*Par exemple*	*Signifie*
>	**M1 > M2**	**M1 plus grand que M2**
<	**N3 < 12**	**N3 plus petit que 12**
>=	**Var >= 25**	**Var plus grand ou égal à 25**
<=	**Var <= 63**	**Var plus petit ou égal à 63**
==	**X == Y**	**X égal à Y**
!=	**m != 41**	**m différend de 41**
&&	**exp1 && exp2**	**ET logique entre exp1 et exp2**
||	**e5 || e4**	**OU logique entre e5 et e4**
&	**val2 & 15**	**ET binaire entre val2 et 15**
|	**val3 | 3**	**OU binaire entre val3 et 3**
^	**valor1 ^ 1**	**OU exclusif entre valor1 et 1**

Incrémentation et Décrémentation

Ces fonctions nous viennent encore du Langage **C .**
Elles sont très compactes et , de ce fait, simplifient beaucoup le programme " source ".
Par ex. :

i++ *permet d'incrémenter* ***i*** *de 1 ..c'est donc équivalent à :* **i = i + 1**
On appelle cela une ***PostIncrémenation.***

i-- *permet de décrémenter* ***i*** *de 1..c'est équivalent à :* **i = i - 1**
On appelle cela une ***PostDécrémentation.***

Il existe aussi:

++i incrémente i de 1 également mais cette incrémentation se fera **avant** d'exécuter l'instruction de la ligne . *On appelle cela une* ***PréIncrémentation.***

--i idem..pour cette décrémentation mais elle aura lieu **avant** le traitement de l'instruction de la ligne.
On appelle cela une ***PréDécrémentation.***

Exercice à résoudre directement sur ce Document

Répondez à droite des Questions ci-dessous (*en imaginant ces différentes questions consécutives au sein d'un programme JAVA)*:

Questions	***Actions réalisées ou résultats et Vos explications*** *(si nécessaire)*
int x = 12;	_____
x++ ;	**x =** _____
x-- ;	**x =** _____
x*= 10 ;	**x =** _____
x %= 3 ;	**x =** _____
x <<= 4 ;	**x =** _____
x /= 2 ;	**x =** _____
Y = --x * 3 ;	**Y =** _____

Les Structures de contrôle de JAVA:

Ce sont les principaux éléments de la programmation. Elles permettent de réaliser des Boucles conditionnelles, des branchements conditionnels ...Nous allons commencer leur étude par :

La Boucle while .. *(Tant que ..)*

Elle permet de répèter une ou plusieurs instructions de programme **tant qu'une condition est vraie.**
Ex.:

```
X = 6 ;
while ( X  < 22 )          // Tant que X est plus petit que 22 on réalise les 2 instructions suivantes
{
        X += 2 ;           // X = X +2....donc X s'incrémente de 2 à chaque tour
        var1 = X / 2 ;
}                          // Dès que X atteindra la valeur 22 le programme quittera la boucle
                           // et se poursuivra vers les lignes suivantes...
```

Conclusion : *Dans cette Boucle* ***la condition est testée en début de boucle*** *..donc si la condition est fausse dès le début la boucle ne sera pas exécutée et aucune des instructions se trouvant dans la boucle ne sera exécutée.*
Dans notre exemple, la boucle sera ici exécutée ***7 fois .***

La Boucle do ... while *(Faire ...tant que)*

Ex.:

```
X = 6 ;
do                         // Faire sans condition
{
        X += 2 ;           // X = X +2....donc X s'incrémente de 2 à chaque tour
        var1 = X / 2 ;
} while ( X < 22)          // Dès que X atteindra la valeur 22 le programme quittera la boucle
                           // et se poursuivra vers les lignes suivantes...
```

Conclusion : *Dans cette Boucle la* ***condition est testée en fin de boucle*** *..donc quelque soit l'état de la condition les instructions seront exécutées au moins une fois .*
Dans notre exemple, la boucle sera ici exécutée ***8 fois .***

2ème Exercice

Vous pouvez créer maintenant un 2ème programme Java en modifiant **exo1.java** afin qu'il devienne **exo2.java.**
Son cahier des Charges sera le même c'est-à-dire écrire plusieurs fois le Texte .
Mais il devra utiliser une boucle de type **while** ou **do while** pour écrire 5 fois le texte **Essai de Texte** .
Testez-le et losqu'il fonctionnera correctement, collez son superbe Listing bien commenté ci-dessous ...!!

suite..

La Boucle for

Elle est utilisée pour répèter des actions un certain nombre de fois . Elle peut être paramètrée de manière plus ou moins complexe. Elle est très employée.
Elle apparaît dans tous les langages de programmation, mais avec une syntaxe et une souplesse qui dépendent de ce langage. En JAVA elle conserve la même syntaxe que celle du langage **C** ou **C++.**

En voici un exemple :

```
for ( x=0 ; x < 20 ; x++)
 {
    g.drawString(" Je répète.. " &  x+1 & "  fois " , x*10+20 , y*20 +40 );
 }
```

Explications:

La 1ère ligne se lit de la manière suivante :
*Pour **x=0 au début** et **tant que x est plus petit que 20 , x sera incrémenté de 1 à chaque tour de boucle.***
*Donc, au 1er tour de boucle, **x=0***
*au second tour **x=1***
*au 3ème **x=2** ...et ainsi de suite ..le dernier tour sera effectué avec **x=19***
*Donc la boucle **for** effectuera 20 tours au cours desquels **x évoluera de 0 à 19** .*

Conclusions:

- **la variable de boucle** (par ex. ici **x**) **est souvent appelée l'index.**
- **Cette structure de boucle possède 3 paramètres :**
  - **initialisation de l'index** (ici : x=0)
  - **condition de fin de boucle** (ici : x< 20)
  - **action réalisée à chaque tour de boucle.** (ici : x++)
- **les autres actions réalisées à chaque tour de boucle suivent l'instruction *for.***
  Ici écrire à l'écran... les accolades ne sont pas nécessaires dans ce cas car une seule action existe. Mais , souvent elles le sont car plusieurs actions peuvent être effectuées ..(d'autres boucles par exemple. On parle alors de boucles imbriquées..!).

Remarques et conseils :

- Attention, en principe pas de point virgule en fin de ligne **for** :
  sinon les instructions suivant l'instruction **for** et devant être effectuées à chaque tour de boucle seront considérées **hors de la boucle .**
- Il est tout à fait possible de placer plusieurs instructions d'**initialisation** , ou plusieurs **actions** de boucle... Par ex.:

```
for (J=10 , toto=12 ; J > 1 ; J -- , toto+= 2) ;
```

- **A votre avis .. que se passe - t-il avec l'instruction for ci-dessus ?**
  - **pendant l'initialisation:**
    _____
  - **quelle est la condition de boucle ?**
    _____
  - **quelles sont les actions à chaque tour de boucle ?**
    _____

suite..

La Boucle for *(suite..)*

Il est possible de ne déclarer la variable d'index que dans la boucle **for** . Elle ne sera alors connue qu'au sein de cette boucle. Par ex.:

```
for ( int nbre=5 ; nbre < 200 ; nbre += 5)
{ actions diverses ....  }
```

Remarques générales aux boucles

- Il est possible de quitter précipitemment une boucle grâce à l'instruction : **break** . (nous reviendrons sur cette utilisation dès que nous aurons appris l' instruction de test **if**).
- Chaque boucle peut être repérée par un label (ou étiquette). Par exemple :

```
        N=1 ;
boucle1:        // label de repèrage de la boucle extérieure.
        while ( N < 20 )
        {
          for ( X=100 ; X < 150 ; X +=10 )
           {
                g.drawString(" Essai ",X-50,N*10);
           }
          N ++ ;
        }
```

Essayez de prévoir

ce que vous verrez à l'écran en éxécutant la compilation de la partie de programme ci-dessus:

_____
_____
_____
_____
_____

*Maintenant appelez ce programme **exo3.java** en insérant cette partie de programme dans votre programme java **exo2** que vous modifierez . Vérifiez vos prévisions ...Collez votre Listing ci-dessous (toujours bien commenté).*

L'instruction de Bloc :

Il est toujours possible de grouper plusieurs instructions à l'intérieur d'un Bloc .
Cet ensemble est délimité par des accolades : **{ }**
Les variables déclarées à l'intérieur d'un Bloc ne sont connues que dans ce Bloc..elles deviennent Locales à ce Bloc.
Par exemple :

```
Float axe=20.3;
{ int capteur=12;
  System.out.println( " Valeur de axe =  " + axe + "  Valeur du capteur =  " + capteur);
}
  System.out.println( " Valeur de axe =  " + axe + "  Valeur du capteur =  " + capteur);
```

L'instruction de Test : if

Elle permet d'exécuter des instructions en fonction du résultat du **test booléen** qui suit le mot clé ***: if***
Le résultat d'un test Booléen ne peut être que Vrai (true) ou Faux (false).
L'instruction (ou le Bloc d'instructions) suivant le Test booléen sera exécuté si le résultat du Test est VRAI .
Par exemple :

```
if  ( X > Y )
     System.out.println("  X est plus grand que Y  " );
```

Remarquez bien le point-virgule qui termine , ici, l'instruction if…Il ne se trouve absolument pas derrière le Test Booléen ..!!

Il est possible de prévoir une instruction ou un Bloc d'instructions à réaliser dans le cas où le résultat du Test est FAUX.
Par exemple :

```
if ( X == Y )
    System.out.println( " Les 2 variables sont égales ");
else
   System.out.println( "  X est différent de Y  " );
```

On remarque là :

1. que le test booléen qui teste l'égalité de X et Y utilise l'opérateur de comparaison : **==**
2. que si le **test est vrai** (X égal à Y) on effectue l'instruction d'affichage d'égalité des valeurs
3. que si le **test est faux** (X différent de Y) on effectue l'instruction suivant le mot clé : ***else***

Le document suivant vous permet de vous familiariser avec l'instruction : ***if….else***

Suite…

L'instruction de Test : if .. else

Voyons au travers de quelques exemples, quelques cas de structures de Test..

```
if ( note >= 18)
   {
    System.out.println(" Bravo…Excellent travail ..! ");
    Max = true;
   }
else {
       if ( note >= 12)
                System.out.println (" Travail correct.. ");
       else     System.out.println(" Résultat insuffisant ");
      }
```

Explications :

*Si la variable **note est égale ou supérieure à 18** ..2 actions sont réalisées :*

1. *Affichage du compliment*
2. *La variable booléenne Max devient Vraie …*

*Si la variable **note est inférieure à 18** ..elle est retestée ..si elle est **égale ou supérieure à 12** ..alors*
Affichage du message 'Travail correct'
***Sinon** (alors elle est inférieure à 12) on affiche le message 'Résultat insuffisant'.*

Remarquez bien que **else** appartient toujours au dernier **if** en cours.

La Forme d'expression de if imbriquées

Cette forme de test est possible mais néanmoins, à éviter car il existe une instruction de contrôle prévue à cet effet.
Etudions quand même cet exemple :

```
if ( carac='E')
       Edition();
else if (carac='D')
       Efface();
else if (carac='F')
       Fin();
```

En effet, cette structure peut-être remplacée avantageusement par la structure de contrôle **switch** que nous allons découvrir d'ici peu !.

Remarques concernant les Test if

Le Test d'égalité utilise l'opérateur ' == ' et non pas ' = ' (qui est réservé aux affectations).
Les conditions à tester sont toujours entre parenthèses et sont évaluées par odre de priorité :
Parenthèses, Gauche vers Droite, Priorité des Opérateurs.

Le document suivant vous amène à résoudre un petit exercice…

Suite…

Quelques petits exercices (if .. else)

1. Quel est le message qui apparaitra sur l'Ecran ?

```
X = 12 ; c = 120 ;
Y = 28 ; C= 150 ;
if  ( ( X < Y )  &&  ( (C – Y)> c))
        System.out.println(“ Le Test est réussi..! ”);   //_____________________________
   else
        System.out.println (“ Le Test est Faux..! ”);   //_____________________________
```

2. Quel est le message qui apparaitra sur l'Ecran ?

```
X = 12 ; c = 120 ;
Y = 28 ; C= 150 ;
if  ( ( X < Y )  ||  ( (C – Y)> c))
        System.out.println(“ Le Test est réussi..! ”);   //_____________________________
   else
        System.out.println (“ Le Test est Faux..! ”);   //_____________________________
```

3. Quel est le message qui apparaitra sur l'Ecran ?

```
X = 12 ; c = 120 ;
Y = 28 ; C= 150 ;
if  ( ( X  & 8 )  == 4)
        System.out.println(“ Le Test est réussi..! ”);  //_____________________________
   else
        System.out.println (“ Le Test est Faux..! ”);   //_____________________________
```

4. Quel est le message qui apparaitra sur l'Ecran ?

```
X = 12 ; c = 120 ;
Y = 28 ; C= 150 ;
if  (( Y >>2 ) == 7)
        System.out.println(“ Le Test est réussi..! ”);  //_____________________________
   else
        System.out.println (“ Le Test est Faux..! ”);  //_____________________________
```

Vous pouvez obtenir le corrigé de ces exercices …

L'instruction de contrôle switch

Si l'instruction de test **if…else** convient très bien pour les situations définies uniquement par 2 conditions , elle n'est pas adaptée aux conditions multiples..En revanche l'instruction **switch** est faite pour ça .!
Exemple :

```
switch ( toto )
{
 case 10 : System.out.println(" cas 10 "); break;
 case 25 : System.out.println(" cas 25 "); break;
 case 78 : System.out.println(" cas 78 "); break;
 default : System.out.println(" toto atteint une valeur en dehors des 3 cas ..! ");
}
```

Explications :
*Si **toto** prend la valeur **10** alors les instructions placées derrière **les 2 points** sont exécutées. L'instruction **break** fait 'sauter' tous les autres cas **et quitter la structure switch**.*
*Idem pour les 2 autres **cases.***
*Le **cas default** est optionnel et permet **de traiter tous les autres cas .***

Switch s'applique uniquement aux types de données primitifs (byte, char, short, int).
Autre exemple :

```
switch (carac)      //attention..pas de point-virgule !
{
 case '+' :  Z=x+y; break;
 case '-' :  Z=x-y; break;
 case '*' :  Z=x*y; break;
 case '/' :  Z=x/y; break;
 default : System.out.println(" mauvaise touche..! ");
}
```

Il est possible d'exploiter de manière intelligente l'instruction **break.**
Par exemple :

```
switch (valor)  // si valor prend les cas 1 ou 3 ou 5 ou 7 la structure saute au cas 9 directement…
{
 case 1 :
 case 3 :
 case 5 :
 case 7 :
 case 9 : System.out.println(" Valor est IMPAIRE "); break;
 default : System.out.println(" Valor est PAIRE ");
}
```

Remarque : *pour utiliser des tests de contrôle sur les chaines de caractères vous devrez utiliser des Tests if..else car switch ne gère que les types primitifs.*

Comment quitter la boucle (for, while, do) en cas d'événement imprévu ?

Il existe les mots clés **break** et **continue** .
Par exemple (utilisation de **break**):

```
int x=2, n=0;
while ( x < 42)
 {
   x +=4 ;
   if ( x==30) break;   // on quitte la boucle définitivement
   Z[n]= x>>2;
 }
```

Par exemple (utilisation de **continue**):

```
int x=2 , n=0;
while ( x < 42)
 {
   x +=4 ;
   if ( x==30) continue;   // on passe au tour suivant sans réaliser l'instruction suivante.
   Z[n]= x>>2;
 }
```

Il est aussi possible de 'sauter' avec **break** ou **continue** vers un label .
Par exemple :

```
etiqette1 :       // label ..vous choisissez un nom comme pour une variable ..suivi de 2 points ( ' :')
 for (int i=0 ; i<3 ;i++)    // boucle labellée
   for(int j=0;j<5;j++)
    {
      System.out.println(" I= "+i+ "   et J= "+j) ;
      if ((i+j) == 5) break  etiquette1;  // si i=1 et j=4 alors on quitte la boucle de label etiquette1
     }
System.out.println(" Sortie des 2 boucles imbriquées.. ");
```

Explications complémentaires :
*La boucle **for extérieure** est une **boucle labellée 'etiquette1'**. Donc l'instruction **break etiquette1** signifie : quitter cette boucle.. Donc sortie complète des boucles.*
*L'instruction **break** sans le **label** aurait déclenché **la sortie de la boucle intérieure** pour revenir dans la boucle extérieure.*

Les boucles , les structures de Test et de Contrôle vont être utilisées dans presque tous vos programmes ..
..elles sont l'essence des langages structurés..

Le document suivant va vous faire découvrir la création et la gestion des Tableaux …