Introduction

La **programmation** est l'ensemble des activités qui permettent la conception de logiciels

Les différentes étapes de la réalisation d'un programme

- Analyse du problème
- Réalisation d'un algorithme
- Codage dans un langage de programmation
- Traduction en langage machine et exécution
- Test et suppression des erreurs

Analyse du problème

- Gestion des réservations pour un gîte
- Édition des statistiques des ventes
- Calcul des chiffres d’affaire réalisés par les commerciaux
- Réalisation d’un jeu
- Réalisation d’un site internet dynamique

Etc …..

Un algorithme, c’est une suite d’instructions, qui une fois exécutée correctement, conduit à un résultat donné.

un algorithme doit contenir uniquement des instructions compréhensibles par celui qui devra l’exécuter

Exemple d’algorithme

Allez tout droit jusqu’au prochain carrefour
Si la rue à droite est autorisée à la circulation
Alors
  Tournez à droite
  Avancez
  Prenez la deuxième à gauche
Sinon
  Continuez jusqu’à la prochaine rue à droite
  Prenez cette rue
  Prenez la première à droite
Finsi

Les langages de programmation

Les algorithmes doivent être traduits dans un langage de programmation:

Exemples :

- C++
- Pascal
- PHP
- Javascript
- Visual Basic , Visul Basic Application (VBA)
- Etc…

Traduction et exécution

Avant d'être exécuté par le processeur, le programme doit être traduit en binaire (0-1)

Traduction

Langage Évolué ⟶ Langage machine

VBA

A = 3	01001001 11100 0011
B = 5	01001001 11101 0101
MsgBox (A+B)	11110011 11100 11101
	11111000 11111

↓ Exécution

Ce travail de traduction est effectué par

– Des compilateurs

Ou

– Des interpréteurs

Différence entre compilateur et interpréteur

Un compilateur, traduit la totalité du programme en langage machine avant de l'exécuter

(toutes les erreurs de syntaxe seront affichées)

Un interpréteur traduit chaque instruction du programme en langage machine et l'exécute si possible

(Chaque erreur nécessite donc une correction)

Microsoft Office dispose d'un interpréteur VBA

Les tests

Une fois les erreurs de syntaxe corrigées, il faudra tester le programmes pour déceler et corriger d'éventuelles erreurs de logique.

Cette phase est très importante avant la mise en service d'un logiciel.

Plan du cours d’algorithmique

1. Les variables et les constantes.
2. Les conditions
3. Les boucles (ou itérations)
4. Les tableaux
5. Les procédures et les fonctions
6. Les principes de la programmation objet

1.Les variables et les constantes

- Les variables sont des objets dont le contenu peut varier pendant l'exécution du programme

- Les constantes sont des objets dont le contenu est figé pendant l'exécution du programme

1.Les variables et les constantes

Les variables et les constantes portent un nom et sont d'un type donné.

Exemples:

A variable de type numérique

Nom variable de type caractères

Trouvé variable de type Booléen

TVA constante de type numérique

1.Les variables et les constantes

Les variables et les constantes peuvent recevoir une valeur (compatible avec leur type) .

On parle alors d'affectation interne ou externe

L'Affectation Interne

On donne un valeur à une variable ou à une constante lors de l'écriture du programme.

La notation est : ←

Exemples: A ← 3

Nom ← "DUPONT"

Trouve ← Vrai

TVA ← 19,6%

L'Affectation externe

On donne la valeur à une variable lors de l'exécution du programme par l'intermédiaire d'un périphérique (clavier, souris etc.).

La notation est : ENTRER (variable)

Exemples: ENTRER(A)
ENTRER(NOM)
ENTRER(DATE)

1.Les variables et les constantes

Il est possible d'afficher le contenu d'une variable ou d'une constante .

Les notations sont :

Afficher(variable) pour l'écran

Imprimer(variable) pour l'imprimante

Exemples:

Afficher(A), Afficher(nom), Afficher("nom"),

Afficher(A+B), Afficher("A=",A)

Imprimer ("A=",A)

Exercice 1

Quel résultat produit le programme suivant ?

Variables val, doub numériques

```
Début
   Val ← 231
   Doub ← Val * 2
   Afficher ("Valeur=",Val)
   Afficher ("Double=",Doub)
Fin
```

Resultat

Valeur = 231

Double = 462

Exercice 2

Écrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

Résultat

Variables N et C numériques

Début

    Afficher("taper un nombre")
    Entrer(N)
    C ⟵ N*N (ou N^2)
    Afficher("le carré de ", N," = ", C)

Fin

La même chose en VBA

```
Public Sub test()
    Dim N, C As Single
    N = InputBox("taper un nombre")
    C = N ^ 2
    MsgBox ("Carré de " & N & " = " & C)
End Sub
```

Exécution

Exercice 3

Écrire un programme qui lit le prix HT d’un article, le nombre d’articles et le taux de TVA, et qui fournit le prix total TTC correspondant. Faire en sorte que des libellés apparaissent clairement

Résultat

Variables PHT, QT,TVA, MTTC numériques

Début
    Afficher("taper le prix Hors taxe")
    Entrer(PHT)
    Afficher("taper la quantité")
    Entrer(QT)
    Afficher("taper le taux de TVA")
    Entrer(TVA)
    MTTC ← PHT * QT * (1+TVA)
    Afficher("Le montant TTC = ", MTTC)
Fin

2. Les conditions

Les conditions permettent d'exécuter ou non certaines parties du programme.

Il existe 3 structures conditionnelles

La structure conditionnelle Simple

SI condition **ALORS**
    Instructions
FINSI

Exemple :

SI sexe="M" **ALORS**
    Afficher("Bienvenue Monsieur")
FINSI

Remarques

La condition retourne toujours un booléen.

La condition peut être simple ou composée avec les opérateurs OU, ET et NON

Exemples :

Si (A=5 **OU** B=3) **Alors**

    Afficher (A+B)

Finsi

La structure conditionnelle Simple

SI (condition) **ALORS**

Instructions..

FINSI

La structure conditionnelle Complexe

SI condition1 **ALORS**

Instructions..

SINON SI condition2 **ALORS**

Instructions..

SINON

Instructions..

FINSI

FINSI

EXEMPLE

MTTC ← **PHT*QT*(1+TVA)**

SI MTTC<=1000 **ALORS**

Remise ← 0%

SINON SI MTTC <= 2000 **ALORS**

Remise ← 5%

SINON

Remise ← 10%

FINSI

FINSI

MR ← MTTC*Remise

MNet ← MTTC-MR

La structure conditionnelle CAS

CAS variable **DE**

    valeur1: Instructions

    valeur2: Instructions

    …….

    Autre: Instructions

FinCas

Exemple

```
Début
    Afficher("Votre Choix (1 ou 2) ?")
    Entrer (ch)
    Cas ch de
        1: Afficher(" Vous avez choisi 1")
        2: Afficher(" Vous avez choisi 2")
        Autre: Afficher(" Erreur de saisie")
    FinCas
Fin
```

Exercice 4

Écrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif , négatif ou nul

Résultat

```
Variable x numérique
Début
    Afficher("Taper un nombre")
    Entrer (x)
    SI x<0 ALORS
                afficher ("Votre nombre est négatif")
            SINON SI x>0 ALORS
                        afficher ("Votre nombre est positif")
                      SINON
                        afficher ("Votre nombre est nul")
                    FINSI
    FINSI
Fin
```

Exercice 5

Écrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul).
Attention toutefois : on ne doit **pas** calculer le produit des deux nombres.

Résultat

```
Variable x,y numériques
Début
    Afficher("Taper le premier nombre différent de zéro")
    Entrer (x)
    Afficher("Taper le deuxième nombre différent de zéro ")
    Entrer (y)
    SI (x<0 ET y<0) OU (x>0 ET y>0 ) ALORS
                        Afficher ("Le résultat sera positif ")
            SINON
                        Afficher (" Le résultat sera négatif ")
    FINSI
Fin
```

Exercice 6

Écrire un algorithme qui demande trois noms à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre alphabétique.

Résultat

```
Variable n1,n2,n3   caractères

Début
    Afficher("Taper le premier nom")
    Entrer (n1)
    Afficher("Taper le deuxième nom ")
    Entrer (n2)
    Afficher("Taper le troisième nom ")
    Entrer (n3)
    SI (n3>=n2 Et n2>=n1) ALORS
            Afficher ("Les noms sont rangés dans l'ordre alphabétique ")
      SINON
            Afficher ("Les noms ne sont pas rangés dans l'ordre alphabétique ")
    FINSI
Fin
```

Exercice 7

Écrire un algorithme qui demande l’âge d’un enfant à l’utilisateur. Ensuite, il l’informe de sa catégorie :

- "Poussin" de 6 à 7 ans
- "Pupille" de 8 à 9 ans
- "Minime" de 10 à 11 ans
- "Cadet" après 12 ans

Résultat

```
Variable age   numérique

Début
    Afficher("Taper l'age de l'enfant")
    Entrer (age)
    Si (age>12) Alors
                Afficher ("Cadet ")
        Sinon  Si(age>9) Alors
                        Afficher ("Minime ")
                    Sinon  Si (age>7) Alors
                                Afficher ("Pupille")
                            Sinon
                                Afficher("Poussin")
                            FinSI
                FinSi

    FinSi
Fin
```

Quelle remarque peut-on faire sur cet algorithme ?

3. Les boucles ou itérations

Elles permettent d'exécuter plusieurs fois les mêmes instructions.

Il existe plusieurs structures itératives

La structure Répéter … Jusqu'à

Répéter

Instructions

Jusqu'à condition

Les instructions seront répétées jusqu'à ce que la condition soit vraie

(une fois au moins)

Exemple

Variable Rep **Caractère**

Début

    Repéter
        Afficher "Voulez vous un café ? (O/N)"
        Entrer (Rep)
    Jusqu'à Rep="O" OU rep="o" OU Rep="N" OU Rep="n"

    Afficher "Saisie acceptée"

Fin

La structure Tantque Condition

Tantque condition **Faire**

Instructions

FinTq

Les instructions seront répétées tant que la condition est vraie

(les instructions peuvent ne pas être exécutées)

Exemple

Variable Rep **Caractères**

Début

```
    Rep ⟵ "z"
    Tantque(Rep<>"O" ET Rep<>"o" ET Rep<>"N" ET Rep<>"n") Faire
        Afficher "Voulez vous un café ? (O/N)"
        Entrer (Rep)
    FinTq
    Afficher "Saisie acceptée"
Fin
```

La structure Pour … FinPour

Pour var←valeur initiale **à** valeur finale [**pas** P]

Instructions

FinPour

Les instructions seront exécutées un nombre fini de fois.

Exemples

```
Pour i⟵1  à  100
|            Afficher (i)
FinPour
```

```
Pour i ⟵ 0  à 100  Pas 2
|       Afficher(i)
FinPour
```

Exercice 8

Ecrire un algorithme qui demande à l’utilisateur un nombre compris entre 1 et 3 jusqu’à ce que la réponse convienne

Correction exercice 8

```
Variable N en Entier

Debut
    N ← 0
    TantQue (N < 1 ou N > 3)
        Afficher ("Entrez un nombre entre 1 et 3")
        Entrer(N)
        Si (N < 1 ou N > 3) Alors
            Afficher( "Saisie erronée.Recommencez")
        FinSi
    FinTantQue
Fin
```

Exercice 9

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10

Correction exercice 9

```
Variable N en Entier
    Debut
    |  N ← 0
    |  Afficher( "Entrez un nombre entre 10 et 20")
    |  TantQue (N < 10 ou N > 20)
    |  |  Entrer (N)
    |  |  Si N < 10 Alors
    |  |  |                          Afficher ( "Plus grand !")
    |  |  |                  Sinon Si N > 20 Alors
    |  |  |                          |       Afficher( "Plus petit !")
    |  |  |                          FinSi
    |  |  FinSi
    |  FinTantQue
    Fin
```

Exercice 10

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27

Correction exercice 10

```
Variables N, i en Entier
   Debut
   |  Afficher ("Entrez un nombre : ")
   |  Entrer (N)
   |  Afficher ("Les 10 nombres suivants sont : ")
   |  Pour i ← N + 1 à N + 10
   |  |          Afficher (i)
   |  Fin Pour
   Fin
```

Exercice 11

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
...
7 x 10 = 70

Correction exercice 11

```
Variables N, i en Entier
   Debut
   |  Afficher ("Entrez un nombre : ")
   |  Entrer (N)
   |  Afficher ("Table de",N)
   |  Pour i ← 1 à 10
   |  |        Afficher (N, " x ", i, " = ", n*i)
   |  FinPour
   Fin
```

Exercice 12

Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

$$1 + 2 + 3 + 4 + 5 = 15$$

- NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Correction exercice 12

```
Variables N, i, Som en Entier
   Debut
   |  Afficher ("Entrez un nombre : ")
   |  Entrer (N)
   |  Som ← 0
   |  Pour i ← 1 à N
   |  |        Som ← Som + i
   |  FinPour
   |  Afficher "La somme est : ", Som
   Fin
```

Exercice 13

Ecrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle.

NB : la factorielle de 8, notée 8 !, vaut

$$1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8$$

Correction exercice 13

```
Variables N, i, F en Entier
    Debut
        Afficher ("Entrez un nombre : ")
        Entrer (N)
        F ← 1
        Pour i ← 2 à N
            F ← F * i
        FinPour
        Afficher ("La factorielle de ",N," = ",F )
    Fin
```

Exercice 14

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres :

Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
etc.
Entrez le nombre numéro 20 : 6

Le plus grand de ces nombres est : 14

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

C'était le nombre numéro 2

Correction exercice 14 V.1

```
Variables N, i, PG en Entier
    Debut
        PG ← 0
        Pour i ← 1 à 20
            Afficher ("Entrez un nombre : ")
            Entrer (N)
            Si i = 1 ou N > PG Alors
                        PG ← N
            FinSi
        FinPour
        Afficher ("Le nombre le plus grand était : ", PG)
    Fin
```

Correction exercice 14 V.2

```
Variables N, i, PG, IPG en Entier
    Debut
    |   PG ← 0
    |   Pour i ← 1 à 20
    |   |       Afficher ("Entrez un nombre : ")
    |   |       Entrer (N)
    |   |       Si i = 1 ou N > PG Alors
    |   |       |               PG ← N
    |   |       |               IPG ← i
    |   |       FinSi
    |   FinPour
    |   Afficher ("Le nombre le plus grand était : ", PG)
    |   Afficher ("Il a été saisi en position numéro ", IPG)
    Fin
```

Exercice 15

Réécrire l’algorithme précédent, mais cette fois-ci on ne connaît pas d’avance combien l’utilisateur souhaite saisir de nombres. La saisie des nombres s’arrête lorsque l’utilisateur entre un zéro.

Correction exercice 15

```
Variables N, i, PG, IPG en Entier
    Debut
    |   N ← 1
    |   i ← 0
    |   PG ← 0
    |   TantQue N <> 0
    |   |           Afficher ("Entrez un nombre : ")
    |   |           Entrer (N)
    |   |           i ← i + 1
    |   |          Si i = 1 ou N > PG Alors
    |   |           |                          PG ← N
    |   |           |                          IPG ← i
    |   |          FinSi
    |    FinTantQue
    |    Afficher ("Le nombre le plus grand était : ", PG)
    |    Afficher ("Il a été saisi en position numéro ", IPG)
    Fin
```

Exercice 16

Lire la suite des prix (en euros entiers et terminée par zéro) des achats d’un client. Calculer la somme qu’il doit, lire la somme qu’il paye, et simuler la remise de la monnaie en affichant les textes "10 Euros", "5 Euros" et "1 Euro" autant de fois qu’il y a de coupures de chaque sorte à rendre.

```
Variables FF, somdue, M, IPG, Reste, Nb10F, Nb5F En Entier
    Debut
    |   E ← 1
        somdue ← 0
        TantQue E <> 0
        |               Afficher ("Entrez le montant : ")
        |               Entrer (E)
        |               somdue ← somdue + E
        FinTantQue
        Afficher ("Vous devez :", E, " euros")
        Afficher ("Montant versé :")
        Entrer (M)
        Reste ← M - E
        Nb10E ← 0
        TantQue Reste >= 10
        |               Nb10E ← Nb10E + 1
        |               Reste ← Reste – 10
        FinTantQue
        Nb5E ← 0
        Si Reste >= 5
        |       Nb5E ← 1
        |       Reste ← Reste – 5
        FinSi
        Afficher ("Rendu de la monnaie :")
        Afficher (" "Billets de 10 E : ", Nb10E)
        Afficher (" "Billets de  5 E : ", Nb5E)
        Afficher (" "Pièces de 1 E : ", reste)
    Fin
```

Exercice 17

Écrire un algorithme qui permette de connaître ses chances de gagner au tiercé, quarté, quinté et autres impôts volontaires.

On demande à l'utilisateur le nombre de chevaux partants, et le nombre de chevaux joués. Les deux messages affichés devront être :

Dans l'ordre : une chance sur X de gagner

Dans le désordre : une chance sur Y de gagner

X et Y nous sont donnés par la formule suivante, si n est le nombre de chevaux partants et p le nombre de chevaux joués (on rappelle que le signe ! signifie "factorielle", comme dans l'exercice précédent) :

X = n ! / (n - p) !
Y = n ! / (p ! * (n – p) !)

NB : cet algorithme peut être écrit d'une manière simple, mais relativement peu performante. Ses performances peuvent être singulièrement augmentées par une petite astuce. Vous commencerez par écrire la manière la plus simple, puis vous identifierez le problème, et écrirez une deuxième version permettant de le résoudre.

Correction exercice 17

4. Les tableaux

Les variables ne peuvent contenir **qu'une seule valeur** à un instant donné.

Si l'on veut entrer 50 notes il faudra alors 50 variables (**ce qui est long**) ou utiliser un tableau (**ce qui simplifie le travail**)

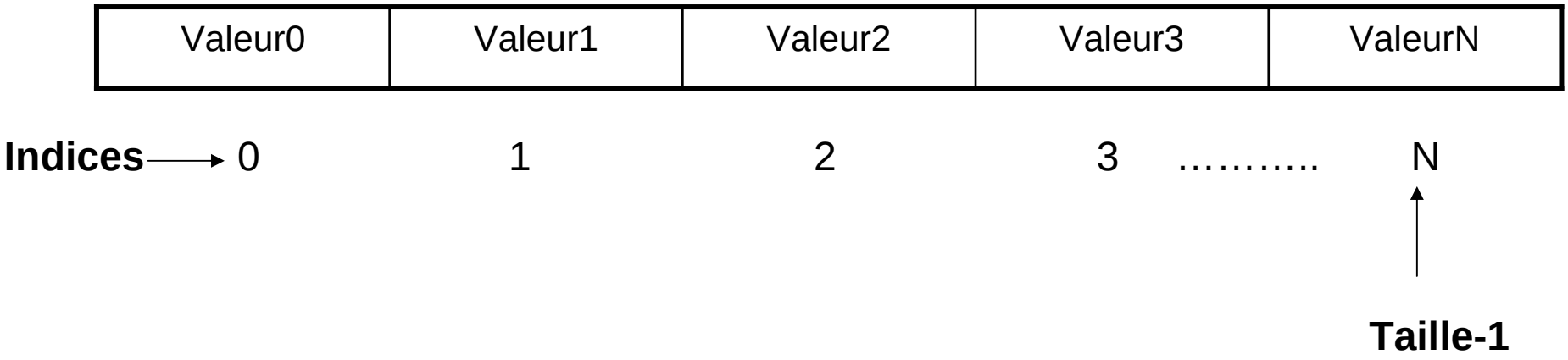
Definition d'un tableau

Un tableau est une structure pouvant contenir un grand nombre d'éléments du même type à un instant donné.

On utilise en général des tableaux à **une** ou **deux** dimensions .

Les tableaux à une dimension

Les tableaux portent un **nom** et peuvent contenir **plusieurs** valeurs .
- Représentation d'un tableau T



- Signification:

T(0)← valeur0; T(1)← valeur1;…… T(N)←ValeurN

Exercice 18

Soit T représenté par:

12	11	8	20	4	11	9	18	12	7

- Quelle est la taille du tableau ?
- Quelle est la valeur de T(5)?
- Quelle est la plus grande valeur de T ?
- Quelle est la plus petite valeur de T ?
- Quelle est la moyenne des valeurs de T ?

Exercice 19

Écrire un algorithme permettant de remplir le tableau précédent (on suppose que le remplissage se fait à partir de l'indice 1)

Correction

```
Var i numérique, T tableau de numériques taille 10
Constante n =10 ;
Debut
    n⟵10
    Pour i ⟵ 1 jusqu'à n
        Afficher("Entrer la valeur",i)
        Entrer(T(i))
    FinPour
Fin
```

Exercice 20

Écrire l'algorithme permettant de déterminer la plus petite valeur contenu dans le tableau précédent.

Modifier cette algorithme pour faire afficher l'indice du tableau ou se trouve la plus dernière plus petite valeur

Modifier cette algorithme pour faire afficher les indices du tableau ou se trouve la plus petite valeur

Correction V.1

```
Var i,min numériques, T tableau de numériques taille 10
Constante n =10 ;
Debut
    n ⟵ 10
    min ⟵ T(1)
    Pour i⟵2 jusqu'à n
        Si (T(i)<min )  alors min⟵T(i)
        FinSI
    FinPour
    Afficher(min)
Fin
```

Correction V.2

```
Var i,min,ind numériques, T tableau de numériques taille 10
Constante n =10 ;
Debut
    n ← 10
    min ← T(1)
    Pour i ← 2 jusqu'à n
        Si (T(i)<min )  alors
                        min ← T(i)
                        ind ← i
        FinSI
    FinPour
    Afficher(min," à l'indice :",ind)
Fin
```

Correction V.3

```
Var i,min,j numériques:    T, R tableaux de numériques taille 10
Constante n =10 ;
Debut
   n ⟵ 10
   min⟵ T(1)
   j ⟵ 1
   R(j ) ⟵  1
   Pour i⟵1 jusqu'à n
      Si (T(i)<min )  alors
                  min⟵ T(i)
                  j ⟵1
                  R(j)⟵ i
         SINON SI (T(i)=min) alors
                  j=j+1
                  R(j)⟵ i
                  FinSi
      FinSi
   FinPour
   Pour i⟵1   jusqu'à j
      afficher ( R(i) )
   FinPour
Fin
```

Exercice 21

Écrire l'algorithme qui affiche la moyenne des valeurs contenues dans le tableau T

Correction

```
Var i,som,moy numériques, T tableau de numériques taille 10
Constante n =10 ;
Debut
    n ← 10
    som ← t(1)
    Pour i ← 1 jusqu'à n
        som ← som+T(i)
    FinPour
    moy ← som/n
    Afficher("Moyenne =",moy)
Fin
```

Les tableaux à deux dimensions

Ils peuvent eux aussi contenir plusieurs valeurs mais Ils sont composés de plusieurs lignes et de plusieurs colonnes comme les feuilles Excel

Représentation

	1	2	3	4
1	Gap	05000	35000	800
2	Nice	06000	700000	20
3	Cannes	06400	200000	15
N	……	…….	……..	……..

Notation: T (Ligne , Colonne)

T(1,0) = "Nice"; T(2,3)=15 etc…

Exercice 22

a)Écrire l'algorithme permettant de remplir le tableau précédent(on supposera N=30).

On n'utilise pas la ligne 0 et la colonne 0

b)Ecrire l'algorithme permettant d'entrer une ville et d'afficher sa population

c)Ecrire l'algorithme permettant d'entrer une altitude et d'afficher le nombre de villes ayant une altitude supérieure

Correction a)V.1avec une Simple boucle

```
Var i numérique, T tableau taille 30
Constante N =30 ;
Debut
   Pour i←1  jusqu'à N
      Afficher("Saisir une ville"): Entrer(T(I,1))
      Afficher("Saisir son code postal"): Entrer(T(I,2))
      Afficher("Saisir sa population"): Entrer(T(I,3))
      Afficher("Saisir son altitude"): Entrer(T(I,4))
   FinPour
Fin
```

Correction a) V.2 avec une double boucle

```
Var i,j numériques, T tableau taille 30
Constante N =30 ; C=4
Debut
   Pour i ← 1  jusqu'à N
      Pour j ←1  jusqu'à C
         Cas j de :
               1: Afficher("Saisir une ville")
               2: Afficher("Saisir son code postal")
               3: Afficher("Saisir sa population")
               4: Afficher("Saisir son altitude")
          FinCas
          Entrer(T(i,j))
      FinPour
   FinPour
Fin
```

Correction b) V.1 avec boucle For

```
Var i numérique, T tableau taille 30, v caractères
Constante N =30 ;
Debut
   Afficher( "saisir le nom de la ville")
   Entrer (v)
   Pour i⟵ 1  jusqu'à N
         Si (T(i,1)=v) Alors afficher T(i,3)
         Finsi

   FinPour
Fin
```

Quels sont les problèmes de cette solution?

Correction b) V.2 avec boucle Tantque

```
Var i numérique, T tableau taille 30, v caractères
Constante N =30 ;
Debut
  Afficher( "saisir le nom de la ville")
  Entrer (v)
  i ⟵ 1
  Tanque (T(i,1) <>v et i<N)
  |     i⟵i+1
  FinTq
  Si (T(i,1)=v) Alors Afficher T(i,3)
  |             Sinon
  |                   Afficher( " Cette ville n'existe pas")
  FinSi
Fin
```

Comment faire si le nombre de lignes dans le tableau n'est pas connu ?

Correction c)

```
Var i , a, cpt numériques, T tableau taille 30
Constante N =30 ;

Debut
   cpt⟵ 0
   Afficher( "saisir l'altitude")
   Entrer (a)
   Pour i⟵ 1  jusqu'à  N
      Si ( T(i,4) > a ) Alors cpt⟵ cpt+1
      FinSi

   FinPour

   Afficher("Il existe ", cpt , "villes dont l'altitude est supérieure à ", a," mètres")
Fin
```

5.Les fonctions et les procédures

Parfois certaines séquences algorithmique peuvent être réutilisées dans d'autres programmes.

Il est alors intéressant de les enregistrer dans des fonctions ou procédures.

Tous les langages de programmation évolués proposent des fonctions

Exemple en VBA: len(ch), left(ch,2), Ucase(ch)

V.1 Les procédures

Une procédure un algorithme qui effectue toujours les mêmes instructions.

Déclarer une procédure consiste à lui donner un nom , à définir les éventuels paramètres formels en entrée, sortie et entrée-sortie.

Pour exécuter une procédure il suffit de taper son nom et de préciser la valeur des paramètres effectifs.

Exemple de procédure

Procédure AfficheAge(Entrée:Année:numérique,Sortie:Age:numérique)

Début

Age←AnnéeSystème – Année

Fin

Remarque: Année et Age sont des paramètres formels

a, b variables numériques

Debut

```
    Afficher( "entrer votre année de naissance")
    Entrer (a)
    AfficheAge(a,b)
    Afficher("Vous avez ", b , "Ans")
```

Fin

Remarque: a et b sont des paramètres effectifs

Exemple de procédure

```
Procedure Cube(Entrée-Sortie: c:numérique)
   Début
   | c ←— c^3
   Fin

v variable numérique
Début
|       Afficher(" taper une valeur ")
|       Entrer ( v )
|       Cube ( v );
|       Afficher(" Le cube de votre valeur est :", v)
Fin
```

V.2 Les fonctions

Une fonction un algorithme qui effectue toujours les mêmes instructions et qui retourne toujours un résultat.

Déclarer une fonction consiste à lui donner un nom , à définir les paramètres en entrée et le type du résultat

Pour exécuter une fonction il suffit de taper son nom et de préciser la valeur des paramètres.

Exemple de fonction

```
Fonction surfaceCercle(Entrée:rayon:numérique):numérique
    s variable locale numérique
    Début
    |   s←3,14 * rayon^2
    |   retourner s
    Fin
```

La fonction s'appelle surfaceCercle, rayon est paramètre en entrée .
Elle retourne un résultat de type numérique

```
R, Surf variables numériques  ( R et surf sont des variables globales )
Début
|       Afficher(" saisir le rayon du cercle ")
|       Entrer ( R )
|       Surf←  surfaceCercle(R)
|       Afficher (la surface du cercle est :", Surf)
Fin
```

Exercice 23

Écrire la fonction

Mention(Entrée:moy:numérique) :Caractères

Si moy < 10 la fonction retourne "Non Admis"
Si 10<=moy <12 la fonction retourne "Passable"
Si 12<=moy<14 la fonction retourne "Assez bien "
Si 14<=moy<16 la fonction retourne "bien"
Si 16<=moy la fonction retourne "Très bien "

voir

Correction

```
Fonction Mention(Entrée:moy numérique) : Caractères
    m  variable locale caractères
    Début
    |       Si moy < 10 Alors m← "Non Admis "
    |       |   Sinon SI moy < 12 Alors m←  " Passable"
    |       |           |   Sinon Si moy <14 Alors m ← "Assez Bien"
    |       |           |           |   Sinon Si moy <16 Alors m ←  "Bien"
    |       |           |           |           |   Sinon m ←  "Très bien"
    |       |           |           |           FinSi
    |       |           |           FinSi
    |       |           FinSi
    |       FinSI
    |       Retourner ( m )
    Fin
```

Conlusion sur les procédures et fonctions

Il est intéressant de décomposer un problème en procédures et/ou fonctions pour différentes raisons.

-Faciliter la lisibilité du code et sa maintenance

-Accélérer les développements futurs

Les fonctions et procédures souvent utilisées seront stockées dans des **bibliothèques**

6. Les principes de la programmation orientée objet

Pour illustrer ce concept nous utiliserons le langage de modélisation UML

Unified **M**odeling **L**anguage

Qui est devenu un standard de modélisation

Introduction

La programmation orientée objet vise à améliorer la modularité des applications en utilisant des abstractions plus évoluées que les fonctions ou les procédures

Une seule entité logique (la classe) peut regrouper un ensemble de données (les attributs) et un ensemble de traitements (les méthodes)

L'objectif de la POO est de faciliter la réutilisation du code

Classes et objets

Le concept central de la POO est la **classe** qui est composée d'attributs et de méthodes.

- Les attributs servent à définir les états de la classe

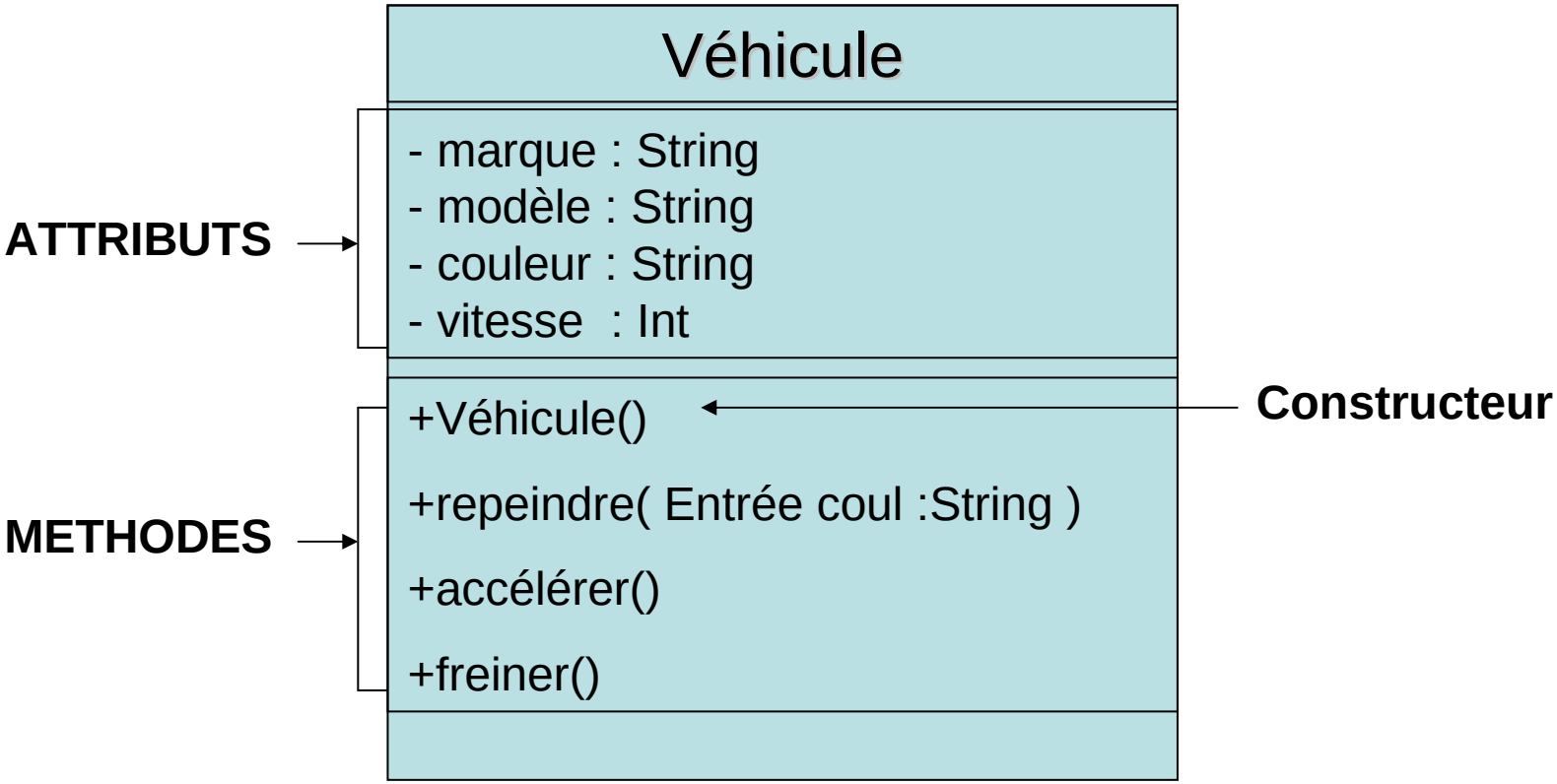
- Les méthodes servent à définir les comportements de la classe

Remarques:

- Le mot-clé *class* est utilisé par la plupart des LOO (Langage Orienté Objet) pour introduire les classes

- Un package est un ensemble de classes ayant une structure arborescente et concernant le même sujet. (Un package réservation contiendra par exemple une classe Client, une classe réservation et une classe participant)

Exemple : Modélisation de la classe Véhicule extrait d'un diagramme de classes



Les classes possèdent au moins une méthode appelée constructeur permettant de construire ses instances (objets)

Instances et objets

Une classe est un concept abstrait et structurel qui ne peut être utilisée directement.

Les applications travaillent sur des *instances* ou *objets* correspondant à des occurrences de la classe.

Les objets sont créés par l'intermédiaire du mot-clé *new*

Exemple d'utilisation de la classe

maVoiture = **new** Véhicule("Nissan","Xtrail","Grise",0)

maVoiture.repeindre("rouge")

maVoiture.accelerer()

On peut dire pour simplifier que maVoiture est de type Véhicule

A l'intérieur d'une méthode, le mot-clé *this* permet à l'objet de s'autoréférencer :

exemple this.couleur = coul ;

Encapsulation

L'encapsulation permet d'interdire l'accès à certains éléments (attributs ou méthodes) d'une classe afin de protéger ses états et fonctionnements internes.

En général les attributs de la classe ne doivent pas être exposés directement à l'extérieur de la classe. La mise en œuvre d'accesseurs et mutateurs est conseillée

Exemple d'accesseur et mutateur

Ce sont des méthodes qui permettent d'accéder aux valeurs des attributs ou de modifier leur contenu

getCouleur( ) { return this.couleur }

setCouleur(c) { this.couleur = c }

La visibilité

Afin de mettre en œuvre l'encapsulation, la plupart des LOO permettent de contrôler la visibilité des attributs et des méthodes.

Il existe trois niveaux de visibilité pour les éléments d'une classe:

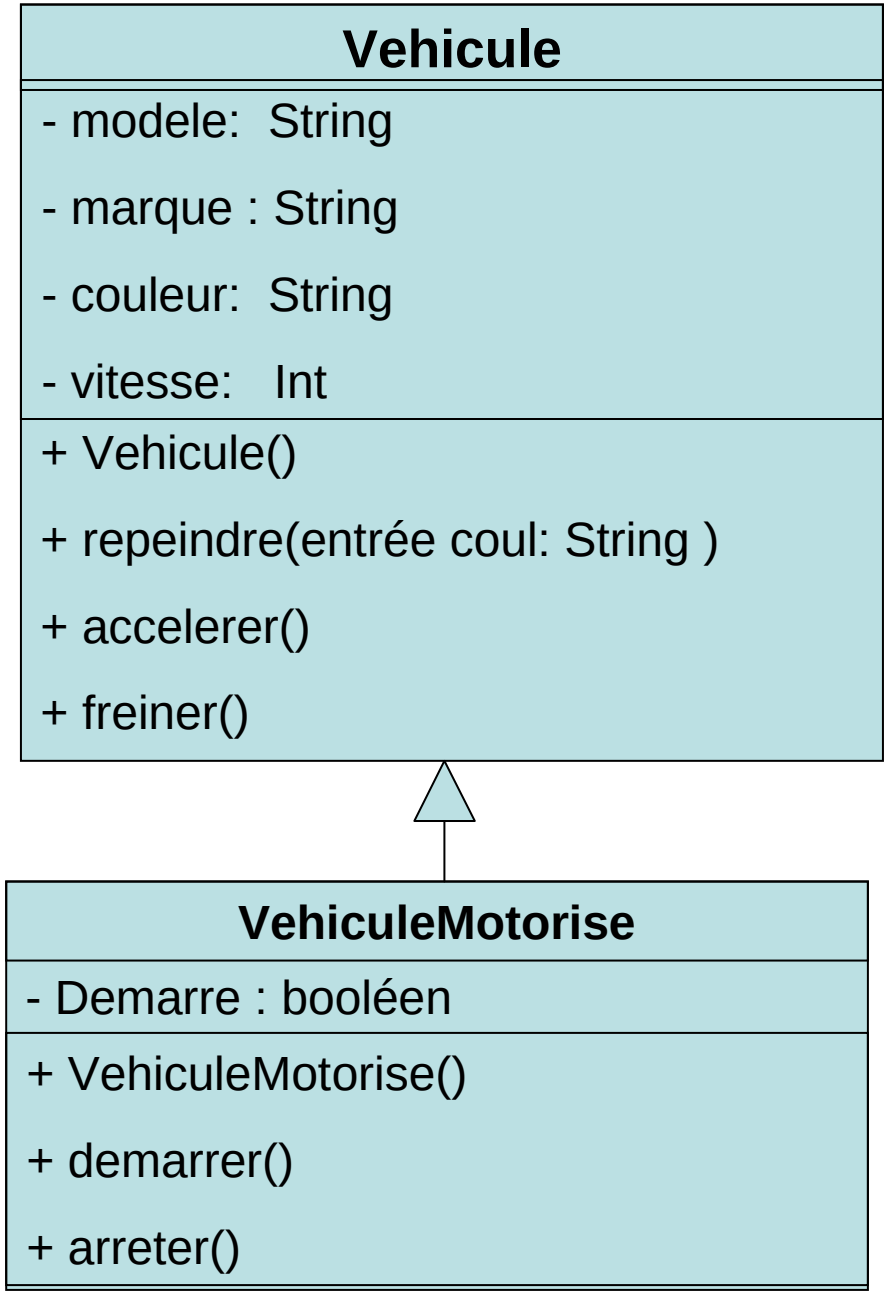
- Privé (-): Les entités externes n'ont pas accès aux éléments
- Protégé(#): Seules les sous-classes ont accès aux éléments
- Public (+) : Les entités externes ont accès aux éléments.

L'héritage

Le mécanisme d'héritage, permet de définir des sous-classes afin d'enrichir et de bénéficier des éléments de classes existantes.

Le mot-clé extends est souvent utilisé pour créer des relations d'héritage

Modélisation UML de l'héritage



Héritage multiple

Certains LOO permettent de faire hériter une classe de plusieurs autres (C++, pas java)

Le principal problème peut provenir de l'existence de méthodes ou attributs similaires dans les parents d'une classe.

Classes et méthodes abstraites

Une classe abstraite est une classe qui ne peut pas être instanciée dans une application.

Elle ne peut être utilisée que par les classes qui l'étendent

Ces classes définissent une ou plusieurs méthodes abstraites dont le code n'est pas défini dans la classe. Ces dernières peuvent cependant être utilisées dans les méthodes concrètes de la classe. Les classes filles concrètes ont la responsabilité d'implémenter ces méthodes abstraites.

Le mot-clé *abstract* est souvent utilisé pour ces classes et méthodes

Exemple de classe abstraites

Supposons que la classe Vehicule soit abstraite ainsi que ses méthodes accelerer() et freiner().

Les sous-classes Voiture et Moto seront obligées d'implémenter ces méthodes mais la méthode concrète roule() de la classe Véhicule pourra les utiliser.

Interface

Une interface est une classe spéciale où toutes les méthodes sont abstraites

Chacune de ses sous-classes devra les implémenter

Exemple :L'interface Telephone peut contenir les méthodes abstraites numéroter(),ecouter(),parler(),sonner()

Surcharge

La surcharge est le concept qui autorise deux méthodes à avoir le même nom , mais avec des paramètres différents ou de type différents.

L'intérêt est de réduire le nombre des méthodes

surface (c: integer) { return x^2}

surface (lo: integer, la: integer) {return long*larg}

surface (r:double) {return 3.14*r^2}

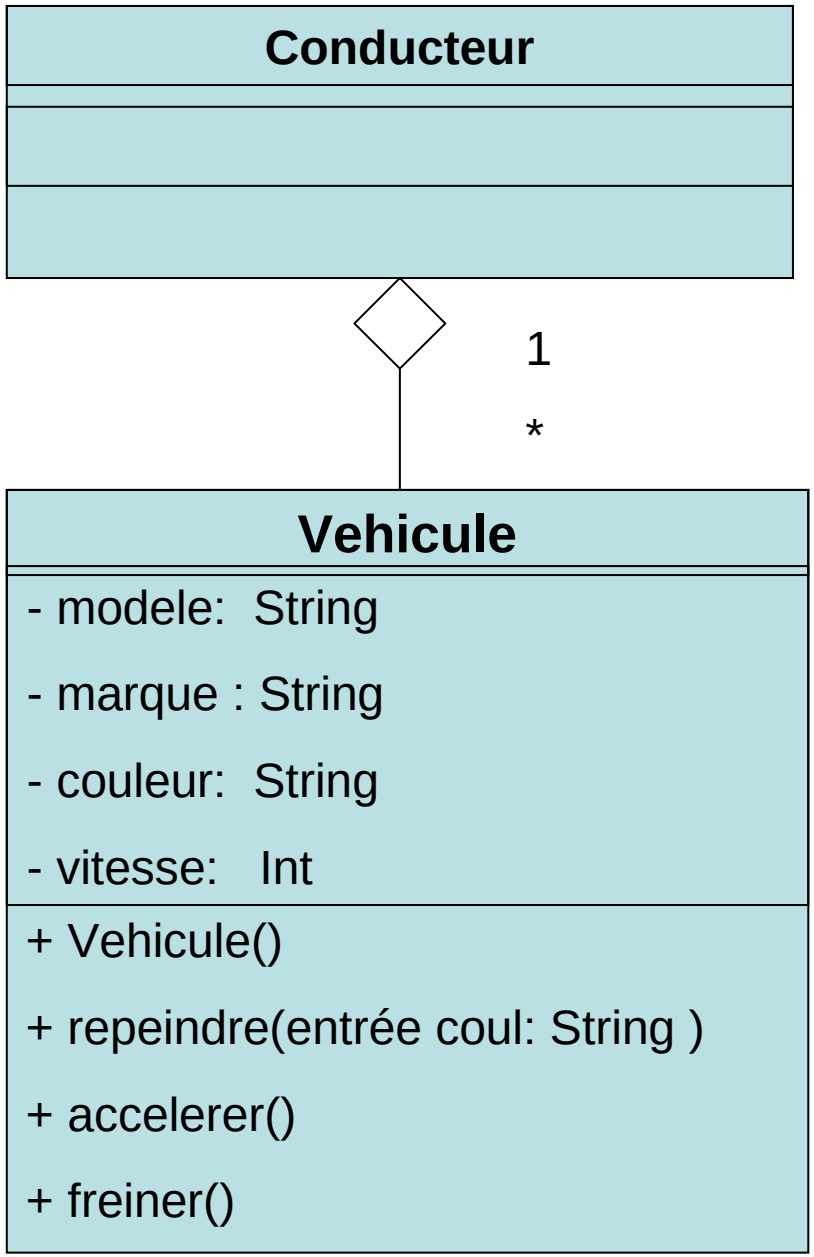
Classes et méthodes finales

Une classe finale est une classe qui ne peut être étendue donc au dernier niveau de l'héritage

Une méthode finale est une méthode qui ne peut plus être surchargée

Les LOO utilisent souvent le mot-clé final

Représentation UML d'une agrégation



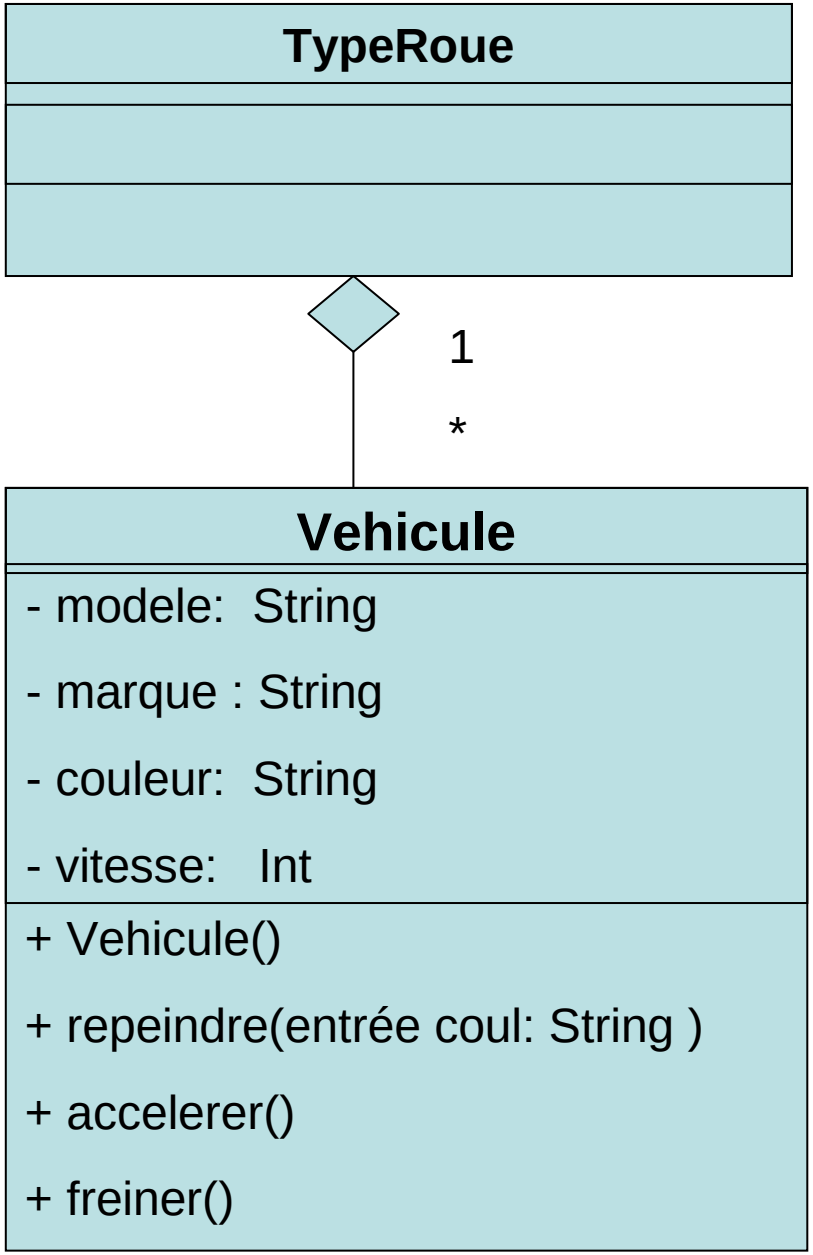
Agrégation

Lors d'une agrégation, un attribut d'une classe est du type d'une autre classe

La classe Vehicule sera enrichie par un attribut conducteurPrincipal de type Conducteur

Si la multiplicité est *, l'attribut correspondra à une collection (conducteur principal et secondaires)

Représentation UML d'une Composition



Composition

La composition correspond à une agrégation forte. Si l'objet contenant est détruit, les objets contenus le sont aussi, contrairement à l'agrégation.

6.Le langage de programmationVBA

Visual **B**asic pour **A**pplications est le langage de programmation des applications de Microsoft Office (Word,Excel,Access etc..)

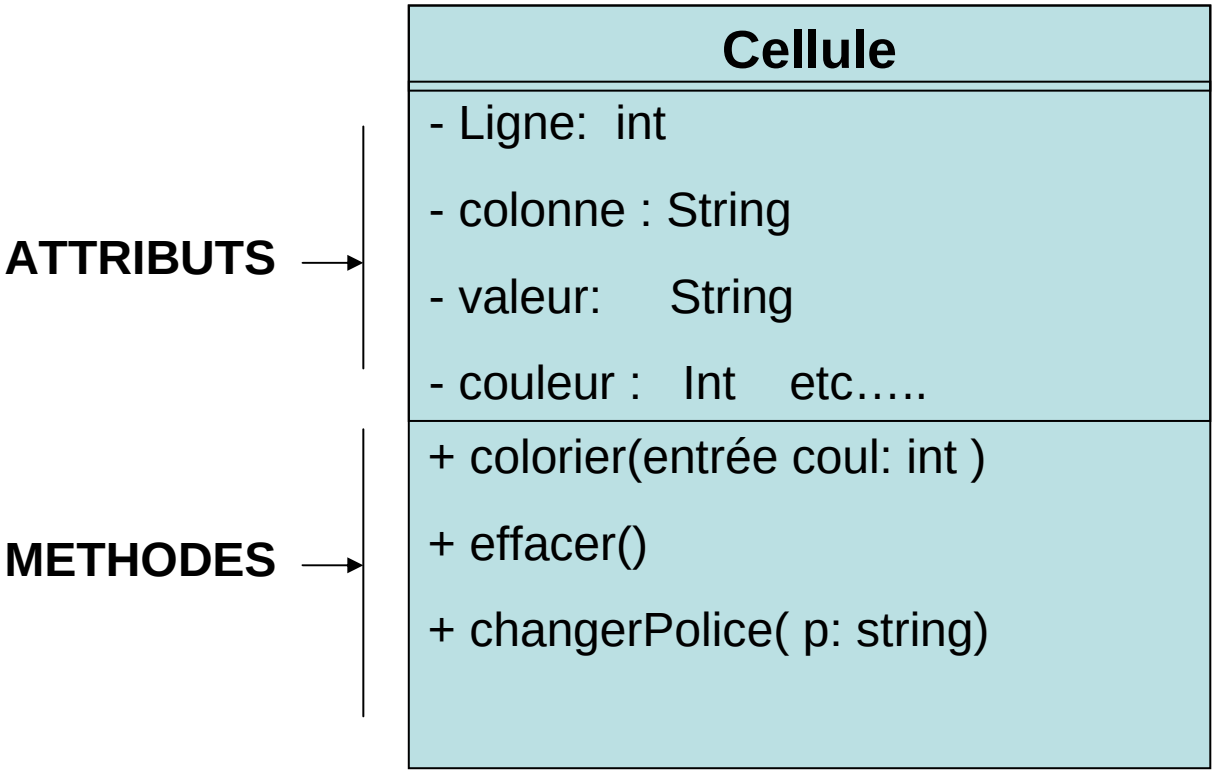
Il permet d’automatiser les tâches et de créer des applications complètes.

VBA est un langage orienté objet

Il met à notre disposition, en plus des structures algorithmiques, des objets caractérisés par des propriétés et des méthodes.

Définition d'un objet

Un objet est une entité composée d' attributs et de méthodes



Les principaux objets d'EXCEL

- Les classeurs : WorkBooks

- Les feuilles : WorkSheets

- Les cellules : Cells ou Range

Les principales méthodes (verbes)

Elles permettent d'agir sur les objets: **Objet.méthode**

Exemples:

- Workbooks.close
- Workbooks("classeur.xls").close
- Workbooks("classeur.xls").Worksheets("Feuil1").delete
- Worksheets.Add
- Workbooks("classeur.xls").activate
- Worksheets("Feuil1").Range("A1").clearContents
- ActiveWorkbook.close(false)
- Worksheets("Feuil1").Range("A1:B20").select

Les principales propriétés (noms)

Elles sont les caractéristiques des objets: **Objet.propriété**

Exemples:

- Worksheets("stat").name="Stat01"
- Worksheets("Stat01").Range("A2").value=3
- Worksheets("Stat01").Range("A2").formula = "=A1+B2"
- Worksheets("Stat01").Range("A2").Font.Bold=true
- Selection.Font.Bold=true
- ActiveCell.value= 35
- ActiveSheet.name="Client"

Les instructions de base du VBA

Déclaration des variables:

Dim NomVar [as Type]

Exemple: Dim I AS Integer

Déclaration des constantes:

Const NomConst = Valeur

Exemple: Const TVA = 19,6%

Les principaux types

- Byte de 0 à 255
- Integer de -32768 à 32767
- Long de -2 147 483 648 à 2 147 483 647
- Boolean true ou false
- Single réel simple précision
- Double réel double précision
- Date date
- String() Chaîne de caractères

L'affectation

Interne :

NomVar = valeur

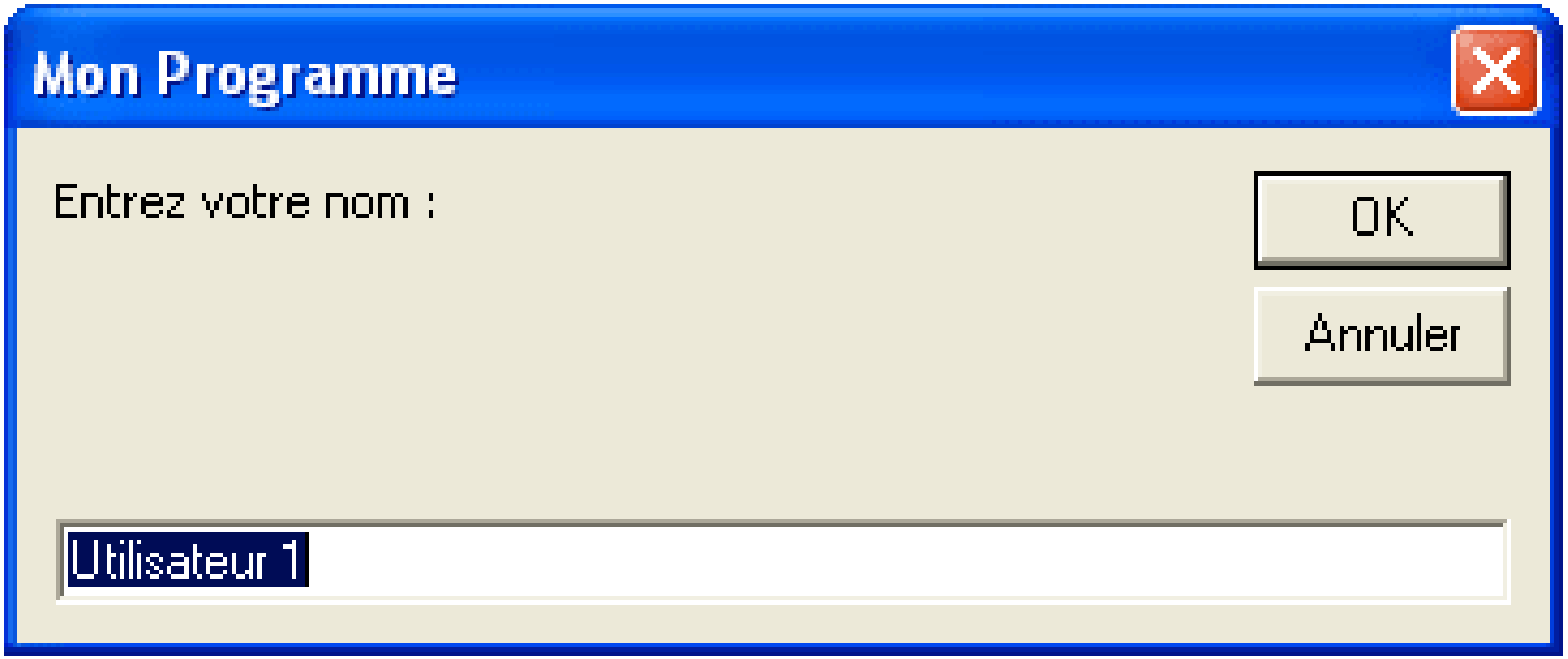
Exemples : X=3 ; nom="Dupont" ; trouve=false ; s=s+1

Externe:

Nomvar = InputBox(message[,titre][,défaut][,xpos][,ypos])

Exemple d'affectation externe

Message = InputBox("Entrez votre nom :", "Mon Programme", "Utilisateur 1")



Affichages

Il existe deux possibilités:

- Dans une cellule

```
Worksheets("Stat01").Range("A2").value = X
Worksheets("Stat01").Range("A2").value = "X ="&X
```

- Dans une boite de dialogue

```
MsgBox message[,bouton][,titre]
```

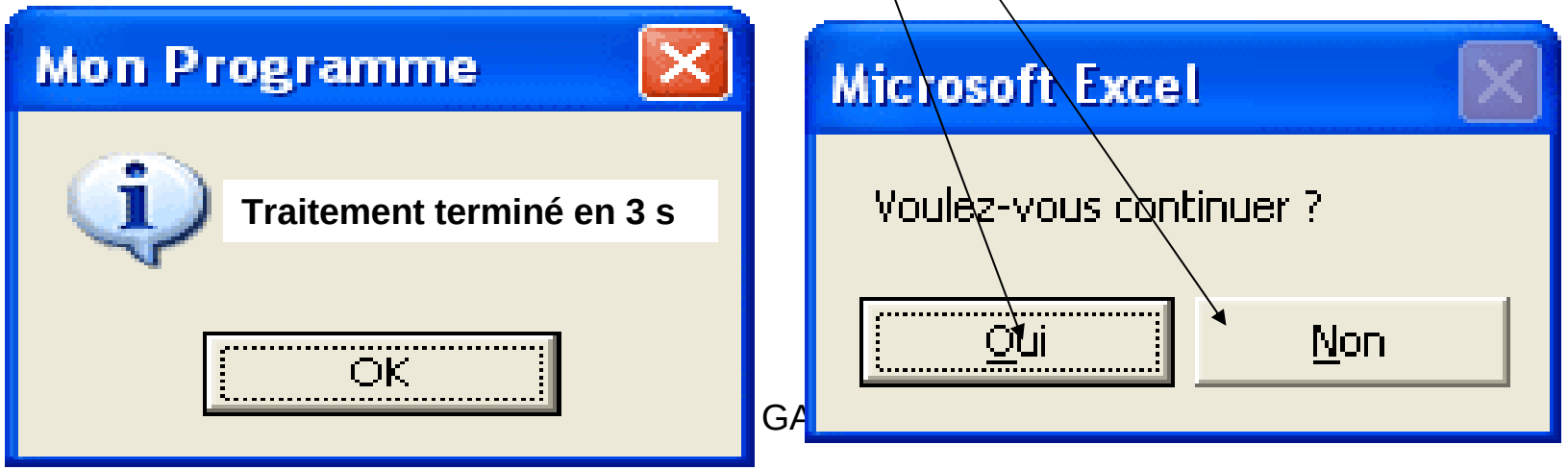
Exemple MsgBox

- Message : valeur à afficher
- Bouton : puissance de 2 entre 0 et 32
- Titre : titre de la boite

MsgBox "Traitement terminé en " & x & "s", 64, "Mon Programme"

Rep = MsgBox "Voulez-vous continuer ?", 1

Rep = 1 ou 2



Les structures conditionnelles

1) La structure de base:

If condition **Then**
actions…
Else
actions..
End If

Les structures conditionnelles

2) La structure Select Case

```
X =inputBox (" Saisir la moyenne " )

Select Case X
    Case 1 To 10    : MsgBox "Insuffisant"
    Case 11 To 19   : MsgBox " Bien"
    Case 20         : MsgBox " Excellent"
    Case Else       : MsgBox " Erreur"
End Select
```

Les structures itératives

Il en existe quatre:

Do While…….Loop

Do Until ……..Loop

For……..……Next

For Each……..Next

Les structures itératives

La structure **Do While:**

```
Mp="***"
Do while Mp <>"qszery09"
    Mp = inputBox("votre mot de passe")
Loop
```

Les structures itératives

La structure **Do Until:**

```
Mp="***"
Do Until Mp = "qszery09"
    Mp = inputBox("votre mot de passe")
Loop
```

Les structures itératives

La structure **For :**

```
For i = 1 to 10 Step 2
        MsgBox i
Next i
```

Les structures itératives

La structure **For Each:**
Elle permet de parcourir un ensemble de cellules

```
Range("A1:B10").Select
For Each C In Selection
        If C.Value < 10 Then
                C.Interior.ColorIndex = 3
        End If
Next
```

Présentation de VBE

Visual **B**asic **E**ditor

C'est un éditeur de texte permettant:

- De saisir des programmes
- De les exécuter
- De les déboguer
- D'avoir une aide en ligne.