1. Java, les bases

1.1. Introduction

1.1.1. Un bref historique

- origine : 1991 - ingénieur de SUN qui ont cherché à concevoir un langage applicable à de petits appareils électriques (code embraqué). L'idée est de traduire un programme source, non pas en langage machine mais en pseudo langage universel disposant des fonctionnalités communes à toutes les machines. Ensuite ce code intermédiaire, appelé byte code est compact et portable. Il suffit alors que la machine dispose d'un programme approprie appelé machine virtuelle permettant d'interpréter le code intermédiaire sur la machine concernée. Le projet de code embarqué n'a pas abouti en tant que tel.
- Ces concepts ont été repris en 1995 pour la réalisation du logiciel HotJava, navigateur web écrit par SUN en java capable d'exécuter des applets (application) écrites en byte code
- D'autres navigateurs web ont suivi et contribué à l'essor du langage : J2SE, JDK (engouement constant)
- Modèle objet simple mais puissant avec une API (bibliothèque standard)
- Langage généraliste pédagogique et utilisé dans les entreprises

1.1.2. Java et la programmation orientée objet (POO)

Objectif de la programmation orientée objet est de contribuer à :

- La fiabilité
- La réutilisation

Elle introduit de nouveaux concepts tels que : l'objet, l'encapsulation, les classes et l'héritage.

Les concepts

- POO :
  Le code est construit non pas autour de ses fonctionnalités mais autour de l'objet
- Concept d'encapsulation : (il n'est pas possible d'agir directement sur les données d'un objet, il est nécessaire de passer par une interface), abstraction des données
- Concept de classe : généralisation de la notion de type
- Concept d'héritage
- Concept de polymorphisme : on traite de la même manière des objets de types différents

1.2. Une première application

1.2.1. Un exemple simple

```
public class Exemple
{
    public static void main(String [] args)
    {
        int n;
        double x;

        n=5;
        x=2*n+1.5;

        System.out.println("n = "+n);
        System.out.println("x = "+x);

        double    y;
        y = n * x+12;
        System.out.println("y = "+y);

    }

}
```

1.2.2. Structure générale d'un programme

- Bloc : ensemble d'instructions délimitées par {}

- 3 formes de commentaires :

  /** commentaire */ forme de commentaire pouvant être exploitée par le générateur javadoc
  /* commentaire */
  // Commentaire jusqu'à la fin de ligne

- Un programme Java est une collection de classe qui décrit des objets qui seront manipulés à l'exécution
  - Structure d'un programme autour des objets
  - Une classe est constituée
    - De variable : désignant des données qui sont des valeurs de tout type
    - De méthode : action, cad fonction ou procédure

API (application programming interface) : classes définies par ailleurs et mises à la disposition du programmeur
Les classes unies par une même sémantique sont regroupées dans des paquetages

- La méthode main :

    - L'exécution d'un programme java commence par l'exécution de la méthode main
    - En-tête toujours identique
    - Public : droit d'accès, obligatoire pour que le programme puisse s'exécuter
    - Args :
        - Tableau de chaine de caractère
        - obligatoire permet de passer des arguments
    - static : précise que la méthode n'est pas liée à une instance (un objet) de la classe
- La casse doit être respectée

Java est un langage typé

1.2.3. Exécution d'un programme

- Saisir et sauvegarder le programme. Le nom du fichier doit impérativement être celui de la classe publique avec l'extension .java.
  Ex : Exemple.java
- Compilation : produit non pas du code source mais du code intermédiaire (bytecode). Si la compilation s'est bien déroulée alors on obtient un fichier avec l'extension .class
  Ex : Exemple.class
- Le code est ensuite lance par l'intermédiaire de la machine virtuelle Java

1.3. Un aperçu général de Java

1.3.1. Les types primitifs de Java

- Type entier :
    - byte 1 octet -128 ;+127
    - short 2 -32768 ;+32767
    - int 4
    - long 8
- Type flottant
    - float 4
    - double 8
- Type caractère

- char : variable de type caractère
- ‘a’ ‘E’ : constantes de type caractère
- Par convention ;
  - \b retour arrière
  - \t tabulation
  - \n saut de ligne
  - \f saut de page
  - \r cariage return
  - \ » «
  - \ ‘ ‘
  - \\ \

- Type booleen boolean

1.3.2. Les variables

Def 1 :
Une variable est une case mémoire permettant de stocker des données

Def 2 :
On appelle identificateur le nom désignant une variable

- Le langage Java fait la différence entre majuscule et minuscule
- Un identificateur est formé de lettres ou de chiffres, _ et $ le premier caractère doit être une lettre ou _ et $
  Rque : il est conseille de ne pas utiliser $ qui intervient dans les mécanismes internes
- Les variables doivent être déclarées avant d’être utilisées
- Les variables peuvent être initialisées lors de leur déclaration

  Ex :

```
public class Var1
{
        public static void main (String args[])
        {
                int n=15

                int k=10 ;
                int p = 2*n ;

                int j ;
                j=Clavier.lireInt() ;
```

```
                int jj=2*j ;

        }
}
```

- Cas des variables non initialisées :
  En java, une variable n'étant pas initialisée ne peut pas être utilisée :

```
Ex
public class Var2
{
        public static void main (String args[])
        {
                int     n ;
                System.out.println (« n= « +n) ; //erreur de
                compilation

                int p=n+3 ; //erreur de compilation

        }
}
```

1.3.3. Les constantes

- Le mot clef final indique que la valeur d'une variable ne doit pas être modifiée pendant l'exécution du programme

```
Ex
public class Cons
{
        public static void main (String args[])
        {
                final int n=20 ;
        n=n+5 ; //erreur n est constante
        n=Clavier.lireInt() ; //erreur

        int p ;
        p=Clavier.lireInt ;  //Ok bien que la valeur ne soit connue qu'a
        l'exécution

        }
}
```

- Rque : L'initialisation d'une variable final peut être différée.

1.3.4. Les opérateurs

Les opérateurs ne sont définis que pour des opérandes de même type sauf par le jeu des conversions.

- Les operateurs arithmétiques :
  - operateurs binaires (portant sur deux opérandes)
    + - * /
  - operateur unaire :
    - (-n) +

  Les règles de priorités des operateurs sont celles de l'algèbre traditionnel (Comportement en cas d'exception)

- Les opérateurs relationnels

<	Inferieur à
<=	inferieur ou égal
>	supérieur
>=	supérieur ou égal
==	égal
!=	différent de

Ca particulier des valeurs infinity et nan

- Les operateurs de manipulation de bits

&	et
|	ou inclusif
^	ou exclusif
<<	décalage à gauche
>>	décalage à droite (avec propagation du bit de signe)
>>>	décalage logique à droite
~	complément à 1

Table de vérité :

opérande 1	0	0	1	1
opérande 2	0	1	0	1
et &	0	0	0	1
ou inclusif |	0	1	1	1
ou exclusif^	0	1	1	0

Exemple

n	01101110
p	10110011
n&p	00100010
n|p	11111111
n^p	11011101
~n	10010001
n << 2	10111000

n >> 3 00001101
p >> 3 11110110
p >>> 3 00010111

- Les operateurs logiques
  - ! négation
  - & et
  - ^ ou exclusif
  - | ou inclusif

  rque : && (et), || (ou inclusif) sont des operateurs de court circuit. La seconde opérande n'est évaluée que si on a besoin de connaitre sa valeur pour décider si l'expression est vraie ou fausse.

  oper
- L'operateur d'affectation :
  - =
  - variable = variable operateur expression variable operateur = expression
    +=, -= *= /= ^= &= <<= >>= >>>=
- Les operateurs de décrémentation et incrémentation
  - pré-incrémentation ++y
  - post-incrémentation y++
  - pré-décrémentation --y
  - post-décrémentation y--

Exercice : Soient x et y deux flottants, quelles sont les valeurs de x et de y ?

```
public class Incrementation
{
    public static void main(String [] args)
    {
      float   x,y=2;


      y++;   /* y=3*/

      x=y++;/* x=3 et y=4 */

      x=++y;/* y=5 et x=5 */
      }
}
```

- Les conversions
  - Les conversions implicites vont du plus petit vers le plus grand :
    int long float double

Ces conversion sont appelées conversion d'ajustement.

- Les operateurs numériques ne sont pas définis pour les types byte, char et short. Java prévoit que toute valeur d'un de ces types est convertie en int. On parle alors de promotion numérique ou encore de conversion systématique
- Les conversions par affectation

  byte short int long float double
  char int long float double
- operateur de cast :

  (type) variable

1.4. Les tableaux

1.4.1. Déclaration et création de tableaux

Les tableaux de données permettent de stocker des variables multidimmensionnées

Ex : vecteur, matrice….

Déclaration

Syntaxe :
TypeT tab[] ; TypeT [] tab ;

Création

L'operateur new permet d'allouer l'emplacement d'un tableau

Syntaxe :

tab[] = new TypeT[size] ou size est la taille du tableau

Remarque : La valeur de l'expression fournie par new est calculée au moment de l'execution et non de la compilation. Elle peut donc etre variable
Exemple :

```
int n = Clavier.lireInt() ;
int t[] = new int[n] ;
```

On peut aussi créer un tableau en l'initialisant :

```
int n,p ;
…..
int [] = {1,n,n+p,2*p, 12}
```

On a alors cree un tableau de 5 entiers

1.4.2. Utilisation de tableaux

Accès aux éléments d'un tableau.

Syntaxe :

t[i] : permet d'accéder au i-1eme élément du tableau
Le premier élément du tableau correspond a l'indice 0

Exemple :

```
int t[]=new int[5] ;
….
t[0] = 15 ;
…
t[2] ++ ;
…
System.out.println(t[4]) ;
…..
```

Taille d'un tableau

Syntaxe :

t.length permet de connaitre le nombre d'elements d'un tableau

Exemple :

```
int t[] = new int[5] ;

System.out.println(« taille de t : «  + t.length ; //affiche 5
```

1.5. Quelques instructions de base

Entrées/sorties

Pour lire des donnees au clavier, on utilisera la classe clavier.

Exemple :

```
public class Lire
{
        public static void main (String args[])
        {
                int nb ;
                nb = Clavier.lireInt() ;

                double x ;
                nb = Clavier.lireDouble() ;
```

```
        }
}
```