Université de Nice – Sophia Antipolis — Deug MASS MIAS SM 1re année
UEF1 : Informatique & Programmation — 2000–2001

TP N° 2

Types de données numériques – Variables

Ça va mieux en le révisant ... Les données manipulées en JAVA sont divisées en deux catégories bien distinctes : les *données primitives* (nombres, caractères, booléens) et les *objets*. Les nombres sont eux-mêmes divisés en plusieurs types, suivant le nombre d'octets sur lesquels ils sont codés dans la mémoire de l'ordinateur.
Une *variable* est un emplacement mémoire destiné à recevoir une valeur de tel ou tel type. La taille de l'emplacement dépend du type en question. Le type d'une variable est fixé lors de sa *déclaration* : cette opération obligatoire a lieu une seule fois. Lors de la déclaration, on peut associer une valeur à la variable ; on parle alors de *définition.* En toute généralité, la première fois que l'on associe une valeur à une variable se nomme l'*initialisation.*
Le procédé qui permet de donner une valeur à une variable est l'*affectation,* symbolisé par le signe `=`, qui n'a pas forcément le même sens qu'en mathématiques[1].
Veillez à ce que toute variable apparaissant dans la partie droite d'une affectation ait déjà été initialisée !
Le principe d'*ajustement de type* permet de forcer le type d'une valeur ; seul un nombre restreint d'ajustements est possible. Les ajustements s'accompagnent le cas échéant d'une perte d'information comme par exemple dans le cas d'un `double` vers un `int`.

Dorénavant, nous vous conseillons de procéder de la manière suivante : faites un seul projet par TP (aujourd'hui `TP2.jpr`). A l'intérieur d'un projet se trouveront plusieurs classes correspondant aux divers exercices. Parmi ces classes, il en sera tour à tour choisie une comme la classe principale avec `Set as main`. Le plus simple serait même de nommer `TP2Exo4` la classe à écrire dans l'exercice 4 du TP2 de façon à vous y retrouver vite lors de vos révisions...

1 Représentation des nombres

Voici les règles de priorité qui régissent les opérateurs arithmétiques :

Les + prioritaires	`* / %`
Les - prioritaires	`+ -`

A priorité égale (et en l'absence de parenthèses), les opérateurs sont lus de gauche à droite.

1. Sachez distinguer en mathématiques les différentes significations du signe $=$. Dans $ax^2 + b = y + 3$, dans $(x^2 = 1 \Rightarrow x = 1$ ou $x = -1)$ ou bien dans $soit\ r = \sqrt{x}$...
Seul le dernier $=$ peut se lire *reçoit*, comme pour l'affectation en JAVA .

Exercice 1) Quel est le résultat de l'expression arithmétique suivante :

```
4 / 2 + 3 - 9 % 2 * 5
```

Vous aurez compris que si l'ordinateur s'y retrouve, l'être humain lui préfère les expressions avec des parenthèses. Ecrivez un programme afin qu'il affiche lui-même le résultat du calcul de cette expression, sans aucune parenthèse puis avec toutes les parenthèses nécessaires à la compréhension. Vous devez bien évidemment trouver le même résultat si vos parenthèses sont correctement placées. ◇

Exercice 2) Il est dit plus haut qu'à priorité égale, les opérations étaient faites de la gauche vers la droite. En vous souvenant que la signification de l'opération `+` dépend du contexte, prévoyez `sur papier` le résultat des affichages suivants, puis véifiez-le en JAVA :

```
System.out.println("1 + 2 + foo");
System.out.println(1 + 2 + "foo");
System.out.println(1 + (2 + "foo"));
System.out.println(1 + "foo" + 2);
System.out.println("foo" + 1 + 2);
```

◇

Exercice 3) Réalisez un programme qui calcule la surface et le volume d'une sphère de rayon 2.

$$S = 4\pi R^2$$
$$V = 4/3\pi R^3$$

Affichez les résultats de la façon suivante :

```
Sphère de rayon R=2
Surface : S=...
Volume : V=...
```

Les réponses sont de l'ordre de $S = 50.2$ et $V = 33.5$. ◇

Exercice 4) a. Ecrivez un programme qui calcule la moyenne de 3 notes lors de l'affichage : par exemple un 13,5 en mathématiques, un 18 en informatique et un 12,5 en électronique.
b. Recommencez l'exercice avec les notes respectives 11, 14, 17. Vérifiez ces résultats (en lançant éventuellement la calculatrice WINDOWS à l'écran). Votre dernier résultat est-il juste ? Pourquoi ? Comment réparer cette erreur dans le programme ? ◇

Exercice 5) Débordement a. Un entier signé de type `int` est mémorisé sur 4 octets (1 octet = 8 bits). Si le type `int` ne contenait que des nombres positifs ou nuls, quel serait l'intervalle contenant tous ces nombres : $[0, ?]$.
b. Bien entendu, il y a aussi des nombres négatifs. Du coup, quelle est la plage de variation du type `int` ?
c. Vérifiez en faisant afficher les constantes `Integer.MIN_VALUE` et `Integer.MAX_VALUE`

qui leur correspondent[2] en JAVA .
d. A votre avis, que se passe-t-il si l'on ajoute 1 à `Integer.MAX_VALUE` ? Vérifiez-le en programmant. ◇

2 Utilisation de variables

Exercice 6) Tapez le programme suivant :

```
class TP2Exo6
{  public static void main(String[] args)
   {  int a = 3, b = 7;
      System.out.println("AVANT : a vaut " + a " et b vaut " + b);
      a = b;
      b = a;
      System.out.println("APRES : a vaut " + a " et b vaut " + b);
   }
}
```

Ce programme a-t-il l'effet escompté ? Le cas écheant, modifiez-le.
Une fois qu'il donne satisfaction – qu'il *échange* bien les valeurs de `a` et `b` – modifiez le type de `b` en `double` et donnez-lui la valeur `7.5`. Que se passe-t-il ? ◇

Remarque : Avant de nommer une variable, pensez à lui donner un nom *parlant*. Il devra être relativement évocateur sans être *cryptique*[3]. Il se composera par exemple d'une lettre, d'une ou plusieurs initiales ou bien d'un ou plusieurs mots (ex : `x`, `résultat`, `nombreLignes`, ...). Le nom d'une variable commence par une lettre minuscule.

Exercice 7) Trouvez un algorithme qui à partir d'un entier correspondant à un nombre de secondes, le convertit en heures/minutes/secondes. Ecrivez le programme JAVA complet correspondant à cet algorithme.
Pour ce faire, vous utiliserez au moins trois variables correspondant respectivement aux nombres d'heures, de minutes et de secondes recherchés. Voici un exemple d'exécution :

```
Le nombre de secondes choisi est: 4567 s.
Cela équivaut à 1 h 16 mn 7 s.
```

Ajoutez un commentaire en haut du programme précédent. Il devra contenir le nom de l'auteur, les numéros du T.P. du jour et de l'exercice, ainsi qu'une explication concise de l'objet du programme. ◇

2. Ces constantes appartiennent à la classe `Integer`, dont nous ne parlerons pas plus et dont vous n'aurez pas besoin ce semestre.

3. Peu clair, difficile à comprendre pour le lecteur.

3 Approfondissement

Exercice 8) On trouve dans la classe Math une fonction `Math.random()` sans argument, qui retourne un nombre de type `double`, choisi aléatoirement (c'est-à-dire au hasard) dans l'intervalle $[0, 1[$.
a. Testez-la.
b. Utilisez cette fonction pour tirer un nombre (entier !) de secondes au hasard (par exemple moins de 10000) et affichez ensuite les nombres d'heures, minutes, secondes équivalents.
Indication : on pourra utiliser astucieusement un ajustement de type. ◇

Vous aurez compris qu'un langage de programmation ne manipule qu'un sous-ensemble fini des ensembles mathématiques $\mathbb{Z}$ et $\mathbb{R}$.
Les *nombres approchés* tiennent leur nom de leur représentation : un `double` est représenté en machine par un bit de signe, une *mantisse m* sur 52 bits et un *exposant e* sur 11 bits[4]. Un calcul sur de telles valeurs, avant de s'opérer, les *réduit* au même exposant, ce qui peut entraîner une perte de précision quand les nombres ne sont pas du même ordre de grandeur.

Attention donc aux erreurs de calcul !! Jugez plutôt :

Exercice 9) Saisissez le programme suivant :

```
class TP2Exo7
{   public static void main (String[] args)
    {   double a = 100000000;
        double b = a * a + 1;
        double c = a * a;
        double d = b - c;
        System.out.println(d);
    }
}
```

Exécutez ce programme. Le résultat est-il juste ? Avez-vous compris la raison de ce comportement ?
Le tout est d'être prévenu : tous les langages de programmation connaissent une limite pour ce qui est du calcul des nombres approchés. ◇

4. Voir la première feuille d'exercices du module *Méthodologie*. On y trouve un schéma du cas du type `float` qui offre moins de précision que notre type `double`.