Département d'Informatique
DUT Informatique S1
Année 2014/2015
Algorithmique et programmation

Algorithmique et programmation

TP 6 : Le langage Java, les fonctions

1 Les fonctions

Les fonctions permettent de décomposer un programme en plusieurs parties. Le comportement d'une fonction est paramétré par un ensemble de variables : les paramètres de la fonction. À la fin de son exécution, une fonction retourne généralement un résultat : sa valeur de retour. Les fonctions permettent de faciliter l'écriture et la (re-)lecture des programmes. Elles permettent également de réutiliser facilement des morceaux de programmes.

Une fonction est définie par :

- un *nom* ;
- une *liste des paramètres* : c'est une liste de variables typées dont la valeur sera renseignée lors de l'utilisation de la fonction ;
- un *type de retour* : c'est le type de la valeur retournée par la fonction à la fin de son exécution ;
- le *corps de la fonction* : c'est le code Java décrivant ce que fait la fonction.

Remarque En Java, les fonctions sont aussi appelées des *méthodes*. Dans le cadre de ce cours d'introduction, nous utiliseront essentiellement des méthodes dites *de classe* (mot clef **static**).

1.1 Définition d'une fonction

En Java, la définition d'une fonction a la forme suivante :

```
static type_de_retour nom_de_la_fonction(liste des paramètres) {
    // corps de la fonction
}
```

Les fonctions doivent être définies en dehors de toute autre fonction (il n'est pas possible de définir une fonction dans une autre).

Type de retour Le type de retour de la fonction est le type de la valeur retournée par la fonction. Tous les types usuels (**int**, **float**, **boolean**, etc.) sont possibles. Dans le cas particulier où la fonction ne retourne aucune valeur, on utilisera le mot clef **void** à la place du type de retour.

Nom de la fonction Le choix des noms de fonctions suivent les mêmes règles que les noms de variables : le nom doit commencer par une lettre ou un tiret bas (_), et ensuite être composé d'une suite de lettres, de chiffres ou de tirets bas. Il est très fortement conseillé de choisir des *noms significatifs*, c.-à-d. donnant une indication sur ce que fait la fonction.

Liste des paramètres Les paramètres sont des variables dont la valeur est renseignée avant d'utiliser la fonction. Ces variables sont données dans la liste des paramètres. Chaque paramètre est défini par son type, suivi de son nom. Lorsqu'une fonction prend plusieurs paramètres, leurs définitions sont séparées par des virgules. Lorsqu'une fonction ne prend aucun paramètre, le nom de la fonction est simplement suivi par une paire de parenthèses vides ().
NB : les variables définies par les paramètres n'existent que pendant l'exécution de la fonction. Elles sont créées au début de l'exécution de la fonction et disparaissent lorsque la fonction retourne.

Corps de la fonction Le corps de la fonction est ensuite décrit dans un bloc délimité par des accolades {}. Le corps de la fonction consiste en une suite d'instructions Java. Comme dans tout bloc d'instructions, il est possible de déclarer des variables qui n'existeront que le temps de l'exécution du bloc.

L'instruction **return** provoque l'arrêt immédiat de l'exécution de la fonction. Ce mot clef doit être suivi d'une expression dont l'évaluation donnera la valeur retournée par la fonction. Le type de cette expression doit bien évidemment correspondre au type de retour déclaré pour la fonction. Si la fonction ne retourne rien (type de retour **void**), le mot clef **return** n'est suivi de rien (hormis le point-virgule terminant l'instruction).

Une instruction **return**; est implicite à la fin de chaque fonction. Il est donc possible de l'omettre pour des fonctions ne retournant rien.

Exemples Voici quelques exemples de fonctions :

```
// Fonction ne prenant aucun paramètre et ne retournant aucune valeur
static void hello() {
    System.out.println("Hello world!");
}

// Fonction ne prenant aucun paramètre et retournant la valeur du
// nombre d'or
static double nombreDOr() {
    return (1.0 + Math.sqrt(5.0)) / 2.0;
}

// Fonction prenant un paramètre réel et retournant le carré du
// paramètre
static double carre(double x) {
    return x * x;
}

// Fonction retournant la valeur absolue de son paramètre entier
static int abs(int x) {
    if (x < 0)
        return -x;
    else
        return x;
}

// Fonction prenant un horaires sous la forme de trois entier :
// heures, minutes, secondes et retournant cet horaire converti en
// secondes
static int hms2s(int h, int m, int s) {
    int m_total = m + h * 60;
    int s_total = s + m_total * 60;
    return s_total;
}
```

```
// Fonction calculant le P.G.C.D. de deux nombres entiers positifs
// en utilisant l'algorithme d'Euclide
static int pgcd(int a, int b) {
    if (a < b) {
        // échanger les valeurs de a et b
        int c = a;
        a = b;
        b = c;
    }
    int r = a % b;
    while (r != 0) {
        a = b;
        b = r;
        r = a % b;
    }
    return b;
}
```

1.2 Appel de fonction

Lorsqu'une fonction est utilisée, on parle *d'appel de fonction*. Pour appeler une fonction, il suffit d'écrire son nom suivi d'une liste d'expressions entre parenthèses, séparées par des virgules. Ces expressions donneront les valeurs que prendront les paramètres de la fonction pendant son exécution. Une fonction retournant une valeur peut être utilisée au sein d'une expression.

Quelques exemples avec les fonctions définies précédemment :

```
public static void main(String[] args) {
    hello();              // utilisation de la fonction hello()

    int x, y;
    System.out.print("Entrer deux nombres : ");
    x = input.nextInt();
    y = input.nextInt();
    int z = pgcd(x, y); // utilisation de la fonction pgcd()
    System.out.println("Le pgcd de " + x + " et " + y + " est " + z);

    double phi2 = carre(nombreDOr()); // utilisation des fonctions carre() et nombreDOr();
    System.out.println("phi^2 = " + phi2);

    int a;
    System.out.print("Entrer un nombre (a) : ");
    a = input.nextInt();
    System.out.println("a + |a| = " + (a + abs(a))); // fonction abs()
}
```

En Java, les paramètres sont toujours passés *par copie*. Cela signifie que la fonction reçoit une copie des valeurs passées en paramètre.

Il n'y a pas, en Java, de moyen direct d'implémenter les paramètres **out** et **in-out** du langage algorithmique. Ils peuvent cependant être émulés en passant des références vers des objets (lorsque cette notion aura été apprise).

2 Quelques conseils

Voici une liste de conseils afin d'utiliser correctement les fonctions :

– Le nom de la fonction doit être choisi soigneusement : il doit refléter ce que fait la fonction.

– De même que le nom de la fonction, c'est une bonne idée de donner des noms significatifs aux paramètres des fonctions.
– Il est conseillé d'essayer de ne mettre qu'une seule instruction **return** dans une fonction. Il faut parfois retravailler l'algorithme pour y arriver. Ceci n'est cependant pas une règle absolue : l'objectif est d'écrire du code qui soit clair et facile à lire et à comprendre.
– Le corps d'une fonction ne doit bien souvent pas dépasser quelques (2 ou 3) dizaines de lignes. Si une fonction devient trop longue, c'est peut-être qu'il faut la décomposer en plusieurs fonctions.
– On veillera à indiquer avant chaque fonction, à l'aide d'un commentaire, le rôle de la fonction, une description des différents paramètres ainsi que de la valeur retournée.

3 Exercices

3.1 Une forêt de sapins

En reprenant de qui a été fait pour le TP4, écrire une fonction affichant un sapin dont la hauteur est donnée en paramètre. Utiliser cette fonction pour afficher 12 sapins dont la hauteur est tirée au hasard entre 3 et 8.

3.2 Tabulation

Définir une fonction prenant en paramètre un angle θ exprimé en degrés et retournant la valeur de $\sin\theta \div \cos\theta$. Utiliser ensuite cette fonction pour imprimer la valeur retournée par la fonction, en donnant en paramètre les valeurs successives $0, 10, 20, 30, \ldots, 350$.

On peut utiliser les fonctions prédéfinies **double** Math.sin(**double**) et **double** Math.cos(**double**) pour calculer respectivement le sinus et le cosinus d'un angle. Attention : l'angle doit alors être donné en radians.

3.3 L'algorithme mystère

Soit l'algorithme suivant :

Lexique des variables

a	(entier)	DONNÉE
b	(entier)	DONNÉE
c	(entier)	RÉSULTAT
d	(entier)	INTERMÉDIAIRE
e	(entier)	INTERMÉDIAIRE

Algorithme

```
d ← b
e ← a
c ← 1
tant que d > 0 faire
    si d mod 2 = 0 alors
        d ← d/2
    sinon
        d ← (d − 1)/2
        c ← c × e
    fsi
    e ← e × e
ftant
```

Écrire une fonction implémentant cet algorithme. Les données a et b de l'algorithme deviennent alors les paramètres de la fonction.

Afficher les différentes valeurs de la fonction en faisant varier le paramètre a entre 1 et 10, et le paramètre b entre 0 et 5.

Pouvez-vous déduire ce que calcule cet algorithme dans le cas général ?

3.4 Table de vérité

Dans l'exercice qui suit, il est évidemment attendu que vous subdivisiez votre programme à l'aide de fonctions judicieusement choisies !

Écrire un programme affichant la table de vérité d'une expression booléenne à 3 variables (A, B et C). Modifiez ensuite votre programme pour afficher les tables de vérités des expressions suivantes.

$$(\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$$
$$(A \wedge \neg B) \vee (B \wedge \neg C) \vee (\neg B \wedge C)$$

Que pouvez-vous en conclure ?