TP 4 : Méthodes

Informatique Fondamentale (IF121)

15 novembre 2004

L'objectif de ce TP est d'apprendre à se servir des méthodes (parfois aussi nommées « fonctions »). Comme les TP précédents, il fera l'objet de plusieurs séances.

Il est vivement recommandé de sauvegarder vos programmes sur disquette afin de pouvoir les réutiliser lors des séances prochaines.

1 Rappel de syntaxe

Pour **définir une méthode** :

```
static <type> <nom de la méthode> (<type du paramètre> <nom du paramètre>) {
   <Instruction>;
   <Instruction>;
   ...
}
```

Exemple :

```
static double carré (double x) {
   return x*x;
}
```

Pour **faire appel à une méthode** : `<nom de la méthode>(<une valeur correspondante>)`

Exemples : `carré(5)`, `Deug.length(str)`, ...

Les deux programmes suivants sont équivalents :

```
import fr.jussieu.script.Deug;
class Carre {
   static double carré (double x) {
      return x*x;
   }

   public static void main (String[] args) {
      Deug.print("Donnez un nombre réel : ");
      double y = Deug.readDouble();


      Deug.println("Son carré est égal à "
         + carré(y));
   }
}
```

```
import fr.jussieu.script.Deug;
class Carre {




   public static void main (String[] args) {
      Deug.print("Donnez un nombre réel : ");
      double y = Deug.readDouble();
      double x = y;
      double resultat = x*x;
      Deug.println("Son carré est égal à "
         + resultat);
   }
}
```

Une méthode peut

- prendre plusieurs paramètres
  - de même type : `double max (double x, double y)`
    $\Longrightarrow$ `max(4.0, 3.0)`
  - de types différents : `Deug.subString (String s, int début, int fin)`
    $\Longrightarrow$ `Deug.subString("Bonjour", 3, 7)`
- ne pas prendre de paramètres : `Deug.readInt()`
- retourner une valeur : *tous les exemples ci-dessus*
- ne pas retourner de valeur :

```
static void afficherEnGros (String str) {
   Deug.println("************************");
   Deug.println("* " + str + " *");
   Deug.println("************************");
}
```

2 Lecture d'un programme

Exercice 1 : *Variantes*

Pour chacun des programmes suivants, indiquer (de tête) ce qu'il affiche. Attention, un programme incorrect s'est glissé dans le lot.

```
import fr.jussieu.script.Deug;
class P1 {
    static int f (int x) {
        x = x + 2;
        return x;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(x));
    }
}
```

```
import fr.jussieu.script.Deug;
class P2 {
    static int f (int y) {
        y = y + 2;
        return y;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(x));
    }
}
```

```
import fr.jussieu.script.Deug;
class P3 {
    static int f (int y) {
        int x;
        x = x + 2;
        return x;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(x));
    }
}
```

```
import fr.jussieu.script.Deug;
class P4 {
    static int f (int y) {
        int x = y + 2;
        return y;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(x));
    }
}
```

```
import fr.jussieu.script.Deug;
class P5 {
    static int f (int x) {
        x = x + 2;
        return x;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(x) + f(x));
    }
}
```

```
import fr.jussieu.script.Deug;
class P6 {
    static int f (int x) {
        x = x + 2;
        return x;
    }
    public static void main (String[] args) {
        int x = 5;
        Deug.println(f(f(x)));
    }
}
```

Exercice 2 : *Correct, mais pas bon*

Que pensez vous du bout de code suivant ?

```
static double produit (double x,double y) {
    double x = Deug.readDouble();
    double y = Deug.readDouble();
    double prod = x * y;
    return prod;
}
```

Exercice 3 : *Même exercice*

```
class test{
    static double produit (double x,double y) {
        double prod = x * y;
        return prod;
    }
    public static void main (String[] args){
        double x = Deug.readDouble();
        double z = Deug.readDouble();
        double y = produit(x,z);
        Deug.println(y);
    }
}
```

3 Exercices de base

Remarque. En TP, lorsqu'un exercice demande d'« écrire une méthode », il faudra aussi la tester en l'incluant dans un programme complet qui l'appelle avec des arguments appropriés.

Exercice 4 : *Maximum*

1. Écrire une méthode `max` qui calcule le maximum de deux nombres.
2. Grâce à `max`, écrire une méthode `max3` qui calcule le maximum de trois nombres, sans utiliser de `if`.

Exercice 5 : *À la lettre*

Il est possible de comparer les caractères. Les caractères sont ordonnés. Les majuscules sont classées par ordre alphabétique. Il en est de même pour les minuscules. Ainsi, l'expression `'a'<'b'` est évaluée à `true` alors que l'expression `'R'<='E'` est évaluée à `false`.

À l'aide des éléments ci-dessus, écrire des méthodes qui renvoient

1. `true` si la lettre transmise en paramètre est une majuscule et `false` si c'est une miniscule (on supposera que la méthode est toujours appelée correctement) ;
2. `true` si un caractère donné c est une lettre (miniscule ou majuscule).

4 Les jours de la semaine

N'oubliez pas de *tester* vos méthodes au fur et à mesure !

Exercice 6 : *Nom du jour dans la semaine*

Écrire une méthode `nomDuJour` qui prend comme paramètre un nombre de 0 à 6 et qui renvoie le jour de la semaine correspondant sous forme d'une chaîne de caractères. On fera correspondre 0 à dimanche, 1 à lundi, etc.

Exercice 7 : *Numéro du jour dans l'année*

Rajouter une méthode `jourDansLAnnee` qui prend comme paramètre une date en 2003 sous forme de deux entiers — jour et mois — et renvoie le nombre de jours passés depuis le début de l'année.

Exercice 8 : *Nom du jour en 2003*

Modifier le programme pour qu'il lise en entrée une date en 2003 (par exemple 14 07) et affiche le jour de la semaine correspondant. (Sachez que 31/12/2002 était un mardi).

Exercice 9 : *Compter les années bissextiles*

Rajouter une méthode `compteBissextiles` prenant comme paramètres deux années arbitraires et qui renvoie le nombre d'années bissextiles entre les deux.

On rappelle qu'une année a est bissextile si et seulement si elle est multiple de 4 mais pas de 100, ou si elle est multiple de 400.

Remarque : on peut utiliser une boucle. Mais on peut le faire plus simplement, en une ligne. Indice : regarder une règle graduée.

Exercice 10 : *Le futur*

En utilisant les méthodes ci-dessus écrire un programme qui prend en entrée n'importe quelle date après le 01/01/2003 et qui affiche le jour de la semaine correspondant.

Quel jour de la semaine sera le Noël 2006 ? Et votre $26^{\text{ème}}$ anniversaire ?

Exercice 11 : *Le passé*

Modifier le programme afin de pouvoir aussi traiter des dates avant 2003.

Quel était le jour de la semaine quand vous êtes né ?

Exercice 12 : *Saisie de la date*

Dans de nombreux exercices nous demandons à l'utilisateur de saisir une date. Pour cela on lui demande d'entrer le jour puis le mois et enfin l'année. Pour simplifier cela, on souhaite demander à ce dernier de rentrer une date sous la forme : jj/mm/aaaa.

Écrire des méthodes `jourDeDate`, `moisDeDate` et `anneeDeDate` permettant de récupérer le jour, le mois et l'année à partir d'une date au format jj/mm/aaaa. (On pourra utiliser `Deug.subString` et `Deug.intToString`.) L'inclure dans le programme de l'exercice précédent.

5 Des boucles et des méthodes

Note : Dans cette série d'exercices, utiliser le type `long` au lieu du type `int` pour représenter factorielles et coefficients binomiaux. Le type `long` permet de manipuler des entiers jusqu'à $2^{63} - 1 \approx 10^{19}$, ce qui permet de calculer jusqu'à 20!, alors que le type `int` ne permet d'aller que jusqu'à 12!.

Exercice 13 : *Factorielle*

On rappelle que la fonction factorielle est définie sur les entiers positifs de la façon suivante :

$$0! = 1 \qquad \text{et} \qquad n! = n \times (n-1)! \quad \text{si } n \geq 1$$

($n!$ se lit « factorielle (de) n ».)

(a) Écrire une méthode `factorielle` qui calcule la factorielle d'un entier passé en argument.

(b) Modifier la méthode précédente pour qu'elle interrompe le programme en appelant `Deug.exit` si on lui passe un argument supérieur à 20.

(c) (plus difficile) Modifier la méthode du *(a)* pour qu'elle détecte lors du calcul si la capacité du type `long` est dépassée (lui faire alors appeler `Deug.exit`).

Exercice 14 : *Coefficient binomial*

On définit le *coefficient binomial* des entiers naturels n et k (dans cet ordre !) ainsi : $C_n^k = \dfrac{n!}{k!(n-k)!}$

Écrire une méthode qui calcule le coefficient binomial de deux entiers. (On utilisera la méthode `factorielle`.)

Exercice 15 : *Calcul de e*

Le nombre e (base du logarithme népérien) peut se calculer ainsi : $e = \dfrac{1}{0!} + \dfrac{1}{1!} + \dfrac{1}{2!} + \ldots + \dfrac{1}{n!} + \ldots$

On note u_n la somme partielle d'ordre n : $u_n = \dfrac{1}{0!} + \dfrac{1}{1!} + \dfrac{1}{2!} + \ldots + \dfrac{1}{n!}$

(a) Écrire une méthode qui prend en argument un entier n et calcule u_n.

(b) On admet que pour $n \geq 1$, $e - \frac{1}{n!} < u_n < e$. Adapter la méthode du *(a)* pour obtenir une méthode qui prend en argument un réel ϵ et qui renvoie une valeur approchée de e à ϵ près.

(c) (plus difficile) Adapter la méthode du *(b)* pour obtenir la meilleure approximation possible de e, sachant que le type `double` a une précision limitée. Comparer à `Math.E`.