Rappel : fonctionnement (TD 4)

1. *Rappel :* Le script de vérification automatique de présence des exercices dans vos répertoires nécessite d'avoir exactement un fichier par exercice (comme demandé dans la page *Fonctionnement*).

   Par ailleurs, il est plus rapide et efficace de créer un nouveau fichier en y insérant un ancien programme que de commenter et/ou décommenter plusieurs programmes dans le même fichier.

   En outre, comme cela a été dit le premier jour du module, il est obligatoire que tous les exercices depuis le début du module soient organisés ainsi. Cette vérification est prévue pour la fin de la semaine.

   En particulier, même si vous pouvez bien sûr travailler chez vous, tous les exercices doivent être présents finalement sur les machines du CCI. (Rappel : on peut travailler de chez soi directement sur les machines du CCI par *ssh*, demandez aux enseignants.)

   Comme cela a été dit (et voir page *Fonctionnement*), le respect de toutes ces consignes de bon sens compte dans le CC.

2. *Travail personnel.* Assurez-vous que vous avez fait tous les exercices depuis le début du module et qu'ils sont organisés comme demandé. Relisez les pages *Fonctionnement* et *Environnement* pour vous assurer de ne rien avoir oublié.

Fonctions (TD 4a)

Exercices table : 3.
Rappel : respectez les consignes de la page *Fonctionnement*.
Attention : Dans chaque exercice, le *main* doit contenir un *unique* appel de fonction. C'est un peu artificiel au début, mais permet un excellent entraînement, rigoureux et très profitable. Même si ce n'est pas explicité dans l'énoncé, c'est à vous de rajouter des fonctions si nécessaire pour cela. Faites de même pour toute la suite du module, sauf mention contraire.

1. Placez-vous dans le répertoire adéquat demandé dans la page *Environnement*.
2. **Familiarisation** Tapez et exécutez le programme *Ex1.java* de la feuille d'exemples du cours, en respectant les consignes de la page *Fonctionnement*.
3. Fonctions et entiers.
   (a) Ecrivez un programme complet qui appellera dans son *main* la fonction suivante, après avoir lu son paramètre sur la ligne de commande. Cette fonction prend un entier et renvoie son carré (sans utiliser *sqr*).
   Dans toute la suite, vous lirez de même les paramètres sur la ligne de commande.
   (b) Ajoutez au programme précédent une fonction répondant à l'exercice 3 du TD 2 (somme des entiers entre 0 et n). Vous aurez donc deux fonctions dans la classe, plus une pour les appeler depuis *main*, puisque *main* ne doit contenir qu'un seul appel de fonction.
   Attention : sur machine, vous ne retaperez pas le code, vous le récupérerez dans votre code du TD correspondant. Vous ferez de même dans toute la suite du module.
   (c) Ajoutez une dernière fonction : la factorielle (exercice 6, TD 2).
4. Fonctions et tableaux. Dans cet exercice, tous les tableaux contiennent des *int*, et seront entrés en paramètre sur la ligne de commande. Placez les fonctions suivantes dans un programme, en suivant les consignes. Remarquez que tous les corps de fonction ont déjà été écrits dans des TD précédents ou exemples du cours : récupérez-les pra "copié-collé" sans les retaper surtout.

   Réfléchissez soigneusement poru choisir la signature de chaque fonction. Testez soigneusement.

   (a) *Affichage* d'un tableau. Dans cette question le tableau sera saisi sur la ligne de commande, mais sans fonction de saisie. Bien sûr, la fonction *main* se compose toujours d'unique appel de fonction.
   (b) *Calcul* du minimum d'un tableau.
   (c) *Modification* d'un tableau par remise à zéro de toutes ses cases.
   (d) *Allocation dynamique* : renvoyer une copie d'un tableau.
   (e) *Allocation dynamique* : fonction de saisie d'un tableau sur la ligne de commande. Réadaptez maintenant le reste du programme pour tirer parti de cette fonction.
5. Il n'y a pas d'exercice mémoire dans cette feuille, mais celui de la suivante, faite dans le même bloc, porte sur fonctions et champs de classe.

Si vous avez fini tous les exercices ci-dessus (et ceux de la feuille Champs de classe) avant la fin du TD

6. Reprenez l'exercice 21 du TD 3 (tri par insertion en place), et recodez-le au mieux avec des fonctions.

Champs de classe (TD 4b)

Exercices table : 3.
Rappel : Respectez les consignes de la page *Fonctionnement*.
Rappel : Dans cette feuille, et dans toute la suite du module, continuez à suivre les directives de la feuille précédente concernant les fonctions.

1. Placez-vous dans le répertoire adéquat demandé dans la page *Environnement*.
2. **Familiarisation** Tapez et exécutez le programme *Ex1.java* de la feuille d'exemples du cours, en respectant les consignes de la page *Fonctionnement*.
3. (a) Déclarer un champ de classe représentant le jour du mois. Ecrire une fonction l'affichant, et une autre l'incrémentant.
   (b) Ecrire une fonction prenant un tableau, et testant si la valeur du champ de classe ci-dessus apparaît dans le tableau. Si c'est le cas elle affiche *oui*, sinon elle affecte le champ de classe à la valeur de la case 0 du tableau et affiche *non, je le change*.
4. *Récapitulatif global.* Reprendre l'exercice 13 du TD 3, et le recoder en utilisant au mieux la totalité du matériel vu jusqu'à présent.
5. *Déroulement mémoire. Travail personnel.*
   On considère l'exécution du programme suivant (il compile sans erreur).
   (a) Donner la configuration de la mémoire immédiatement après l'exécution de la ligne contenant le commentaire : `// ici`. Vous utiliserez un croquis détaillé et l'algorithme vu en cours et TD. En particulier, on allouera obligatoirement les cases mémoire dans l'ordre croissant à partir de 101. Comme en cours et en TD, si une case contient successivement plusieurs valeurs, on les écrira de gauche à droite dans la case en les barrant d'un seul trait au fur et à mesure. De même, on ne réutilisera pas les zones de fonctions.
   (b) Donner le dernier état de la mémoire avant l'arrêt du programme.
   (c) Dire ce qu'affiche ce programme.

```
class Memoire {
    static int c;
    static int f (int[] s, int n) {
        n++;
        c--;
        s[n] = 2;
        int[] t = s;
        t[3] = 3;
        return n;
    }
    public static void main (String[] args ) {
        int[] t = new int[4];
        t[1] = t[0];
        t[c] = 1;
        int n = 0;
        int m = f(t, n);
        System.out.println(m+" "+n+" "+c); // ici
    }
}
```