Méthodes dynamiques (TD 9)

Exercices table : 1(a)i.
Rappel : Respectez les consignes de la page *Fonctionnement* et celles des feuilles précédentes.

Attention : A partir de ce bloc et dans toute la suite du module, on écrira tout directement avec des fonctions. Mais bien sûr, il faut aussi tout savoir faire sans fonctions.

0. (a) Placez-vous dans le répertoire adéquat demandé dans la page *Environnement*.
   (b) **Familiarisation** Tapez et exécutez le programme suivant de la feuille d'exemples du cours, en respectant les consignes de la page *Fonctionnement* : *Ex1.java*.

1. Dans cet exercice on considère le cahier des charges du TD (*cf.* TD récapitulatif bloc 8), mais avec les restrictions suivantes.

   - Un client a un seul compte, qui est un compte simple. (Il n'y a donc pas de tableau de comptes d'un client, et les comptes négociés ne sont pas reliés aux clients dans cet exercice.)

   (a) A partir du corrigé correspondant fourni, reprogrammez-le en utilisant des méthodes dynamiques chaque fois que c'est pertinent, et justifiez à chaque fois (statique ou dynamique). Suivez la progression suivante, en transformant une seule classe à la fois.
      i. Comptes simples seuls.
      ii. Clients.
      iii. Comptes négociés.
   (b) Indiquez tous les appels de méthode surchargée de votre code, et pour chacun comment Java résoud la surcharge (et ce que signifie "resoudre la surcharge pour un appel").
   (c) Indiquez tous les points de votre code dont la programmation n'est pas satisfaisante, en explicitant les critères considérés (non à cause de vous mais à cause du matériel Java vu qui est encore insuffisant).

2. Même exercice, mais un client a son ensemble de comptes, avec les restrictions suivantes.

   - Un client n'a que des comptes simples. (Les comptes négociés ne sont donc pas reliés aux clients dans cet exercice.)

3. *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 Bidule {
    int valeur;

    int g (int n) {
        n--;
        this.valeur += n;
        return this.valeur;
    }

    static void f (Bidule[] t, int n) {
        t[n] = t[n-1];
        t[n-1].valeur++;
    }
}

class Memoire {
    public static void main (String[] args ) {
        Bidule[] t;
        int n = 2;
        n--;
        t = new Bidule[n];
        t[n-1] = new Bidule();
        n--;
        System.out.println(t[n].g(n)); // ici
        n += 3;
        t = new Bidule[n];
        n--;
        t[n] = t[n-1];
        Bidule.f(t, n);
        System.out.println(t[n].valeur+" "+n);
        Bidule.f(t, n);
        System.out.println(t[n].valeur+" "+n);
    }
}
```

4. *Entraînement. Travail personnel.* Comme aux TD précédents, reprogrammez l'examen de l'an passé, en utilisant tout le matériel vu jusqu'à présent chaque fois que c'est pertinent, et justifiez à chaque fois. Indiquez tous les points de votre code dont la programmation n'est pas satisfaisante, en explicitant les critères considérés (non à cause de vous mais à cause du matériel Java vu qui est encore insuffisant).