TP 4

M.Adam – N.Delomez – JF.Kamp – L.Naert

8 août 2022

Objectifs du TP

- Utiliser des méthodes
- Écrire des méthodes

Attention les paramètres des méthodes à réaliser dans le cadre de ce TP sont toujours corrects. Ils ne peuvent provoquer une erreur à l'exécution. Aucun test d'erreurs n'est à réaliser dans les méthodes.

Exercice 1 (reprise du TD)

En mathématique, le nombre de combinaisons de k éléments parmi n s'écrit C_n^k et est défini pour $k \leq n$ par :

$$C_n^k = \frac{n!}{k!(n-k)!}$$

1. Écrire les deux méthodes suivantes :

```
/**
 * calcul de la factoriel du paramètre
 * @param n valeur de la factoriel à calculer
 * @return factoriel de n
 */
int factoriel (int n)
```

```
/**
 * calcul de la combinaison k parmi n
 * @param n cardinalité de l'ensemble
 * @param k nombre d'éléments dans n avec k<=n
 * @return nombre de combinaisons de k parmi n
 */
int combinaison (int n, int k)
```

2. Tester la méthode `factoriel()` en utilisant la méthode `testFactoriel()` :

```
/**
 * Teste la méthode factoriel()
 */
void testFactoriel () {
    System.out.println ();
    System.out.println ("*** testFactoriel()");

    testCasFactoriel (5, 120);
    testCasFactoriel (0, 1);
    testCasFactoriel (1, 1);
    testCasFactoriel (2, 2);
}

/**
 * teste un appel de factoriel
 * @param n valeur de la factoriel à calculer
 * @param result resultat attendu
 **/
void testCasFactoriel (int n, int result) {
    // Arrange
    System.out.print ("factoriel (" + n + ") \t= " + result + "\t : ");

    // Act
    int resExec = factoriel(n);

    // Assert
    if (resExec == result){
        System.out.println ("OK");
    } else {
        System.err.println ("ERREUR");
    }
}
```

3. Sur le modèle de `testFactoriel()`, écrire la méthode `testCombinaison()`.
4. **Rendre le code du programme et les tests d'exécution.**
5. **Bonus :** calculer `combinaison(25,24)`. Comment expliquer le résultat obtenu ?
6. **Bonus :** rendre l'explication sur la valeur obtenue.
7. **Bonus :** programmer, tester une méthode `combinaison` qui permette de passer outre le problème mis en évidence dans la question précédente.
8. **Bonus : rendre le code modifié et les tests.**

Exercice 2

1. Écrire une méthode `estDiviseur()` qui rend vrai si le deuxième entier divise le premier, faux sinon.

Par exemple, 2 est diviseur de 10 car 2*5 vaut 10, par contre 3 n'est pas diviseur de 10.

```
/**
 * teste la divisibilité de deux entiers
 * @param p entier positif à tester pour la divisibilité
 * @param q diviseur strictement positif
 * @return vrai ssi q divise p
 */
boolean estDiviseur (int p, int q)
```

2. Écrire la méthode `testEstDiviseur()` qui teste la méthode `estDiviseur()`.
3. **Rendre le code de la méthode `estDiviseur()`, de la méthode `testEstDiviseur()` et l'exécution**.

Exercice 3

Un nombre parfait est un nombre entier tel que la somme de ses diviseurs est égale à ce nombre. Par exemple, 6 est un nombre parfait car $6 = 1 + 2 + 3$ et que 1, 2 et 3 sont les diviseurs de 6. De même, 28 est aussi un nombre parfait car $28 = 1 + 2 + 4 + 7 + 14$. Les trois premiers nombres parfaits sont 6, 28, 496.

1. En utilisant la méthode `estDiviseur()`, écrire la méthode `estParfait()` :
```
/**
 * teste si un nombre est parfait
 * @param a entier positif
 * @return vrai ssi a est un nombre parfait
 */
boolean estParfait (int a)
```
2. Écrire la méthode `TestEstParfait()` qui teste la méthode `estParfait()`.
3. **Rendre le code de `estParfait()`, de `testEstParfait()` et l'éxécution**.

Exercice 4

1. Écrire une méthode `QuatreNbParfaits()` qui affiche les quatre premiers nombres parfaits.
```
/**
 * Affiche les quatre premiers nombres parfaits
 */
void quatreNbParfaits(){
```
2. **Rendre le code de `QuatreNbParfaits()`, et l'éxécution**.

Exercice 5

1. Écrire une méthode `testEstCroissant()` qui teste la méthode :
```
/**
 * teste si les valeurs d'un tableau sont triées par ordre croissant
 * @param t tableau d'entiers
```

```
     * @return vrai ssi les valeurs du tableau sont en ordre croissant
     */
    boolean estCroissant (int[] t)
```

2. Écrire le code de la méthode `estCroissant()`
3. **Rendre le code de `estCroissant()`, de `testEstCroissant()` et le résultat de l'exécution du test**.