TP 6

L. Naert – T. Ferragut – E. Lemonnier

8 juillet 2024

Objectifs du TP

- Initiation au debug
- Gestion des tableaux 2D : exemple du Morpion

Exercice 1 : Maintenance

Vous avez engagé un stagiaire pendant les vacances d'été pour écrire un petit programme qui calcule le nombre de jours entre deux dates. A la fin de son stage, le stagiaire disparait en vous laissant son code et un petit mot : "Mon code fonctionne dans tous les cas que j'ai testé. Je l'ai même commenté ! Vous n'aurez aucun mal à vous en servir.". Vous froncez les sourcils et ouvrez son fichier (présent sur Moodle) :

```
/**
* Compte le nombre de jours entre deux dates
* @author Stagiaire LN
*/
class ADebugger {

    void principal () {

    }


    //Pas important mais ne supprimez pas !
    boolean bsx (int annee){
        return (annee%4 == 0 && annee%100 !=0)||(annee%400 == 0);
    }

    //Pas important mais ne supprimez pas !
    int nbja (int a){
        int nb = 0;
```

```
    if (bsx(a))
    nb = 366;
    else
    nb = 365;


    return nb;
}

///Pas important mais ne supprimez pas !
int nbjm (int a, int b){
    int nb = 0;

    if (a == 2){
        if (bsx(b)){
            nb = 29;
        }else{
            nb = 28;
        }
    }else if(a <= 7){
        nb = 30 + a%2;
    }else{
        nb = 31 - a%2;
    }

    return nb;

}

// ! C'est celle la qui fait ce que l'on veut !
/// Attention : a/b/c est avant d/e/f
int nbj (int a, int b, int c, int d, int e, int f){
    int nb = 0;
    int i = c+1;


    while (i < f) {
        nb += nbja (i);
        i = i + 1;
    }

    i = b + 1;
    while (i <= 12) {
        nb += nbjm (i,c);
        i++;
    }
```

```
        i = 1;
        while (i < e) {
            nb += nbjm (i, f);
            i++;
        }

        nb += nbjm (b,c)-a;
        nb += d ;

        return nb;

    }


}
```

Votre première réaction est de refermer rapidement le fichier mais vous vous retenez et essayer de comprendre comment tout cela fonctionne.

1. Essayez de calculer le nombre de jours entre votre année de naissance et la date actuelle en utilisant la fonction prévue à cet effet.
2. Ce code sera bientôt mis à disposition du grand public sur le site web de votre entreprise mais la personne en charge de mettre les nouveaux programmes sur le site bloque les programmes qu'il ne comprend pas pour éviter tout problème de sécurité. Faites en sorte de rendre le code plus compréhensible grâce à l'ensemble des bonnes pratiques que vous connaissez.
3. Anticipant les manipulations incongrues de vos clients, vous essayez de calculer plusieurs cas d'utilisation et tombez sur un bug : le nombre de jours entre le 27/10/1818 et le 30/10/1818 est de 368 au lieu de 3 ! Faites en sorte que ce code fonctionne dans tous les cas (en considérant cependant que les clients ne sont pas trop taquins : les dates qu'ils écrivent sont toujours correctes et la première date est inférieure à la seconde).

   Rendre le code corrigé du programme et des exemples pertinents d'exécution de chacune des fonctions.
4. **Pour l'été prochain, quelles instructions supplémentaires donnerez-vous à votre nouveau stagiaire ?**

Exercice 2 : Morpion de taille variable

Vous allez coder un jeu de morpion de taille choisie par l'utilisateur (taille entre 2 et 9 inclus).

Rappel des règles du morpion : Les joueurs désignés par X et O jouent l'un après l'autre en plaçant des pions sur un quadrillage de taille n choisie (habituellement, n = 3). Le premier joueur qui aligne n pions en vertical, horizontal ou diagonal gagne. Si toutes les cases sont remplies sans aucun alignement, le jeu s'arrête sur une égalité.

Dans ce morpion d'exercice, **vous n'allez vérifier que les alignements horizontaux** (les plus motivés pourront rechercher les alignements verticaux et diagonaux).

Le plateau de morpion est un tableau de caractères à deux dimensions. Le symbole espace `' '` représente l'absence de pion. Le symbole `'X'` représente les pions du joueur X et `'O'` les pions du joueur O.

Pour simplifier le codage, nous avons séparé la tache en différentes fonctions. Les tests unitaires de chaque fonction ne sont pas demandés mais il est conseillé d'utiliser chacune des fonctions juste après l'avoir codé pour vérifier son bon fonctionnement.

1. Écrire la méthode d'affichage suivante :

   ```
   /**
   * Affichage du plateau de Morpion avec les indices de lignes
   * et de colonnes
   * @param plateau le tableau a afficher
   */
   void affichePlateau(char[][] plateau)
   ```

   Par exemple,

   ```
   void principal () {
       char[][] morpion = {{'X','O','X'},{' ','X',' '},{'O','O',' '}};
       affichePlateau(morpion);
   }
   ```

   doit afficher :

   ```
        0   1   2
   0  | X | O | X |
   1  |   | X |   |
   2  | O | O |   |
   ```

2. Écrire la méthode `creerPlateau()` :

   ```
   /**
   * Créer un plateau de jeu carré rempli de caractere espace ' '
   * @param lg taille du plateau (lg < 10 : pas à vérifier)
   * @return tableau de caractere en deux dimensions
   */
   char[][] creerPlateau(int lg)
   ```

3. Écrire la méthode `ajoutePion()` :

```
/**
* Demande deux coordonnees à l'utilisateur. Si les coordonnees sont
* correctes (dans le plateau et pas de pion déjà mis à cet endroit),
* le caractere du joueur est ajouté au plateau sinon, on redemande
* des coordonnées à l'utilisateur en expliquant l'erreur
* @param plateau le plateau de jeu
* @param joueur le caractere representant le joueur : X ou O
*/
void ajoutePion(char[][] plateau, char joueur)
```

4. Écrire la méthode `estRempli()` :

```
/**
* Verifie si toutes les cases du plateau sont remplies
* (différentes de ' ')
* @param plateau le plateau de jeu
* @return true si tout le plateau est rempli, false sinon.
*/
boolean estRempli(char[][] plateau)
```

5. Écrire la méthode `alignHorizontal()` :

```
/**
* Verifie s'il y a un alignement de n mêmes caracteres horizontalement
* (n étant la longueur ou largeur du plateau)
* @param plateau le plateau de jeu
* @return true s'il existe un alignement horizontal sur une ligne du plateau
*/
boolean alignHorizontal(char[][] plateau)
```

6. Écrire la méthode `changeJoueur()` :

```
/**
* Change le joueur courant
* @param joueurInitial un caractère représentant le joueur : X ou O
* @return si le joueur 'X' est en parametre alors renvoie 'O'
*  sinon renvoie 'X'
*/
char changeJoueur(char joueurInitial)
```

7. Remplir la méthode `jouer()` en utilisant les fonctions précédentes.

```
/**
* Lance une partie de morpion
* @param plateau le tableau a afficher
*/
void jouer(){
    //Demande une taille de plateau à l'utilisateur
```

```
        int taille = //...

        //Création du jeu
        char[][] morpion = //...

        // On commence par le joueur avec les X
        char joueur = 'X';

        //Boucle de jeu : elle continue tant que l'une des conditions de fin n'est pas att
        while(/* ...*/){

            //Affichage du tableau
            //...

            //les deux joueurs jouent l'un après l'autre
            //...

            //Changement de joueur
            //...

        }

        // Affichage du tableau final
        //...

        // Annonce de fin du jeu et déclaration du gagnant s'il existe
        // ...
    }
```

Rendre le code de la classe et un exemple d'exécution de la méthode `jouer()`.

8. **Facultatif** : Amélioration de votre morpion :
   - détecter les alignements verticaux et horizontaux
   - dans la méthode `ajoutePion()`, lister pour l'utilisateur les lignes avec des places disponibles puis, en fonction de la réponse de l'utilisateur, lister les colonnes autorisées.