TP 8: Awalé

Informatique Fondamentale (IF121)

8 Décembre 2004

L'objectif est de programmer le jeu africain à deux joueurs appelé Awalé et d'y jouer ensuite quand on a fini. Une description peut être trouver sur http ://www.ludoteka.com/awale.html, ou tout simplement sur Google.

Le plateau de jeu est représenté par un tableau `T` de 12 cases, dont les cases 0 à 5 représentent un camp et les cases 6 à 11 représentent l'autre camp. Le nombre de graines acquises par chaque joueur au cours de la partie seront conservées un tableau `butin` à 2 cases.

`T,butin` seront déclarés variables globales à la classe (chose que vous n'avez encore pas vu en TP), comme l'indique le code ci-après. On va programmer cinq méthodes principales. Des méthodes auxiliaires peuvent être utilisées, bien que ce ne soit pas nécessaire.

- la première affiche le plateau de jeu
- la deuxième sème n graines à partir de la case i
- la troisième capture les graines des cases qui n'en comportent que 2 ou 3
- la quatrième teste si un joueur ne peut plus jouer
- le `main` gère l'initialisation du jeu, l'interaction entre les deux joueurs et le décompte de la fin pour savoir qui a gagné.

Exercice 1 : *Affichage du plateau de jeu*

Comme son nom l'indique...

exemple : le plateau de jeu

```
4 5 2 3 0 1
3 0 4 5 2 0
```

correspond au tableau `T={3 0 4 5 2 0 1 0 3 2 5 4}`

Exercice 2 : *Semer les graines*

n est le nombre de graines à semer, i est la case d'où on les a prises.

Notons que la variable d'incrémentation `case` peut dépasser la taille du tableau s'il y a trop de graines à semer. Pour éviter de faire plein de tests, l'astuce consiste à considérer que la vraie case décrite par cette variable est en fait "case" modulo 12, i.e. `case%12`.

On fera attention que si le nombre de graines est supérieur à 12 et que l'on doit donc faire au moins un tour complet du plateau, il faut toujours sauter la case d'où l'on est parti afin de la laisser vide.

La méthode renvoie au final la dernière case où l'on a semé une graine, afin de regarder s'il y a des captures.

Exercice 3 : *Capture*

`trou` étant la dernière case où l'on a semé une graine, on regarde si la case contient 2 ou 3 graines, auquel cas on les ajoute au butin, et on recommence en parcourant les cases dans le sens inverse où on a semé les graines.

On s'arrête lorsque la case du plateau contient moins ou plus de 2 ou 3 graines, ou lorsque l'on passe dans l'autre camp du plateau.

Astuce : pour tester si deux cases sont dans le même camp, il suffit de faire la division entière par 6 et comparer les résultats.

Exercice 4 : *Test pour savoir si un joueur peut jouer*

On regarde si toutes les cases de son camp sont vides ou non (à l'aide d'une petite boucle). Il est pratique de considérer que la variable `joueur` a la valeur 0 lorsque l'on parle du 1er joueur, et la valeur 1 lorsque l'on parle du 2ème.

Exercice 5 : *Main*

On initialise le tableau `T`, que l'on remplit de la valeur 4.
Les butins seront conservés dans un tableau `butin`, à 2 cases, que l'on initialise et que l'on remplit de la valeur 0 au début de la partie.
On a une variable `joueur` qui vaut 0 ou 1 selon que ce soit au 1er ou au 2ème joueur de jouer.

Les tours de jeu seront dans une boucle (`while` est la plus pratique), dont il faut pouvoir sortir.

A l'intérieur de la boucle, il faut donc :

- afficher le plateau de jeu en appelant la méthode affichage
- afficher le butin du joueur qui doit jouer
- lui demander de rentrer un coup à jouer, c'est-à-dire un entier entre 0 et 11 qui corresponde à une case non-vide de son camp. Une entrée non-valide sera considérée comme une proposition à l'autre joueur d'arrêter la partie.
- dans ce dernier cas : demander alors à l'autre joueur s'il accepte d'arrêter la partie, et dans le cas contraire, redemander au joueur quel coup il joue.
- dans le cas d'un coup valide, appeler la méthode `seme` et récupérer la dernière case où une graine a été semée,
- dans le cas où celle-ci est dans le camp adverse, appeler la méthode `capture` et récupérer le nombre de graines à ajouter au butin du joueur.

Astuce : le numéro du joueur dont le camp possède la case `i` est `i/6`.

Le premier cas de sortie de la boucle est lorsqu'un joueur ne peut plus jouer. Ensuite, il faut gérer les cas où le jeu s'arrête parce qu'il tourne en boucle et aucune capture n'est plus possible. C'est aux joueurs de se mettre d'accord sur l'identification d'une telle situation. Autrement dit, lorsqu'un joueur doit jouer, il doit pouvoir soit choisir une case, soit demander à l'autre joueur s'il est d'accord pour s'arrêter (par exemple, en entrant un chiffre non-compris entre 0 et 11). Si l'autre est en effet d'accord, alors la variable `veulentarreter` passe à `true` et le jeu s'arrête. Cette technique permet aussi de gérer les abandons.

Lors de l'arrêt du jeu, il faut ajouter au butin de chacun les graines qui sont encore dans chaque camp. Puis on compare les butins et on annonce le vainqueur.

Exercice 6 : *Tester le jeu*

Le programme aura donc la tête suivante :

```
import fr.jussieu.script.Deug;

class Awale{

int[] T, butin;

    static void affichage(){
        ...
    }

    static int seme(int n, int i){
        int trou=i;
        int graines=n;
        while (...){
                ...
                }
        return (trou%12);
    }

    static int capture(int trou){
        int resultat=0;
        int j=trou;
        while(...){
                ...
                }
        return resultat;
    }

    static boolean peutjouer(int joueur){
        ...
    }

    public static void main (String[] args){
        T=new int[12];
        butin=new int[2];
        int joueur=0;
        boolean veulentarreter=false;
        ...
        while(peutjouer(joueur)&&!veulentarreter){
                ...
                }
        ...
        if (butin[0]>butin[1]) Deug.println("Le joueur 1 est vainqueur");
        if (butin[0]<butin[1]) Deug.println("Le joueur 2 est vainqueur");
        if (butin[0]==butin[1]) Deug.println("Les deux joueurs sont ex-aequo");
    }
}
```