Énoncés 10

Exercices sur les méthodes récursives

1 Exercice 1 : réécriture de boucles

Pour chacun des programmes suivants, remplacez la boucle par l'invocation d'une méthode récursive que vous écrirez.

Question 1

```
import java.util.Scanner;
public class RecQ1{
    public static void main(String[] args){
        for (int i=1; i<12; i++){
            System.out.println(i + " kilomètres à pied, ça use les souliers");
        }
    }
}
```

Question 2

```
import java.util.Scanner;
public class RecQ2{
    public static void main(String[] args){
        int sommePairs = 0;
        for (int i=1; i<12; i++){
            if (i % 2 == 0){
                sommePairs = sommePairs + i;
            }
        }
        System.out.print("la somme des nombres pairs inférieurs à 12 est ");
        System.out.println(sommePairs);
    }
}
```

Question 3

Il s'agit d'une variante du précédent.

```
import java.util.Scanner;
public class RecQ3{
    public static void main(String[] args){
        int sommePairs = 0;
        int max;
        Scanner scan = new Scanner(System.in);
        System.out.println("Nombre␣max:␣");
        max = scan.nextInt();
        for (int i=1; i<max; i++){
            if (i % 2 == 0){
                sommePairs = sommePairs + i;
            }
        }
        System.out.print("la␣somme␣des␣nombres␣pairs␣inférieurs␣à␣" + max +
                         "␣est␣");
        System.out.println(sommePairs);
    }
}
```

Notez que `max` n'est pas connu de la méthode récursive : il faut lui donner cette valeur en paramètre (en plus des autres paramètres nécessaires pour faire le calcul).

Question 4

Somme des valeurs d'un tableau.

```
import java.util.Scanner;
public class RecQ4{
    public static void main(String[] args){
        int somme = 0;
        int[] tab = {1, 17, 6, 8, 9, 14, 12, 10, 9};
        for (int i=0; i<tab.length; i++){
            somme = somme + tab[i];
        }
        System.out.print("la␣somme␣des␣nombres␣est␣");
        System.out.println(somme);
    }
}
```

Question 5

Dans cet exemple, il s'agit de remplacer la méthode qui comporte une boucle par un méthode récursive qui renvoie le résultat booléen. La méthode en question teste si un élément appartient à un tableau.

```
import java.util.Scanner;
public class RecQ5{
    public static boolean appartient(int elem, int[] tab){
        int i=0;
        while(i<tab.length && tab[i]!=elem)
            i=i+1;
```

```
        if (i<tab.length){
            return true;
        }else{
            return false;
        }
    }
    public static void main(String[] args){
        int atester;
        int[] tab = {1, 17, 6, 8, 9, 14, 12, 10, 9};
        Scanner scan = new Scanner(System.in);
        System.out.println("Entrez␣le␣nombre␣à␣tester:␣");
        atester = scan.nextInt();
        System.out.println("résultat␣du␣test:␣"+appartient(atester,tab));
    }
}
```

2 Exercice 2 : distance des cases accessibles

Dans cet exercice, vous allez reprendre et adapter le code de l'exemple du cours sur les labyrinthes (ce code est rappelé en annexe de ce document). La but de l'exercice est d'écrire une méthode qui calcule pour chaque case accessible, en combien de déplacements d'une case le pion peut l'atteindre. La case où est le pion est accessible en 0 coups, les cases voisines en 1 coup, les voisines des voisines en 2 coups, etc.

Pour ce faire, au lieu d'utiliser un tableau de caractères, on va utiliser un tableau d'entiers. Chaque case contiendra (à la fin du programme) le nombre de coups qu'il faudra au pion pour l'atteindre. Par convention, les cases non accessibles contiendront -9 s'il s'agit d'un mur, -1 s'il s'agit d'un couloir non accessible.

Question 1

Pour initialiser le processus, il faut transcrire le tableau de caractères en un tableau d'entiers de mêmes dimensions avec les valeurs initiales suivantes :

— chaque case contenant un mur est représentée par une case du tableau contenant la valeur -9.
— chaque case centenant un couloir est représentée par une case du tableau contenant la valeur -1 (jusqu'à preuve du contraire, tous les couloirs sont inaccessibles).

Écrivez une méthode qui prend en paramètre le tableau de caractères représentant le labyrinthe et renvoie en résultat le tableau d'entiers correspondant.

Question 2

Écrivez une méthode qui affiche le tableau d'entiers en considérant que les nombres du tableau s'affichent tous sur deux caractères au maximum (on prévoira un emplacement à deux cases dans tous les cas pour assurer que les colonnes sont correctement alignées, plus un espace pour séparer les nombres).

Question 3

Adaptez la méthode récursive du cours pour qu'elle inscrive dans chaque case accessible le nombre de coups mimimum pour le pion pour atteindre cette case.

La case où est le pion a pour valeur 0. Ensuite, chaque appel à une voisine doit avoir la valeur de la case d'où on part plus un.

Il y a plusieurs chemins pour atteindre une case, certains plus courts, d'autres plus longs. Quand on visite une case, il ne suffit pas de regarder si elle contient -1 : elle peut contenir une certaine valeur correspondant à un chemin, et en la visitant à nouveau, on peut avoir un nombre de coups plus petits pour l'atteindre. Dans ce cas, il faut remplacer l'ancienne valeur par la nouvelle et revisiter toutes les voisines qui seront moins éloignées du pion grâce à ce chemin plus court.

Annexe

Rappel du code.

```
public class AccessibiliteCase{
    public static void marqueAccessibleCaseEtVoisines(int x, int y,
                                                      char[][] tab){
        if (tab[x][y]==' '){
            tab[x][y] = 'ř';
            marqueAccessibleCaseEtVoisines(x-1,y,tab);
            marqueAccessibleCaseEtVoisines(x+1,y,tab);
            marqueAccessibleCaseEtVoisines(x,y-1,tab);
            marqueAccessibleCaseEtVoisines(x,y+1,tab);
        }
    }
    public static void afficheTableau(char[][] tab){
        for (int i=0;i<tab[0].length;i++){
            for (int j=0; j<tab.length; j++){
                System.out.print(tab[j][i]);
            }
            System.out.println();
        }
    }
    public static void main(String[] args){
        char[][] tab = {{'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', 'X', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', 'X', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', 'X', 'X'},
                        {'X', 'X', 'X', ' ', ' ', 'X', ' ', 'X'},
                        {'X', ' ', 'X', 'X', ' ', 'X', ' ', 'X'},
                        {'X', ' ', ' ', 'X', 'X', 'X', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                        {'X', 'X', 'X', 'X', ' ', 'X', 'X', 'X'},
                        {'X', ' ', ' ', ' ', ' ', 'X', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                        {'X', ' ', 'X', 'X', 'X', 'X', ' ', 'X'},
                        {'X', ' ', ' ', ' ', ' ', ' ', 'X', 'X'},
                        {'X', 'X', 'X', 'X', ' ', 'X', 'X', 'X'},
```

```
                                {'X', ' ', ' ', ' ', ' ', 'X', ' ', 'X'},
                                {'X', ' ', 'X', 'X', 'X', 'X', ' ', 'X'},
                                {'X', ' ', ' ', ' ', ' ', ' ', ' ', 'X'},
                                {'X', 'X', 'X', 'X', 'X', 'X', ' ', 'X'},
                                {'X', ' ', 'X', ' ', ' ', ' ', ' ', 'X'},
                                {'X', ' ', 'X', ' ', 'X', 'X', 'X', 'X'},
                                {'X', ' ', 'X', ' ', ' ', ' ', ' ', 'X'},
                                {'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'}};
        System.out.println("Voici le labyrinthe:\n");
        afficheTableau(tab);
        marqueAccessibleCaseEtVoisines(7,3,tab);
        System.out.println("\n\nLes cases accessibles depuis la case (7,3)" +
                                   " sont marquées par un rond: \n");
        afficheTableau(tab);
    }
}
```