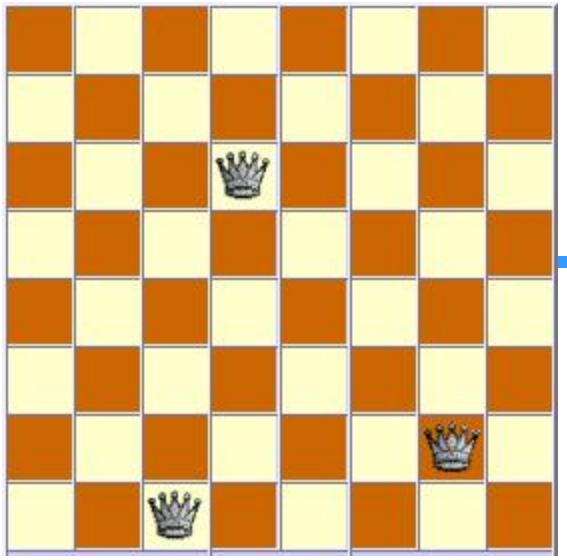
Informatique 1
L1 Portail IE
L1 Portail MA

Responsables:

pierre-alain.fouque@univ-rennes1.fr

patrick.derbez@univ-rennes1.fr



Tableaux bidimensionnels

Tableau à 2 dimensions de booléens T[0..7, 0..7]

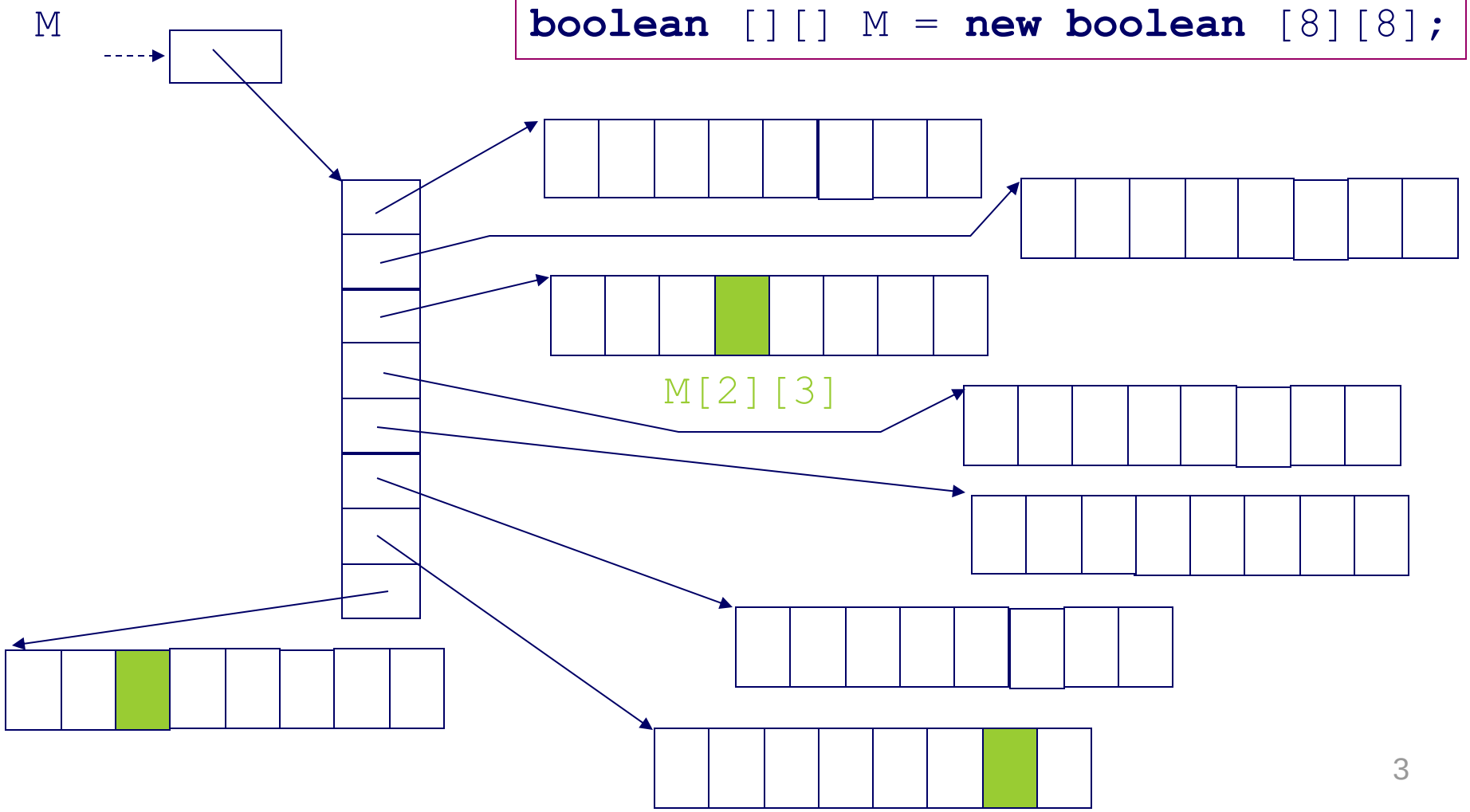
`T[i, j]` : la case ligne i, colonne j

Il y a des reines sur les cases

`T[2, 3]`
`T[7, 2]`
`T[6, 6]`

En java : tableau de tableaux

```
boolean [][] M = new boolean [8][8];
```



Déclarer et Afficher : tableau à 2 dimensions

```
public static void afficheTab2D(int[][] t) {
    for (int i=0; i<t.length; i++) {
        for (int j=0; j<t[i].length;j++)
            System.out.print(t[i][j] + " ");
        System.out.println();
    }
    System.out.println();
}
```

```
public static void main(String[] args) {
    int[][] tab = {{1,3},{4,5,6},{7,8,9}};
    afficheTab2D(tab);
}
```

```
1 3
4 5 6
7 8 9
```

Déclarer un tableau à 2 dimensions avec des valeurs et le programme va trouver les dimensions.

On écrit les lignes du tableau.

La fonction afficheTab2D affiche les lignes après les autres. Attention à la longueur !

Attention : toutes les lignes d'un tableau à 2 dimensions n'ont pas toute la même longueur !

Afficher un tableau à 2 dimensions

```
public static void afficheTab1D(int[] tab) {
    for (int i=0; i<tab.length; i++)
        System.out.print(tab[i] + " ");
    System.out.println();
}
```

```
public static void afficheTab2Dv1(int[][] tab) {
    for (int i=0; i<tab.length; i++)
        afficheTab1D(tab[i]);
    System.out.println();
}
```

```
public static void afficheTab2D(int[][] tab) {
    for (int i=0; i<tab.length; i++) {
        for (int j=0; j<tab[i].length; j++)
            System.out.print(tab[i][j] +" ");
        System.out.println();
    }
    System.out.println();
}
```

Initialiser tableau à 2 dimensions

```
public static int[][] init1(int m, int n) {
    int[][] tab = new int[m][n];
    return tab;
}

public static int[][] init(int m, int n) {
    int[][] tab;
    tab = new int[m][];
    for (int i=0; i<m; i++)
        tab[i] = new int[i+1];
    return tab;
}
```

La fonction init1 va réserver la mémoire pour stocker un tableau de m lignes et n colonnes et l'initialise à zéro.

La fonction init va réserver un tableau avec un nombre de colonne différentes suivant les lignes

Transposer: tableau à 2 dimensions

```
public class Tab2D {

    public static int[][] init1(int m, int n) {
        int[][] tab = new int[m][n];
        return tab;
    }

    public static void afficheTab2D(int[][] tab) {
        for (int i=0; i<tab.length; i++) {
            for (int j=0; j<tab[i].length; j++)
                System.out.print(tab[i][j] +" ");
            System.out.println();
        }
        System.out.println();
    }

    public static int[][] transpose(int [][] t){
        int[][] tt = init1(t[0].length, t.length);
        for (int i=0; i<t.length; i++)
            for (int j=0; j<t[i].length; j++)
                tt[j][i] = t[i][j];
        return tt;
    }

    public static void main(String[] args) {
        int[][] tab = {{1,2,3},{4,5,6}};
        afficheTab2D(tab);
        afficheTab2D(transpose(tab));
    }
```

Echanger la case (i,j) avec la case (j,i)

```
1 2 3
4 5 6

1 4
2 5
3 6
```

Echanger 2 lignes

```
public class Exe3 {

    public static void afficheTab2D(int[][] t) {
        for (int i=0; i<t.length; i++) {
            for (int j=0; j<t[i].length;j++)
                System.out.print(t[i][j] + " ");
            System.out.println();
        }
        System.out.println();
    }

    public static void echangeLigne(int[][] t, int i, int j) {
        int[] tmp = t[i];
        t[i]=t[j];
        t[j]=tmp;
    }


    public static int[][] init(int m, int n){
        int[][] tab = new int[m][n];
        return tab;
    }

    public static void main(String[] args) {
        int[][] tab = {{1,2,3},{4,5,6},{7,8,9}};
        afficheTab2D(tab);
        echangeLigne(tab,1,2);
        afficheTab2D(tab);
    }
}
```

On peut échanger les pointeurs des lignes

```
1 2 3
4 5 6
7 8 9

1 2 3
7 8 9
4 5 6
```

Echanger 2 colonnes: tableau à 2 dimensions

```
public static void echangeColonne(int[][] t, int i, int j) {
    int tmp;
    for (int k=0; k<t.length; k++) {
        tmp = t[k][i];
        t[k][i]=t[k][j];
        t[k][j]=tmp;
    }
}
```

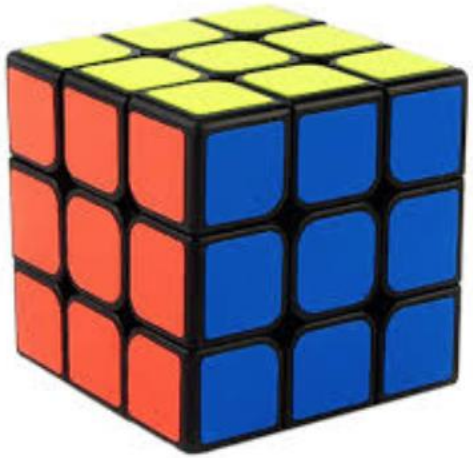
1 2 3
4 5 6
7 8 9

C'est plus de travail ….

1 3 2
4 6 5
7 9 8

Tableaux à 3 dimensions

```
int[][][] tab3D = {{{1,2,3},{4,5,6},{7,8,9}},{{10,11,12},{13,14,15},
    {16,17,18}},{{19,20,21},{22,23,24},{25,26,27}}};
```



Tester ligne du Sudoku

```java
// verifie si la ligne k contient tous les entiers
// comme le tableau contient 9 cases, si je vérifie que tous les
// entiers de 1 à 9 sont présents, il ne peut pas y avoir 2 fois
// le même entier
public static boolean checkLigne(int[][] tab, int k) {
    boolean b = true;
    boolean bc = false;
    // Lors de la i-ième passe, je vérifie si i+1 est dans le tableau
    for (int i=0; i<9; i++) {
        bc = false;
        // je recherche si i+1 est dans le tableau
        for (int j=0; j<tab[k].length; j++)
            if (tab[k][j]==i+1)
                bc=true;
        b = b & bc;
    }
    return b;
}
```

Tester ligne du Sudoku plus rapide

```java
// Version plus rapide: on ne parcourt qu'une fois le tableau
public static boolean checkLigne1(int[][] tab, int k) {
    boolean[] bc = new boolean[9];

    for(int i=0; i<bc.length; i++)
        bc[i] = false;

    for (int i=0; i<tab[k].length; i++) {
        if (tab[k][i]>=1 && tab[k][i]<=9)
        bc[tab[k][i]-1] = true;
    }

    for (int i=0; i<bc.length; i++)
        if(!bc[i])
            return false;

    return true;
}
```

Utiliser de la mémoire!

Tester un carre 3x3 de Sudoku

```
public class Exe3 {

    public static boolean checkCarre(int[][] carre) {
        boolean b = true;
        int[] btab = new int[9];

        for (int i=0; i<3; i++) {
            for (int j=0; j<3; j++) {
                if ((carre[i][j]>9) || (carre[i][j]<1))
                    return false;
                btab[carre[i][j]-1] = 1;
            }
        }
        for (int i=0; i<9; i++)
            if (btab[i]==0)
                b = false;
        return b;
    }

    public static void main(String[] args) {
        int[][] tab = {{1,2,3},{4,5,6},{7,8,9}};
        int[][] tab1 = {{2,2,3},{4,5,6},{7,8,9}};
        int[][] tab2 = {{0,2,3},{4,5,6},{7,8,9}};

        if (checkCarre(tab2))
            System.out.println("Carre correct");
        else
            System.out.println("Carre non correct");
    }
}
```

Tester un Sudoku

Le main ne contient que des appels de fonctions

Les fonctions peuvent s'appeler entre elles

```
public static boolean checkSudoku(int[][] tab) {
    boolean b = true;
    int[] btab = new int[9];

    for (int i=0; i<9; i++) {
        b &= checkLigne(tab[i]);
        b &= checkColonne(tab,i);
        b &= checkCarre(extractCarre(tab,i));
    }
    return b;
}

// la fonction verifie la colonne i
public static boolean checkColonne(int[][] tab, int i) {...}

// la fonction extrait le sous-tableau de taille 3x3 en numerotant
// a partir de 0 le tableau en haut a gauche
public static int[][] extractCarre(int[][] tab, int i) {...}

public static void main(String[] args) {
    int[][] tab = {{...}};
    if(checkSudoku(tab))
        System.out.println("correct");
    else
        System.out.println("pas correct");
}
```

Comment afficher les figures avec des étoiles avec un tableau à 2D ?

```
public class Exe3 {

    public static void afficheTab2D(char[][] t) {
        for (int i=0; i<t.length; i++) {
            for (int j=0; j<t[i].length; j++)
                System.out.print(t[i][j]);
            System.out.println();
        }
        System.out.println();
    }

    public static void main(String[] args) {
        char[][] tab = {{'*','*','*','*','*'},{'*',' ',' ',' ','*'},{'*
        afficheTab2D(tab);
    }
}
```