Programme mode projet

# Sommaire

## Comprendre ce qu’est un tableau

### [Expliquer ce qu’est un tableau et ses caractéristiques](#_Expliquer_ce_qu’est)

## Comprendre ce qu’est une matrice

### [Expliquer ce qu’est une matrice et à quoi cela peut servir](#_Expliquer_ce_qu’est_1)

## [Implémenter une matrice avec un tableau à deux dimensions depuis spécification](#_Implémenter_une_matrice)

## Implémenter la classe MatriceTab qui représente une matrice carrée d'entiers

### [La classe maintient deux propriétés :](#_La_classe_maintient)

### [Un tableau à deux dimensions matrix](#_Un_tableau_à)

### [Un entier size](#_Un_entier_size)

### [La classe à deux constructeurs](#_La_classe_à)

### [Un constructeur qui prend en paramètre un tableau à deux dimensions](#_Un_constructeur_qui)

### [Un constructeur qui prend en paramètre un entier et une valeur par défaut](#_Un_constructeur_qui_1)

### [La classe à une méthode ToString qui affiche la matrice](#_La_classe_à_1)

### [La classe à une méthode qui permet de récupérer une valeur aux coordonnées x, y](#_La_classe_à_2)

### [La classe à une méthode qui permet d’affecter une valeur à la case de coordonnées x, y](#_La_classe_à_3)

## [Chaque fonction sera commentée :](#_Nom_de_la)

### [Nom de la fonction](#_Nom_de_la)

### [Type de retour](#_Type_de_retour)

### [Nom et type des paramètres](#_Nom_et_type)

### [Une phrase qui explique ce que fait la fonction](#_Une_phrase_qui)

## Comprendre ce qu’est une liste

### [Expliquer ce qu’est une liste et la différence avec un tableau](#_Expliquer_ce_qu’est_2)

## Implémenter une matrice avec une liste de listes depuis spécification

### [Implémenter la classe *MatriceList* qui représente une matrice carrée d’entiers :](#_Implémenter_la_classe)

* + 1. Une liste de listes matrix
    2. Un entier size

### [La classe à deux constructeurs :](#_La_classe_à_4)

### [Un constructeur qui prend en paramètre une liste de liste](#_Un_constructeur_qui_2)

### [Un constructeur qui prend en paramètre un entier et une valeur par défaut](#_Un_constructeur_qui_3)

### [La classe à une méthode *ToString* qui affiche la matrice](#_La_classe_à_5)

### [La classe à une méthode qui permet de récupérer une valeur aux coordonnées x, y](#_La_classe_à_6)

### [La classe à une méthode qui permet d’affecter une valeur à la case de coordonnées x, y](#_La_classe_à_7)

## Pourquoi implémenter une matrice avec des listes

### [Expliquer l’intérêt de cette implémentation. Que pourrions-nous faire avec celle-ci (*MatriceList)* que l’implémentation *MatriceTab* ne permet pas.](#_Expliquer_l’intérêt_de)

## Implémenter la fonction Increase dans la MatriceList

### [La fonction increase agrandir la matrice de la classe *MatriceList* elle prend en paramètre un entier et une valeur par défaut.](#_La_fonction_increase)

## Comprendre ce qu’est un tableau

### Expliquer ce qu’est un tableau et ses caractéristiques

Il existe plusieurs types de tableau en CSharp :

* Il y a les tableaux simples (qui permet de rassembler sous un même identificateur des données de même type)
* Les tableaux à deux dimensions (c’est-à-dire un tableau avec n nombre de ligne et m nombre de colonnes)
* Les tableaux de tableaux.

Lors de l’implémentation d’un tableaux simple la première valeur se voit initialiser à nul, pour crée un tableau de 4 entiers on utilise donc la commande suivante :

1. int[] myArray = new int[4];

On peut aussi crée un tableau avec des valeurs par défaut :

1. int[] myArray = new int[] {10, 20, 30, 40};

On peut simplifier et noter directement :

1. int[] myArray = {10, 20, 30, 40};

Un tableau à deux dimensions se déclare de cette manière :

1. Type[,] nom = new type [n,m]
2. n est le nombre de ligne et m le nombre de colonnes.

De cette manière avec des réels cela nous donne :

1. double[,] myArray = new double[,]
2. {
3. {0.1, 0.5},
4. {1.3, 1.7}
5. };

On déclare donc un tableau double à deux lignes et à deux colonnes.

Les tableaux de tableau sont déclarés de cette façon :

1. Type[][] nom = new type[n][]; ou n est le nombre de ligne

De cette façon on peut donc initialiser un tableau de tableau :

1. Int[][] myInArray = new int[2][];
2. On déclare un tableau de deux lignes
3. myInArray[0] = new int[5];
4. Le premier tableau à une longueur de 5-1 (car le premier terme et a la case 0 du tableau)
5. myIntArray[0][1] = 42
6. La deuxième case du tableau

## Comprendre ce qu’est une matrice

### Expliquer ce qu’est une matrice et à quoi cela peut servir

Une matrice est un tableau de tableau, il permet d’afficher celui-ci sous plusieurs formes

## Implémenter une matrice avec un tableau à deux dimensions depuis spécification

### Implémenter la classe MatriceTab qui représente une matrice carrée d'entiers

### La classe maintient deux propriétés :

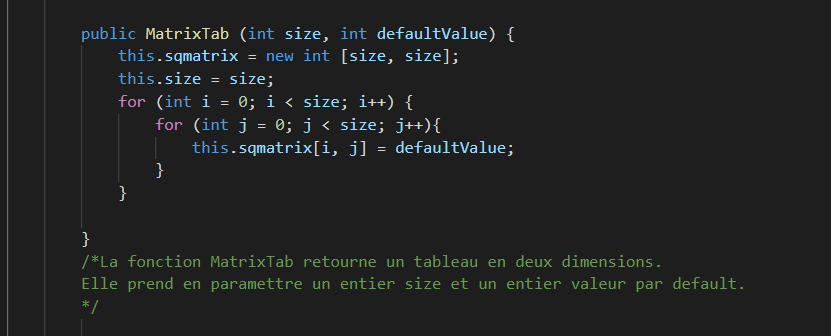
### Un tableau à deux dimensions matrix

### Un entier size

### La classe à deux constructeurs

### Un constructeur qui prend en paramètre un tableau à deux dimensions

### Un constructeur qui prend en paramètre un entier et une valeur par défaut

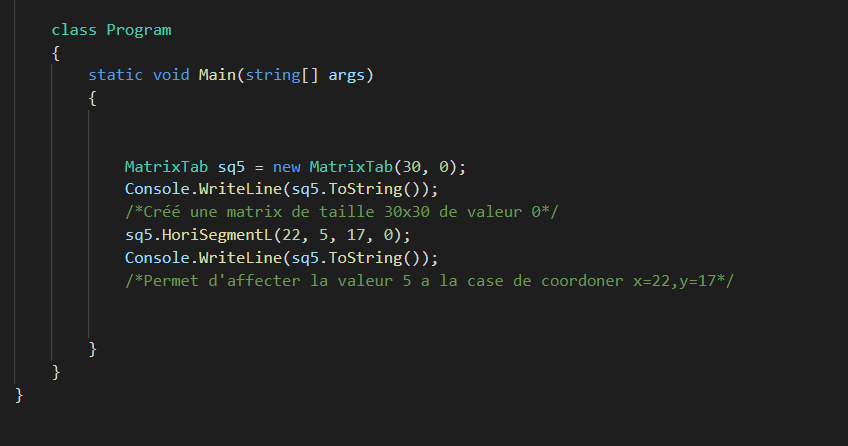


### La classe à une méthode ToString qui affiche la matrice

### La classe à une méthode qui permet de récupérer une valeur aux coordonnées x, y

### La classe à une méthode qui permet d’affecter une valeur à la case de coordonnées x, y

Le résultat nous donne donc :





## Chaque fonction sera commentée :

### Nom de la fonction

### Type de retour

### Nom et type des paramètres

### Une phrase qui explique ce que fait la fonction

## Comprendre ce qu’est une liste

### Expliquer ce qu’est une liste et la différence avec un tableau

Une liste est une structure de données qui pointe vers plusieurs données, comparée à un tableau qui lui contient des données.

Une liste est donc redimensionnable, à l’inverse un tableau reste fixe est donc pour le redimensionner la mémoire de l’ordinateur va donc recrée un tableau avec les anciennes données et les nouvelles données et supprime l’ancien pour libéré l’espace.

## Implémenter une matrice avec une liste de listes depuis spécification

### Implémenter la classe *MatriceList* qui représente une matrice carrée d’entiers :

* + 1. Une liste de listes matrix
    2. Un entier size

### La classe à deux constructeurs :

### Un constructeur qui prend en paramètre une liste de liste

### Un constructeur qui prend en paramètre un entier et une valeur par défaut

### La classe à une méthode *ToString* qui affiche la matrice

### La classe à une méthode qui permet de récupérer une valeur aux coordonnées x, y

### La classe à une méthode qui permet d’affecter une valeur à la case de coordonnées x, y

## Pourquoi implémenter une matrice avec des listes

### Expliquer l’intérêt de cette implémentation. Que pourrions-nous faire avec celle-ci (*MatriceList)* que l’implémentation *MatriceTab* ne permet pas.

## Implémenter la fonction Increase dans la MatriceList

### La fonction increase agrandir la matrice de la classe *MatriceList* elle prend en paramètre un entier et une valeur par défaut.