

# Namespace Interface

## Classes

[Dictionnaire](#)

[Jeu](#)

[Joueur](#)

[Plateau](#)

# Class Dictionnaire

Namespace: [Interface](#)








Assembly: Interface.dll

```
public class Dictionnaire
```

## Inheritance

[object](#)  ← Dictionnaire

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Constructors

### Dictionnaire(string, string)

Constructeur du dictionnaire prend en entrée le nom du fichier qu'il va lire et convertir en tableau de tableaux Tout en triant grace à un tri fusion les elements de chaque ligne quand il l'ajoute au tableau Les try catch permettent d'éviter les exception du type : Fichier non trouvé ou indice hors des limites

```
public Dictionnaire(string nomDuFichier, string l)
```

## Parameters

nomDuFichier [string](#) 

l [string](#) 

## Properties

### NomFichier

```
public string NomFichier { get; }
```

Property Value

[string](#)

## Methods

### DicoToString()

Methode pour decire une instance de la classe Dictionnaire en donnant pour chaque lettre le nom de mots qu'il contient

```
public string DicoToString()
```

Returns

[string](#)

### RechDichoRecuratif(int, int, string[], string)

Methode de classe qui effectue la recherche recursive dans le dico comme vu en TD et cours de Complexité

```
public static bool RechDichoRecuratif(int debut, int fin, string[] dico, string mot)
```

Parameters

debut [int](#)

fin [int](#)

dico [string](#) []

mot [string](#)

Returns

[bool](#)

## RechDichoRecurusifInstance(string)

Methode d'instance qui fait la recherche Recursive dans le dico pour plus de praticité

```
public bool RechDichoRecurusifInstance(string mot)
```

### Parameters

mot [string](#)

### Returns

[bool](#)

## TriFusionRecurusif(string[], int, int)

Methode qui effectue le tri fusion comme vu en TD

```
public static void TriFusionRecurusif(string[] tab, int debut, int fin)
```

### Parameters

tab [string](#)[]

debut [int](#)

fin [int](#)

# Class Jeu

Namespace: [Interface](#)








Assembly: Interface.dll

```
public class Jeu
```

## Inheritance

[object](#)  ← Jeu

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors


Jeu(Dictionnaire, Plateau, string, string, int, int)

```
public Jeu(Dictionnaire dico, Plateau plateau, string joueur1, string joueur2, int tempsTour  
= 10, int tempsTot = 180)
```

## Parameters

dico [Dictionnaire](#)

plateau [Plateau](#)

joueur1 [string](#) 

joueur2 [string](#) 

tempsTour [int](#) 

tempsTot [int](#) 

## Methods

## Joueur1(TimeSpan, DateTime)

```
public void Joueur1(TimeSpan t, DateTime debut)
```

### Parameters

t [TimeSpan](#)

debut [DateTime](#)

## Joueur2(TimeSpan, DateTime)

```
public void Joueur2(TimeSpan t, DateTime debut)
```

### Parameters

t [TimeSpan](#)

debut [DateTime](#)

## Resume()

```
public void Resume()
```

# Class Joueur

Namespace: [Interface](#)

Assembly: Interface.dll

```
public class Joueur
```

## Inheritance

[object](#) ← Joueur

## Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

## Constructors

### Joueur(string)

```
public Joueur(string n)
```

## Parameters

n [string](#)

## Properties

### Mots

```
public List<string> Mots { get; }
```

## Property Value

[List](#) <[string](#)>

# Nom

```
public string Nom { get; }
```

Property Value

[string](#)

# Score

```
public int Score { get; }
```

Property Value

[int](#)

# Methods

## Add\_Mot(string)

```
public void Add_Mot(string mot)
```

Parameters

**mot** [string](#)

## Add\_Score(string)

```
public void Add_Score(string mot)
```

Parameters

**mot** [string](#)



## Contient(string)

```
public bool Contient(string mot)
```

### Parameters

mot [string](#)↗

### Returns

[bool](#)↗

## CreerLettresRegles(string)

```
public void CreerLettresRegles(string fLettresRegles)
```

### Parameters

fLettresRegles [string](#)↗

## toString()

```
public string toString()
```

### Returns

[string](#)↗

## toStringListe()

```
public string toStringListe()
```

### Returns

[string](#)↗



# Class Plateau

Namespace: [Interface](#)








Assembly: Interface.dll

```
public class Plateau
```

## Inheritance

[object](#)  ← Plateau

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Constructors

### Plateau(int, string, string)

Constructeur d'un plateau pour le cas sans fichier

```
public Plateau(int taille = 8, string fPlateau = "Plateau.csv", string fLettresRegles  
= "Lettre.txt")
```

## Parameters

taille [int](#) 

fPlateau [string](#) 

fLettresRegles [string](#) 

### Plateau(string)

Constructeur du plateau pour le cas avec un fichier

```
public Plateau(string fPlateau)
```

## Parameters

fPlateau [string](#)

## Methods

### CreerLettresRegles(string)

Met le contenu du fichier Lettre.txt dans string[][] lettresRegles

```
public void CreerLettresRegles(string fLettresRegles)
```

## Parameters

fLettresRegles [string](#)

### DecaleColonnePlateau(char[,], int, int)

Methode qui decale une colone d'une matrice plusieurs fois en fonction de la ligne à laquel la lettre effacée est et si la case en dessous est effacé aussi ou pas

```
public static void DecaleColonnePlateau(char[,] plateau, int col, int ligne)
```

## Parameters

plateau [char](#)[,]

col [int](#)

ligne [int](#)

### DecaleColonnePlateau1(char[,], int)

Methode qui decale une colonne d'une matrice de 1

```
public static void DecaleColonnePlateau1(char[,] plateau, int col)
```

## Parameters

plateau [char](#)[]

col [int](#)

## InitiaIntTab0(int[])

Methode qui initialise chacun des membres d'un tableau de int à 0 de facon à ne pas avoir un tableau rempli de valeur null

```
public static void InitiaIntTab0(int[] tab)
```

## Parameters

tab [int](#)[]

## MajPlateau(Dictionary<(int, int), char>)

Methode qui met à jour le Plateau

```
public void MajPlateau(Dictionary<(int, int), char> res)
```

## Parameters

res [Dictionary](#) <(int, int), [char](#)>

## PlateauToString()

Methode qui retourne le plateau en string (toString())

```
public string PlateauToString()
```

## Returns

[string](#)

## RechercheMot(string)

Methode qui Recherche un mot dans la matrice et qui retourne un Dictionary avec les coordonnées des lettres si le mot est trouvé

```
public Dictionary<(int, int), char> RechercheMot(string mot)
```

### Parameters

mot [string](#)

### Returns

[Dictionary](#) <(int, int), char>

## RemplirPlateauLettres()

Rempli le plateau de lettre en appelant ChoisirLettre pour choisir des lettre avec une probabilité d'apparition cohérente à la probabilité de retrouvé ces lettres dans la langue française

```
public void RemplirPlateauLettres()
```

## ToFile()

Methode qui écrit le fichier du plateau et sauvegarde la plateau à l'intérieur

```
public void ToFile()
```

## ToRead()

Methode qui lit le fichier csv et convertit son contenu en Plateau

```
public char[,] ToRead()
```

### Returns

