# PROBLEME S3 : Algo et POO

# Préambule

Ce travail est à réaliser seul

Il est à rendre au plus tard le vendredi 20 décembre 2019

Vous déposerez votre solution complète Visual Studio sur Devinci-online au format .zip sous votre nom.

Un programme ne compilant pas ou ne s’exécutant pas entraîne une note de 0/20.

L’ensemble des codes sera analysé par un système anti-plagiat. Un plagiat entraîne une note de 0/20 au module (pour les 2 protagonistes)

Le sujet proposé a pour objectif de mettre en application tous les concepts vus en TD, c’est pourquoi il est impératif de suivre les énoncés tels qu’ils ont été écrits.

# Recommandations générales

Ne pas oublier votre projet Test Unitaire le plus exhaustif possible

Ne pas oublier les commentaires (/// et la génération de votre code XML)

Ne pas oublier d’écrire les variables et fonctions avec des noms lisibles

Ne pas oublier l’indentation.

Un ensemble de méthodes sont imposées avec des signatures précises.

Vous pouvez rajouter d’autres méthodes si les besoins de votre code l’imposent

Ne pas oublier que l’utilisation de

Random r = new Random()

ne peut se faire qu’une seule fois. Ensuite r.Next(..) peut se faire autant de fois que nécessaire

# 1 - Présentation du problème MIXMO réel

Il s’agit d’un jeu de mots (revu pour les besoins de la programmation). Chaque joueur compose sa propre grille de mots. Le gagnant est bien sûr le premier à terminer sa grille une fois que tous les jetons ont été utilisés

Au départ, les 120 jetons sont au centre de la surface du jeu. Chaque joueur en pioche 6 au hasard.

Les jetons sont comme suit

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Joker |
| 10 | 2 | 3 | 4 | 17 | 2 | 3 | 3 | 9 | 2 | 1 | 6 | 3 | 7 | 7 | 3 | 2 | 6 | 7 | 7 | 6 | 3 | 1 | 1 | 1 | 1 | 2 |

.

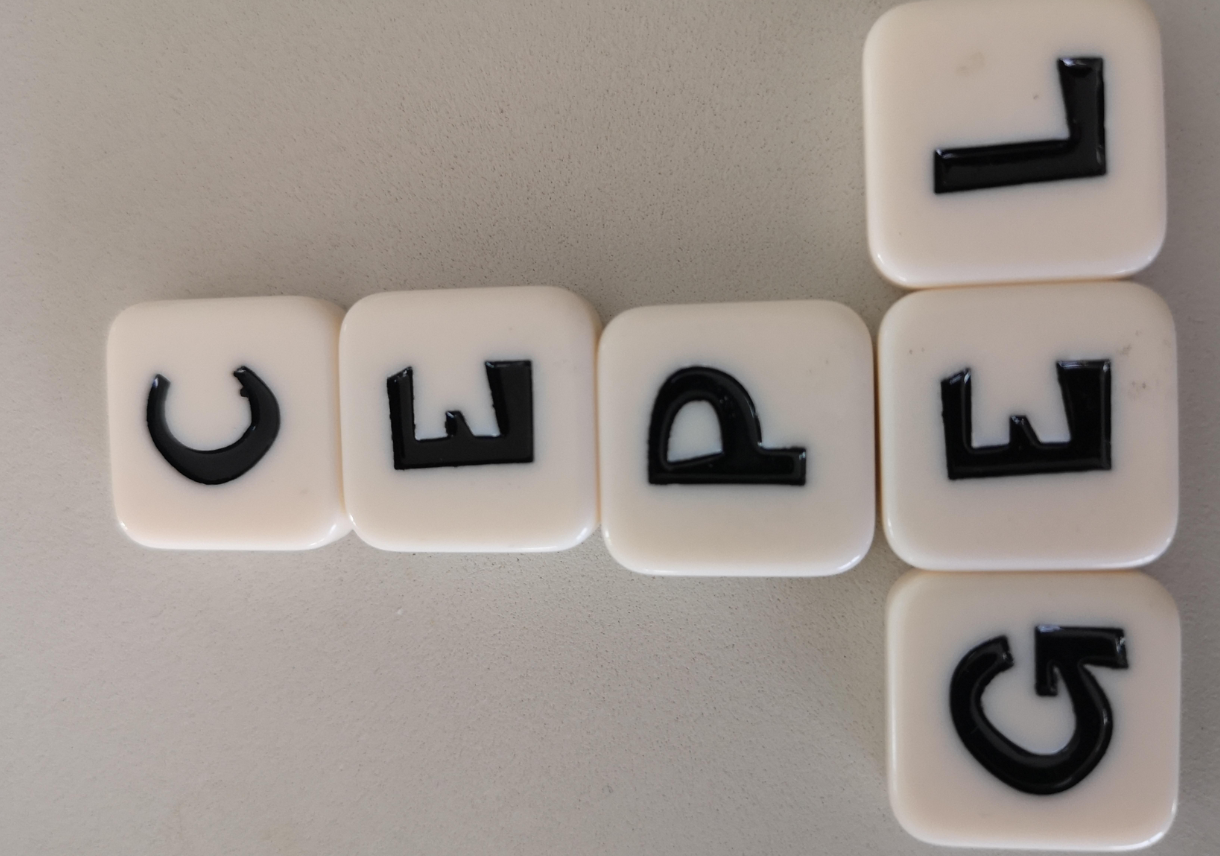
Au top départ, chaque joueur affiche ses 6 jetons et doit tous les placer aussi vite que possible de façon à former un mot de 6 lettres ou plusieurs mots qui s’entrecroisent comme sur une grille de mots croisés. Le premier qui parvient à placer ses 6 lettres, signale à tous les joueurs de piocher 2 nouvelles lettres en même temps que lui ; ces 2 lettres doivent être incorporées au jeu existant.

# Exemple

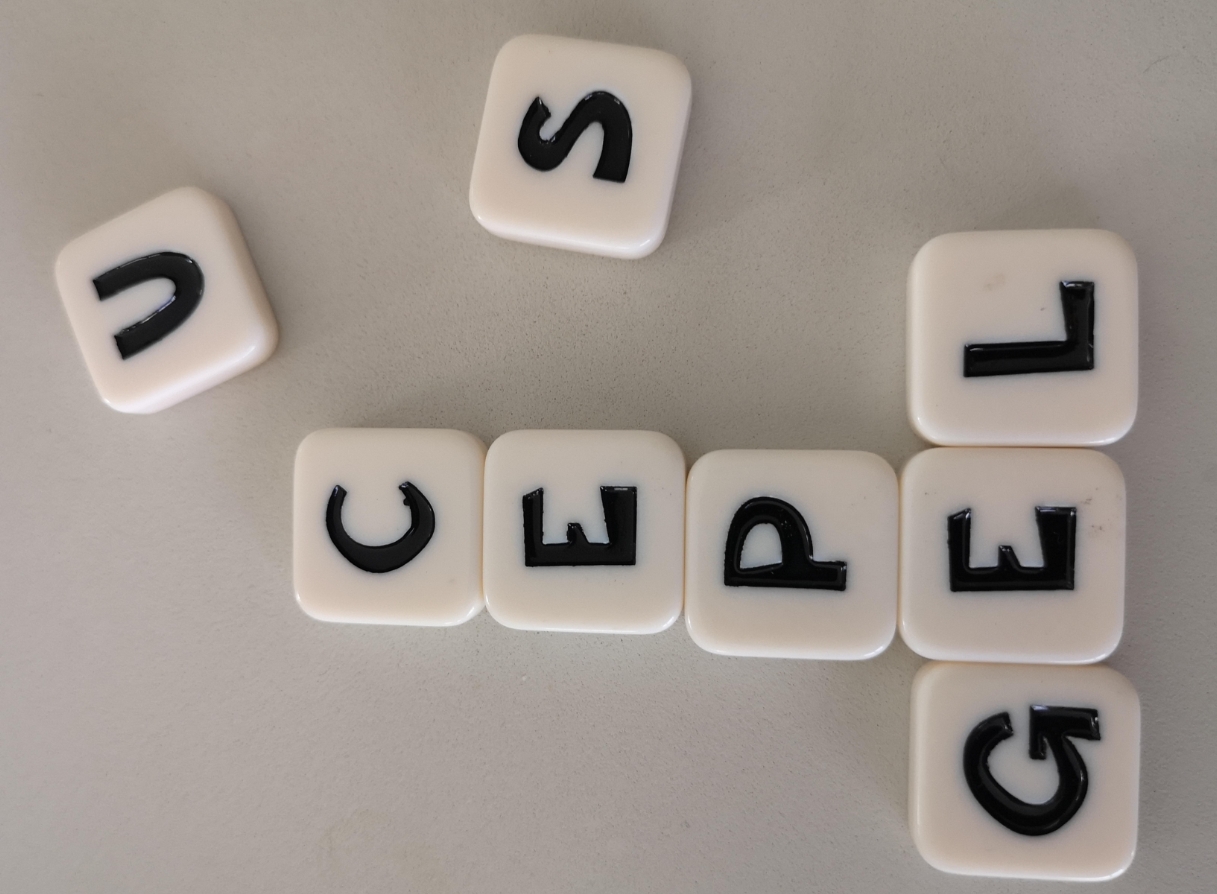
Supposez les 6 lettres tirées au hasard par le joueur 1 :

C – E – P – G – L – E – E

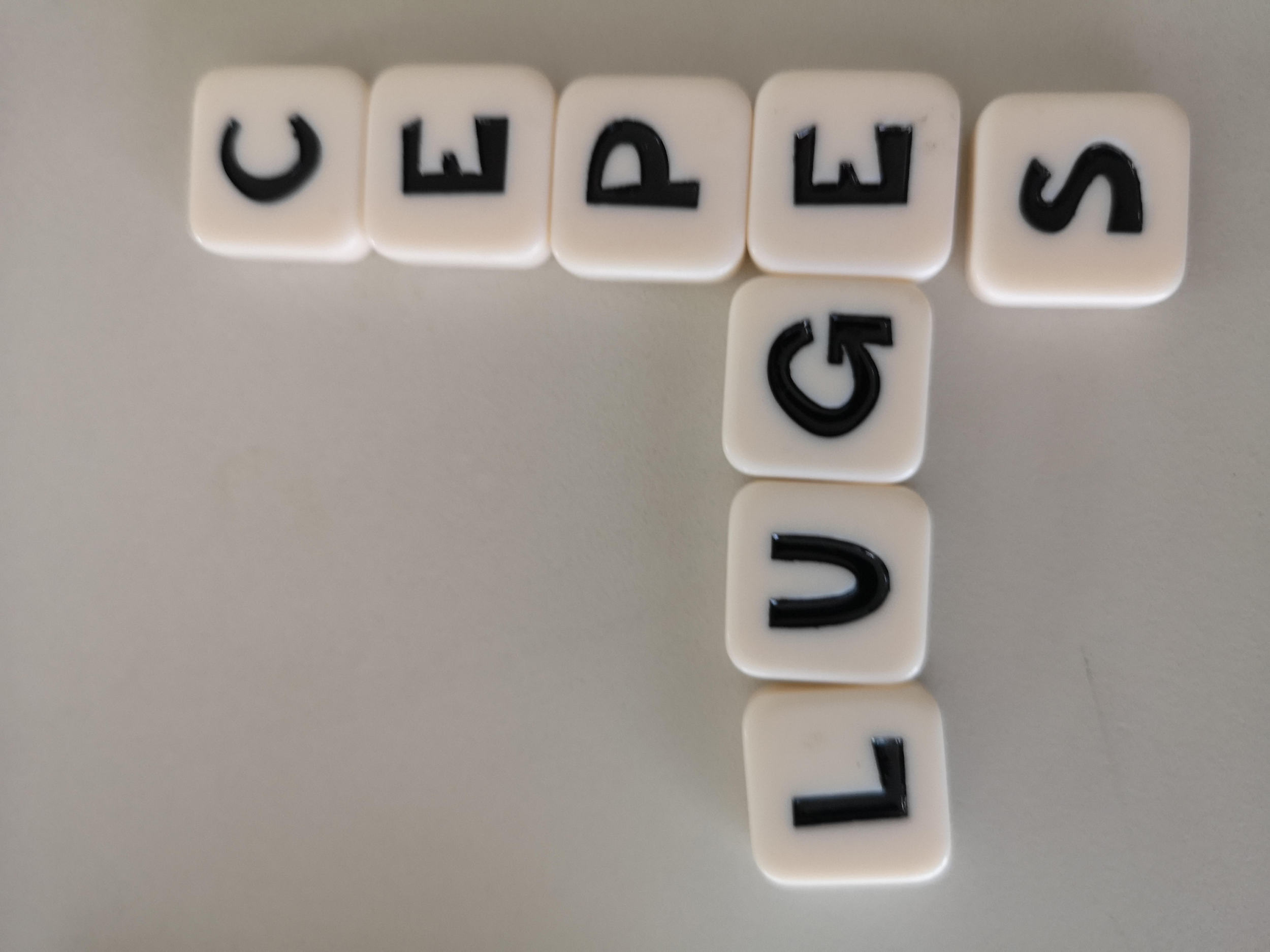
Le joueur 1 trouve la combinaison suivante : Il saisit Mixmo (ou un équivalent)



Il doit comme ses adversaires re-piocher 2 lettres au hasard : ici U - S



Il reformule avec ces 8 lettres une autre combinaison comme suit :



Vous remarquez que rien n’est figé. Vous pouvez défaire partiellement ou complètement votre grille et de réorganiser vos lettres pour toutes les placer dans votre grille. Chaque fois qu’un joueur réussit à intégrer toutes ses lettres à sa grille, le même processus se répète. Vous avez aussi 2 jokers qui peuvent remplacer n’importe quelle lettre de l’alphabet. (lorsqu’un joker est utilisé au croisement de 2 mots, la lettre remplacée doit être la même pour chacun des mots

La partie se termine lorsqu’il n’y a plus de lettres à piocher et que l’un des joueurs réussit à insérer la totalité de ses lettres dans sa grille de mots. Lorsque le « Mixmo » final est annoncé tous les joueurs doivent cesser de composer leur grille.

# 1-1 Pour être valide une grille doit comporter

1. Exclusivement des mots autorisés et bien orthographiés (cf le dictionnaire joint pour le programme)
2. Les mots doivent se croiser correctement suivant les mêmes règles qu’aux mots croisés : A savoir une succession quelconque de lettres (au moins 2) en horizontal ou vertical doit être valide
3. Le joueur ne peut pas réutiliser plusieurs fois le même mot

Vous limiterez le tour de jeu soit par la liquidation des lettres d’au moins un joueur soit par un timing d’une minute (afin d’éviter que la partie soit trop longue). A l’issue du tour, tous les joueurs reprennent 2 lettres.

Vous pourrez utiliser les classes suivantes (DateTime et TimeSpan) pour déterminer un timer.

De même, pour déterminer la fin de la partie, soit les joueurs ont épuisé toute la pioche et un joueur ne possède plus aucune lettre, soit vous pouvez mettre un timing à votre convenance pour éviter une partie trop longue.

# 1-2Pour calculer le score :

Une victoire rapporte 20 points. Si le timer a eu raison de la fin de la partie, il n’y a pas de victoire.

Vous comptez ensuite les points des grilles des joueurs.

* Un K, W, X, Y ou Z rapporte 5 points
* Les mots de 5 lettres rapportent un bonus égal au nombre de lettres contenues dans vos mots (5 lettres 5 points, 6 lettres 6 points …)

# 2 - MIXMO A PROGRAMMER

Nous allons édulcorer la version réelle

Version 0

Vous ne prenez pas toutes les lettres (120) mais un sous ensemble avec un pourcentage de voyelles plus important pour vous donner plus de facilité.

Vous n’utilisez pas le joker

Vous ne re-mélangez pas les lettres, vous vous contentez d’ajouter des lettres à la grille en construction.

2 joueurs

Version 1

Utilisez les 118 lettres (sans le joker) telles que le jeu le préconise

Nombre de joueurs > 2

Version 2

Vous rajoutez le fait que vous pouvez repartir de zéro pour votre grille

Vous utilisez le joker

En revanche, vous reprenez les points de la version réelle :

# Pour être valide une grille doit comporter (1-1)

* Pour calculer le score (1-2)

Vos algorithmes seront basés sur la programmation objet et pour cela vous créerez au moins 4 classes : Joueur, Plateau, Dictionnaire et la classe Jeu

# Exercice 1 : Classe Joueur

Un joueur est caractérisé a-minima par son nom, son score, une liste de lettres tirées au hasard, une liste des mots déjà trouvés et par une grille de mots croisés.

La création d’un joueur n’est possible que si celui-ci a un nom.

Vous créerez les propriétés en fonction des besoins de votre programme

Les méthodes avec les signatures suivantes sont imposées :

public bool Add\_Lettres(int nb, Lettres pioche, Random r) tire au hasard nb lettres dans la pioche

public override string ToString() ou public string toString() qui retourne une chaîne de caractères qui décrit un joueur.

public void Add(string mot) qui ajoute un mot à la liste des mots déjà trouvés

public void OteLettre(string mot)qui retire les lettres du mot des lettres tirées au hasard par le joueur

D’autres méthodes seront à créer avec les signatures à votre convenance.

Il faudra en particulier s’assurer que le mot proposé par le joueur soit bien composé des lettres en sa possession

# Exercice 2 : Classe Lettre(s)

Une Lettre est définie par son symbole et son poids qui permettra de finaliser le score à la fin de la partie.

Les lettres seront lues depuis le fichier donné en annexe.

Les lettres devront être gérées de telle manière qu’à chaque fois qu’un joueur tire une lettre au hasard, il faudra retirer cette lettre des lettres candidates.

Vous créerez les propriétés en fonction des besoins de votre programme

Les méthodes avec les signatures suivantes sont imposées :

public override string ToString()ou public string toString() qui retourne une chaîne de caractères qui décrit un plateau.

# Exercice 3 : Classe Dictionnaire

Une instance de dictionnaire va contenir l’ensemble des mots du fichier des mots. Vous réfléchirez à une structure de données qui permet de conserver la structure du fichier de mots, à savoir conserver les mots par groupe en fonction du nombre de lettres.

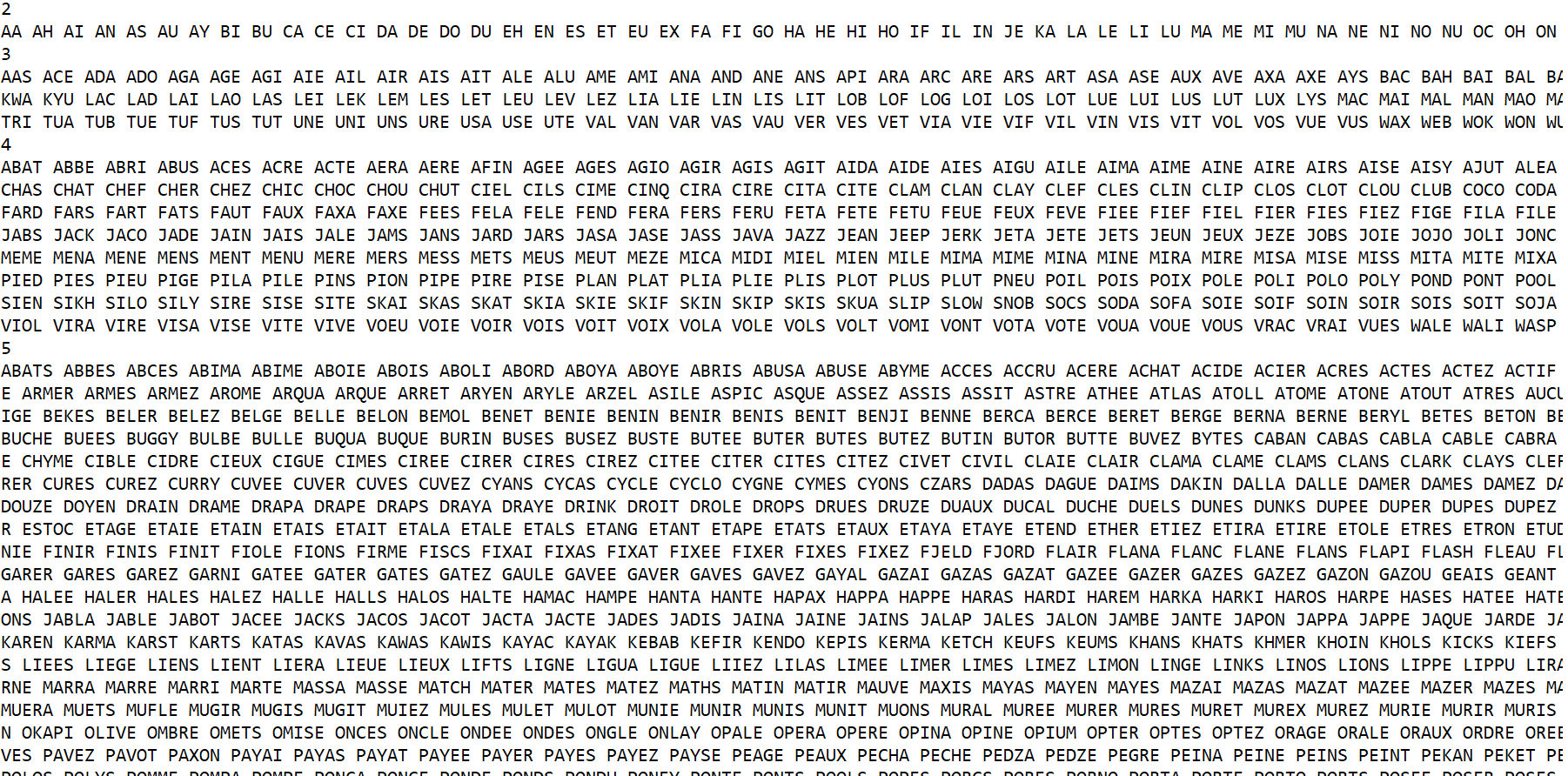
Vous créerez les propriétés en fonction des besoins de votre programme

Les méthodes avec les signatures suivantes sont imposées :

public override string ToString() ou public string toString() qui retourne une chaîne de caractères qui décrit un dictionnaire.

Vous utiliserez la méthode vue en TD public bool RechDichoRecursif(int debut, int fin, string mot) afin de tester que le mot appartienne bien au dictionnaire

Un extrait du dictionnaire



# Exercice 3 : Classe MotsCroises

Il faut vérifier

* pour chaque instance de mots croisés que les mots inscrits dans la grille soient valides c’est-à-dire appartiennent au dictionnaire (attention certains mots se créent par juxtaposition de 2 mots placés) que ce soit horizontalement ou verticalement
* il faut toujours au moins une intersection entre le mot à placer et les mots de la grille déjà placés (sauf dans le premier cas)

# Exercice 4 : Classe Jeu (ou Program)

La classe Jeu possède a minima 3 attributs :

* Un dictionnaire : Le dictionnaire sera instancié par la lecture du fichier donné en pièce jointe : Motspossibles.txt. Prenez le temps d’analyser la construction de ce fichier
* Une liste de joueurs
* Une pioche composée de lettres

Le programme principal Main va donc donner la main aux joueurs. Chaque joueur va inscrire successivement les mots qu’il trouve et sa position dans la grille

Chaque joueur a une minute maximum (voir la classe DateTime et TimeSpan) pour placer les mots sur sa grille avec ses lettres tirées au hasard s’il ne peut pas placer toutes ses lettres

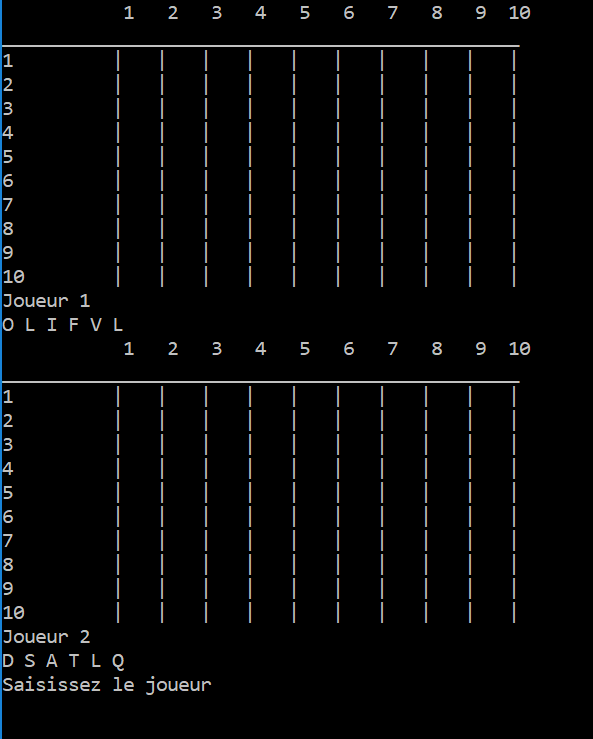
A l’issue de ce tour, les joueurs vont piocher 2 lettres.

Et le tour continue

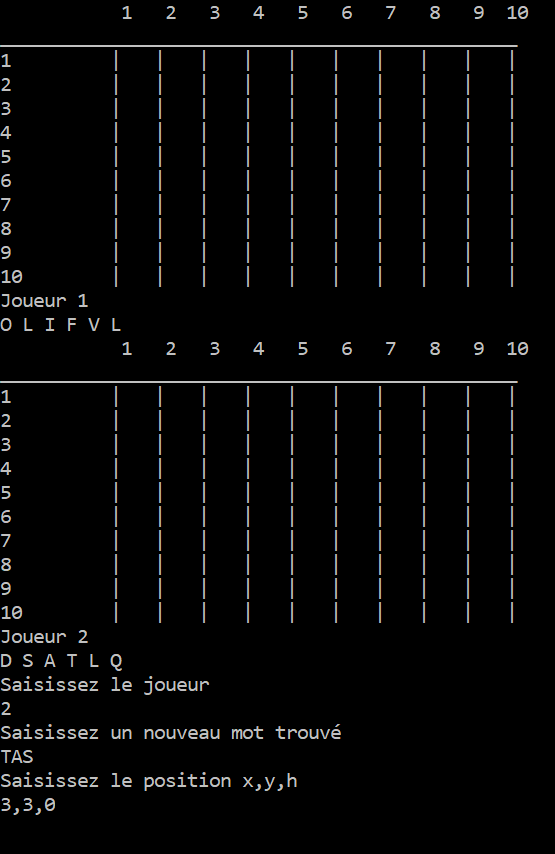
Vous afficherez à la fin de la partie (5mn) ou utilisation de toute la pioche, les scores des joueurs et indiquerez le gagnant.

Exemple de Simulation du jeu Version 0

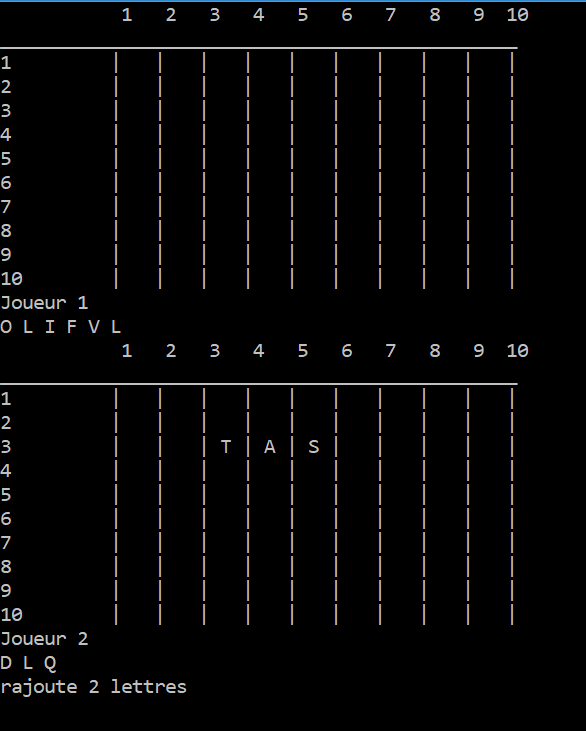
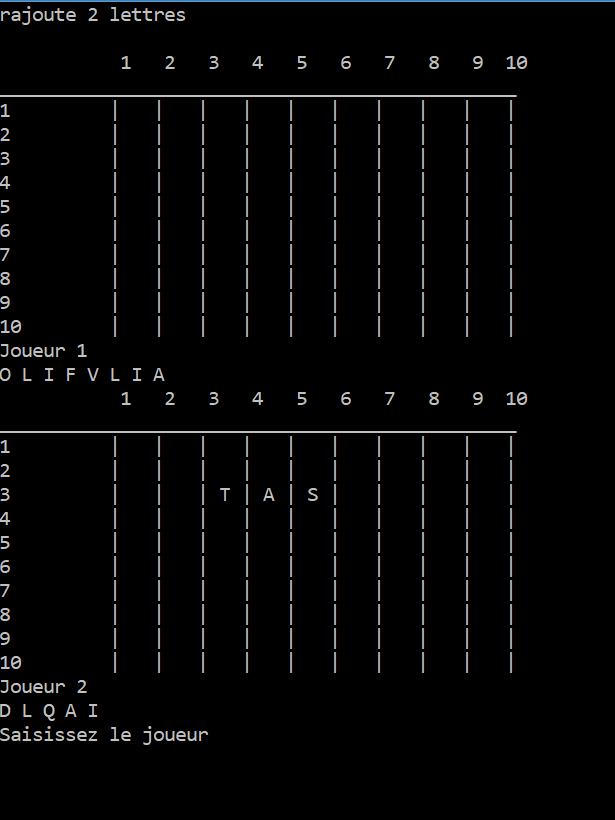
Etape 1 : Les grilles des 2 joueurs sont vides. Ils ont tirés au hasard 6 lettres chacun



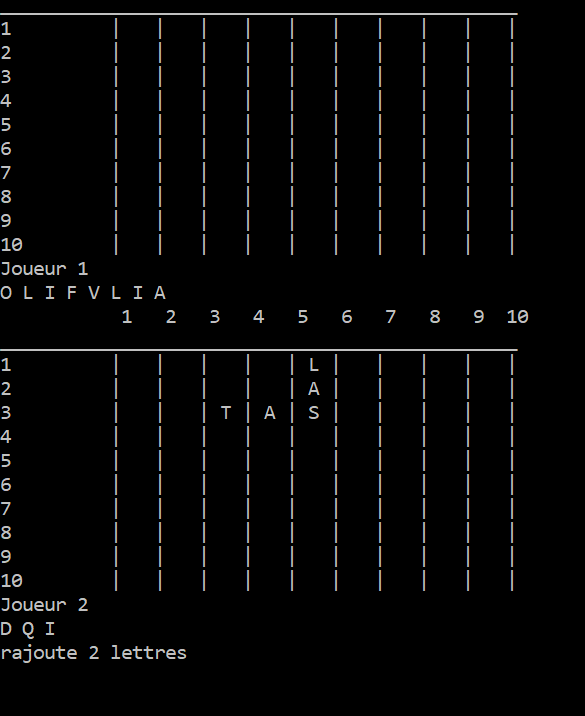
Etape 2 : Le joueur 2 propose le mot TAS à la position 3,3 et de façon horizontale (0, verticale =1)



Etape 3 : le temps s’est écoulé (1 mn) sans que les joueurs aient eu le temps de terminer leurs lettres, 2 lettres ont été affectées à chaque joueur

Etape 4 : Le joueur 2 trouve le mot LAS et le place sur sa grille

 …

Le jeu se termine soit en terminant les lettres de la pioche, soit lorsque le temps maximum est écoulé.

Le calcul du score est ensuite calculé pour chacun des joueurs.