**Classe Joueur:** La classe Joueur représente un joueur du jeu. Elle possède un nom, une liste de mots trouvés, et un score. Le constructeur initialise ces attributs. La méthode Add\_Mot ajoute un mot à la liste des mots trouvés. La méthode toString affiche les attributs du joueur. La méthode Add\_Score ajoute une valeur au score. La méthode Contient teste la présence d'un mot dans la liste. Les propriétés permettent d'accéder aux attributs en lecture ou écriture.

**Classe Plateau:** La classe Plateau représente le plateau de jeu. Elle peut être initialisée à partir d'un fichier ou générée aléatoirement. La méthode ReadFile lit le fichier ou crée une matrice aléatoire. La méthode Melanger mélange une liste pour créer une matrice aléatoire. La méthode AfficherListe affiche le contenu d'une liste. La méthode toString affiche le plateau. La méthode WriteFile crée un fichier CSV à partir de la matrice. Les méthodes ChercheMot et Cherche recherchent un mot dans la matrice. La méthode Maj\_Plateau actualise le plateau après avoir trouvé un mot. La propriété Plateaumat permet d'accéder à la matrice.

Structure Coordonees: La structure Coordonees représente des coordonnées avec les propriétés X et Y. Utilisée pour représenter les mouvements possibles et les coordonnées du mot trouvé.

**Classe Dictionnaire:** La classe Dictionnaire représente un ensemble de mots. Elle peut être initialisée à partir d'un fichier texte et offre des méthodes pour effectuer des opérations sur le dictionnaire. La méthode ReadFile convertit un fichier en liste de mots. La méthode toString affiche le nombre de mots par lettre dans le dictionnaire. La méthode RechDichoRecursif effectue une recherche binaire récursive dans le dictionnaire trié. Les méthodes Tri\_XXX, QuickSort, et Partition permettent de trier le dictionnaire. La méthode EstTriee vérifie si le dictionnaire est déjà trié. La propriété Dico permet d'accéder à la liste de mots.

**La classe Minuteur :** représente un minuteur avec des méthodes pour démarrer, mettre en pause, reprendre, mettre à jour, et vérifier si le minuteur est terminé. Ses attributs incluent le moment du démarrage, la durée totale, le temps écoulé, et des indicateurs d'état. Les méthodes permettent un contrôle flexible du minuteur. La classe offre aussi la fonction TempsRestant() pour obtenir le temps restant du minuteur.

**La classe Jeu** crée un jeu de mots croisés entre deux joueurs, utilisant un dictionnaire, un plateau et des minuteurs. Les joueurs alternent leurs tours, formant des mots à partir de la matrice de lettres. Le jeu prend en charge la gestion du temps avec des minuteurs individuels par joueur. Les méthodes incluent la logique du tour de jeu, le calcul des points pour un mot et l'application de couleurs à la console pour afficher les changements. La partie se poursuit jusqu'à ce que le temps des deux joueurs expire.

**Le programme** invite l'utilisateur à saisir un nom de fichier, permettant au joueur de choisir de jouer avec un fichier existant, un plateau aléatoire ou de quitter le jeu. Il vérifie également si le fichier existe. Si le joueur choisit de jouer avec un fichier existant, le programme vérifie si le nombre de colonnes est le même pour chaque ligne du fichier.

En cas de jeu avec un plateau aléatoire, le programme crée un plateau, un dictionnaire, et permet aux joueurs de définir la durée de jeu. Ensuite, le jeu commence, alternant entre les tours des joueurs jusqu'à ce que le temps d'un joueur expire.

La méthode Début est utilisée pour afficher une bannière de début et recueillir le nom du joueur.

La méthode MemeNbColonne vérifie si toutes les lignes du fichier ont le même nombre de colonnes.

Le programme continue à s'exécuter jusqu'à ce que l'utilisateur saisisse "sortir".