Projet BPO
La recherche de chemin

Le deuxième objectif est la réalisation de la recherche d'un chemin dans le graphe des états.

Remarque préliminaire :

- *Les mêmes conventions continuent de s'appliquer. Le diagramme de classes de référence est à la fin de cet énoncé. Attention aux packages et aux noms de classes.*

1. Écrire l'interface **`IRecherche`** et le squelette de la classe **`ProfondeurDAbord`** (c'est-à-dire juste le profil des fonctions).

   Compléter le corps de la fonction **`existeChemin`** avec l'instruction `return true` , de sorte que la compilation puisse se faire. Ajouter de la même façon un `return` dans les autres fonctions, si besoin.

   Vérifier que l'ensemble compile correctement.

2. Écrire une classe cliente **projetBPO.Main** dans le package **`projetBPO`**, avec une fonction **main** qui :

   a. crée une instance de **`ProfondeurDAbord`** et de **`JeuDeMots`**,
   b. fixe l'état final du jeu,
   c. applique l'algorithme de recherche de chemin pour savoir s'il existe un chemin,
   d. et affiche le résultat.

   En l'état actuel des choses, si tout est programmé correctement, le résultat affiché doit être **`true`**.

3. Il reste à écrire la fonction **`existeChemin`**. En TD, nous avons détaillé une version de l'algorithme en faisant l'hypothèse que le graphe ne comporte pas de circuit : pour mémoire, cet algorithme itère sur tous les états accessibles en une action à partir de l'état passé en paramètre. L'itération s'arrête dès que l'on constate que l'un de ces états mène à l'état final, en faisant un appel récursif de la fonction **`existeChemin`**.

   Pour lever l'hypothèse sur l'absence de circuit, on ajoute une fonction booléenne privée **`existeChemin(Etat e, Historique h)`**. Le résultat de cette fonction est vrai s'il existe un chemin partant de **`e`** et allant jusqu'à un état final, sans passer par un des états de l'historique ; elle est appelée dans **`existeChemin(Etat e)`**, avec **`e`** comme premier état de l'historique.

À faire avant la prochaine séance

Terminer le TP si besoin.

Diagramme de classes UML (en complément du précédent)