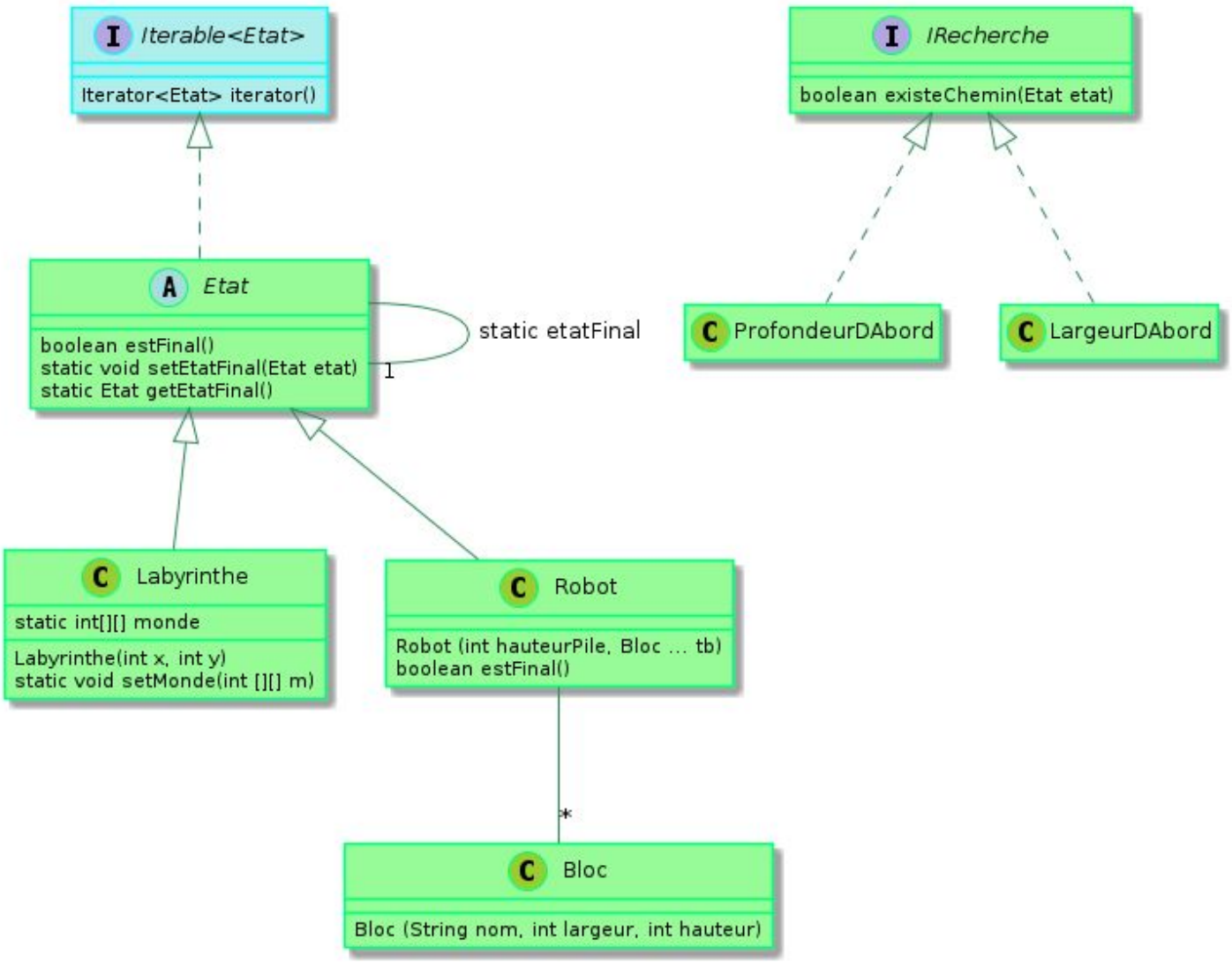
Projet BPO
Recherche (naïve) de chemin
dans un graphe d'états

Diagramme de classe



Les fonctions présentes dans ce diagramme doivent être obligatoirement programmées. Bien évidemment, vous pouvez ajouter vos propres fonctions.

1. Écrire l'interface **projetBPO.algos.IRecherche**.

2. Écrire la classe abstraite **projetBPO.jeux.Etat**. L'état final **etatFinal** est fixé par un champ statique, déclaré dans la classe **Etat**, donc commun à toutes les instances. Les fonctions **getEtatFinal** et **setEtatFinal** sont statiques. La fonction **estFinal** compare l'état courant à l'état final ; pour l'écrire on peut utiliser **equals**, à condition de redéfinir cette fonction dans les sous-classes.

3. Écrire la classe **projetBPO.algos.ProfondeurDAbord** ; la fonction ***existeChemin*** retourne vrai si l'état en paramètre est final, faux sinon.

4. Écrire la classe **projetBPO.jeux.Labyrinthe** : un état est représenté par la position du personnage ; le monde est rangé dans un tableau statique, donc commun à tous les états. La classe **Labyrinthe** propose le constructeur **Labyrinthe(int x, int y)** où **x** et **y** représentent la position courante du personnage, numérotés à partir de 0.

   Pour le moment, la fonction ***iterator*** retourne **null**.

5. Compléter le code des classes de façon à pouvoir écrire et tester une fonction ***main*** qui :
   - crée une instance de **Labyrinthe** et une instance de **ProfondeurDAbord**,
   - fixe l'état final du jeu,
   - et applique l'algorithme de recherche de chemin à partir de cet état final... pour constater qu'on peut atteindre l'état final. Ouf.

6. Écrire la fonction ***iterator*** de la classe **Labyrinthe**. Cet itérateur de **Etat** permet de parcourir tous les états accessibles par une action appliquée à l'état courant.

7. On ignore l'existence de circuit dans le graphe d'états. Écrire la fonction ***existeChemin***. Pour mémoire, cette fonction itère sur tous les états qu'il est possible d'atteindre en une action à partir de l'état passé en paramètre. L'itération s'arrête dès qu'on constate que l'un de ces états mène à l'état final.

8. Compléter les classes **Etat** et **Labyrinthe** pour que l'algorithme en profondeur d'abord puisse être appliqué à ce jeu.

   Faire des tests avec des états initiaux proches de l'état final, pour aller vite et surtout ne pas se placer dans des configurations de circuits.

9. On gère les graphes avec circuits. Dans la classe **ProfondeurDAbord**, on ajoute la fonction privée ***existeChemin(Etat etat, Historique h)***. Cette fonction recherche un chemin qui ne passe pas par les états de l'historique **h**. Cette fonction est appelée dans ***existeChemin(Etat etat)***, avec l'état **etat** comme premier état de l'historique. La classe **Historique** se trouve dans le package **projetBPO.algos**.

Compléter les tests précédents.

10. Écrire la classe **projetBPO.algos.LargeurDAbord**. Vous pouvez vous appuyer sur la page :

    http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur

    Tester sur les mêmes exemples.

11. Comparer les temps d'exécution des deux algorithmes sur le même jeu.

    Idée : la fonction System.currentTimeMillis() donne l'heure.

12. On ajoute le jeu du robot :

    > Le robot dispose de blocs de tailles et poids différents ; il doit tous les empiler sur une seule pile, mais un bloc empilé ne doit pas être de taille ou de poids supérieur à 20% de celui sur lequel il est empilé.

    Écrire les classes **projetBPO.jeux.Robot** et **projetBPO.jeux.Bloc**. Respectez les profils des fonctions donnés dans le diagramme de classe.

13. Comparer les temps d'exécution des deux types de recherche.

14. Copiez dans votre projet et à sa place la classe **projetBPO.Check** disponible sur arche :

    - la fonction ***main()*** attend trois arguments : **-l** ou **-r** correspondant au choix du jeu (labyrinthe ou robot), puis **-l** ou **-p** correspondant au choix de l'algorithme (largeur ou profondeur), puis un entier entre **1** et **2** correspondant à un numéro de test.

    - Demandez l'exécution de cette classe, par exemple :
      - **java projetBPO.Check -l -p 1**
      - **java projetBPO.Check -r - l 2**