L'algorithme d'Ariane

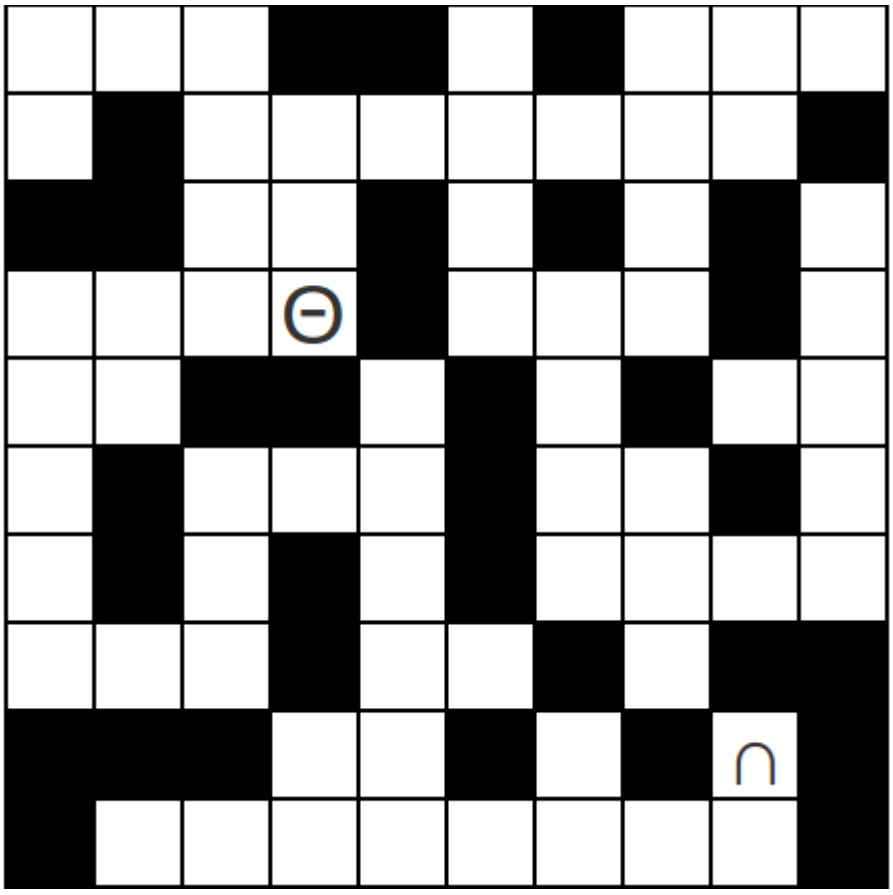
Ce projet a pour but d'étudier un algorithme de guidage, visant à conduire un objet mobile jusqu'à son but à travers un parcours d'obstacles. En hommage à la mythologie grecque, nous supposerons qu'il s'agit de Thésée, perdu dans le labyrinthe crétois, à la recherche de la sortie (on aurait pu aussi choisir une souris recherchant un morceau de fromage, ou un robot cherchant son point de ravitaillement).

Vous produirez un programme, écrit en Java, sans emprunt extérieur (sauf l'API officielle), et accompagné d'un rapport (voir annexe II). Le travail sera fait seul ou en binôme (un binôme est fortement recommandé, afin de vous permettre de vous familiariser avec les techniques de développement collaboratif). Il devra impérativement être rendu avant le **lundi 13 mai 2019 à 18h00**.

La partie logicielle sera développée *dès le départ* dans un dépôt dédié du serveur GoGS du département. Le rapport prendra la forme d'un fichier au format PDF joint aux sources.

Principes généraux

Pour simplifier le problème, le labyrinthe sera représenté sous la forme d'une grille carrée. Chaque cellule de la grille peut être bloquée ou libre. La position initiale de Thésée ainsi que celle de la sortie pourront être placées sur n'importe quelles cases libres distinctes.



À chaque étape de la simulation, Thésée a quatre options :

- Se déplacer d'une case vers le nord.
- Se déplacer d'une case vers le sud.
- Se déplacer d'une case vers l'est.
- Se déplacer d'une case vers l'ouest.

Ce mouvement peut avoir trois conséquences :

- La case de destination est bloquée (et Thésée n'a donc pas changé de case).
- La case de destination est libre (et Thésée s'y est rendu).
- La case de destination est la sortie (et la simulation est terminée).

L'algorithme que vous devez écrire dictera à Thésée son prochain déplacement à chaque étape.

Fonctionnalités demandées

Le programme se déroulera en trois parties : choix de la grille, choix de l'algorithme, puis test de l'algorithme.

1. Pour choisir la grille, l'utilisateur pourra charger une grille existante (en sélectionnant à l'aide d'un JFileChooser, un fichier au format décrit dans l'annexe I), ou construire une nouvelle grille.
2. S'il choisit cette dernière option, l'utilisateur décidera d'abord de la taille de la grille. Il pourra ensuite partir d'une grille vide, ou d'une grille remplie aléatoirement. Il sera alors capable de modifier individuellement l'état de chaque case, puis décidera de la position initiale de Thésée et de la sortie.
3. Une fois la grille terminée, l'utilisateur aura la possibilité de la sauvegarder (sous un nom de son choix) au format décrit dans l'annexe I.
4. L'utilisateur pourra ensuite choisir entre deux algorithmes : l'un, totalement aléatoire (chaque direction a autant de chances d'être choisie), et l'autre déterministe.
5. Ce dernier devra baser ses décisions uniquement sur les coordonnées actuelles de Thésée et sur sa mémoire des conséquences de ses actions précédentes. Il n'aura aucune connaissance préalable de l'emplacement des cases obstruées ou de la sortie.
6. Une fois l'algorithme et la grille choisie, l'utilisateur devra sélectionner une visualisation du déroulement de la simulation : manuelle ou automatique.
7. Dans le mode manuel, l'avancement de Thésée sera représenté sur la grille, et le choix préconisé par l'algorithme pour la prochaine étape sera affiché. L'utilisateur fera défiler les étapes en appuyant sur une touche, jusqu'à la fin de la simulation. Ce mode sert à voir en détail le comportement de l'algorithme.

8. Dans le mode automatique, la grille n'est pas affichée, et la simulation est effectuée sans intervention de l'utilisateur. On affichera seulement le nombre d'étapes nécessaires pour compléter la simulation. Dans le cas de l'algorithme aléatoire, on fera tourner la simulation 100 fois avant d'afficher le nombre d'étapes moyen (puisque le résultat sera différent à chaque simulation).
9. Vous vous efforcerez d'écrire l'algorithme déterministe le plus efficace possible. Il devra impérativement être capable de trouver la sortie si un chemin existe (vous devrez en fournir la preuve dans le rapport), et ses performances (en termes de nombre d'étapes nécessaires pour trouver la sortie) devront être visiblement meilleures que celles de l'algorithme aléatoire.

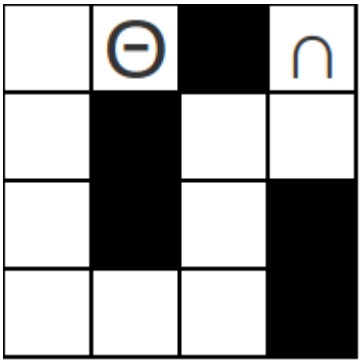
Toutes les interactions devront impérativement être en mode graphique. Les seuls affichages permis à la console sont les messages envoyés à la sortie sur erreur afin de documenter les problèmes rencontrés.

Annexe I : format de sauvegarde des grilles

Le fichier ne contiendra pas de texte. Le premier octet servira à stocker la taille de la grille. Le second et le troisième stockeront respectivement les numéros de ligne et de colonne de la position initiale de Thésée, le quatrième et le cinquième octet jouant le même rôle pour la position de la sortie.

A partir de ce point, chaque bit représente l'état d'une case : 0 si elle est libre, ou 1 si elle est bloquée. La grille est ainsi décrite colonne par colonne.

Exemple :



Cette grille correspond au fichier suivant :

```
00000100 00000000 00000001 00000000 00000011 0000011010000011
```

Annexe II : sources

Les sources de votre projet (et **pas** les fichiers générés .class) devront être disponibles à tout moment sur le serveur GoGS du département. Votre dépôt sera privé, nommé obligatoirement PT21_APL2018 et inclura Luc Hernandez (login : hernand) dans la liste des collaborateurs. N'attendez pas la date limite pour profiter de ce service !

Pour chaque classe, vous prévoirez un fichier source. Suivez-les consignes habituelles scrupuleusement. La définition de chaque classe et de chaque membre de classe sera précédée d'un commentaire formaté pour permettre la génération de documentation technique par l'outil javadoc (ceci est un exemple de fichier source bien commenté).

Un fichier Makefile devra permettre la compilation de votre projet (par la commande make) ainsi que son exécution (par la commande make run). Transcrivez bien toutes les dépendances entre vos fichiers dans les règles. La commande javac fait déjà (très mal) un travail similaire à make, ce qui peut créer des interférences. Pour éviter cela, pensez à lui passer l'option -implicit:none.

Annexe III : rapport

Le rapport d'avancement prendra la forme d'un fichier PDF disponible avec les sources sur le serveur Gogs. Vous y inclurez en particulier :

- une introduction contenant une brève description du sujet.
- la description des fonctionnalités de votre programme, aidée de captures d'écran.
- une présentation de la structure du programme, avec diagramme de classes à l'appui.
- une exposition de l'algorithme déterministe choisi, illustrée par un algorigramme (ou diagramme d'activité) et complétée par une démonstration prouvant que l'objectif est toujours atteint lorsqu'il est accessible.
- une conclusion personnelle pour chaque auteur.

Soignez la présentation ! L'orthographe, la grammaire, les pages de garde, la table des matières, les en-tête et pieds de page ne sont pas en option...

Notez bien que le rapport ne doit pas contenir d'extrait du code source du projet, étant donné que le correcteur peut aller le consulter directement s'il en éprouve le besoin. N'hésitez pas en revanche à illustrer vos propos par des schémas. Ceux-ci peuvent être construits directement dans le logiciel de traitement de texte s'il le permet, ou dans un logiciel dédié, tel que Inkscape ou Draw (tous deux gratuits).