Carte d'un antre sousterrain

1 INTRODUCTION

Pour le premier TP, vous allez construire un programme généreant la carte d'un antre sousterrain. Ce programme va lire de l'information dans un fichier décrivant la taille des pièces à placer dans le sousterrain et ensuite placer ces pièces aléatoirement. Le résultat sera affiché à l'écran, à l'aide de caractères.

La prochaine section va décrire le programme et la dernière section va décrire les directives pour ce logiciel.

2 DESCRIPTION

Le logiciel va demander le nom du fichier contenant la description des pièces. Ensuite il va générer la carte et finalement l'afficher.

2.1 ENTRÉES

Le logiciel va afficher une question pour que l'utilisateur entre le nom du fichier. L'utilisateur va entrer le nom du fichier au clavier. Le nom du fichier ne contiendra pas de répertoire et va contenir l'extension (.txt). Par exemple : « niveau1.txt ». Vous n'avez pas a vérifier que le nom est valide, seulement que le fichier existe. Votre programme doit utiliser le chemin par défaut de la machine virtuelle de Java. Vous ne devez pas ajouter de chemin au nom du fichier. La machine virtuelle va automatiquement lire le fichier dans le répertoire du projet.

Le fichier va simplement contenir des valeurs entières positives. Il y aura deux valeurs par ligne. La première ligne va contenir la taille de la carte : la base suivit de la hauteur. La deuxième ligne va contenir la taille de la pièce centrale du sousterrain. Les autres lignes vont contenir les tailles des autres pièces, que votre programme doit essayer de placer sur la carte. Il n'y a pas de limite sur le nombre de pièces supplémentaires.

```
50 30
8 6
4 1
2 5
1 1
5 3
9 1
8 4
```

Figure 1 : Exemple de fichier d'entrèes

2.2 CONSTRUCTION DE LA CARTE

La taille de la carte vous indique le nombre de colonne (base) et le nombre de ligne (hauteur) utilisées pour construire la carte. L'algorithme va essayer de placer les pièces dans cet espace. Il est conseiller de construire une matrice pour représenter la carte complète. Chaque case de la matrice représente un espace de la carte, disons 1 mètre carré. Cette espace peut être soit un espace vide (intérieur d'une pièce), soit une porte (une porte va occuper 1 mètre carré 😊), soit un espace plein (terre, c'est une carte d'un antre sousterrain).

La deuxième ligne du fichier contient la taille de la pièce de départ. Cette pièce est placé au milieu de la carte. S'il n'est pas possible de la placer directement au milieu, elle peut être arrondissez vers le bas.

2.2.1 Algorithme de base

Pour construire le reste de la carte, nous allons utiliser une technique simple. Vous devez construire une structure (ArrayList peut être) qui va contenir la liste des murs sur la carte qui n'ont pas encore de porte. Par exemple, au début, il y a une pièce au centre. Cette pièce à 4 murs, chaque murs peut avoir plusieurs case de long. Ces quatre murs seront placés dans la liste.

Ensuite, vous choisissez un mur libre au hasard et une pièce qui n'a pas été placée (au hasard aussi). Lorsque le mur est choisi, il faut choisir une case de ce mur pour placer la porte (au hasard) et choisir une case au hasard dans la nouvelle pièce, pour la même porte. Enligner les deux pièces sur la porte et vérifier que la nouvelle pièce n'est pas sur une pièce existante. Si elle est sur une pièce alors votre algorithme va essayer de la déplacer. La description de ces déplacements est dans la section 2.2.2. Si l'algorithme ne réussi pas à placer la pièce, alors un notre choix aléatoire est fait : nouveau mur, nouvelle pièce, nouvelle position de la porte.

Si la pièce peut être placé, alors le mur qui à été choisie est enlevé de la liste de mur et les nouveaux murs libre sont placés dans la liste. Aussi, la nouvelle pièce ne peut plus être choisie. Chaque pièce est placée une seule fois sur la carte.

2.2.2 Déplacement d'une pièce

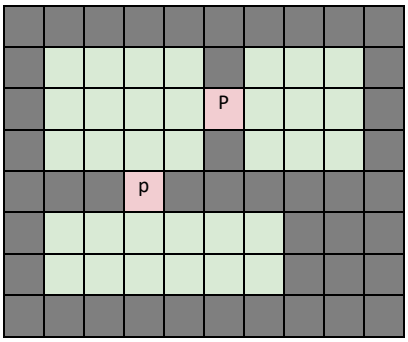
Lorsqu'une pièce est par dessus d'une autre pièce dans le plan, votre algorithme doit essayer de la déplacer pour qu'elle ait de la place. Par exemple, si nous essayons de placer une pièce B à droite d'une pièce C. Nous essayons de monter ou descendre la pièce B pour qu'elle ne soit plus par dessus l'autre autre pièce. Cela va déplacer la position de la porte dans la pièce B. La position de la porte dans la pièce C ne peut pas être déplacé.

Donc, si la pièce à placer est à gauche, ou à droite, alors nous déplaçons la pièce vers le haut ou vers le bas. Si la pièce à placer est en haut, ou en bas, alors nous déplaçons la pièce vers la gauche ou vers la droite.

Si la pièce ne peut pas être placé, même avec un déplacement, c'est un **échec** pour cette essaie. L'algorithme essaye avec une autre pièce (voir plus haut).

2.2.3 Fin de l'algorithme

L'algorithme termine si toutes les pièces sont placées, ou s'il y a eu 100 **échecs**.



Dans cet exemple, Il y a trois pièce (vert), 4 x 3, 3 x 3, 6 x 2. Il y a deux portes. Remarquez qu'il y a une largeur de 1 case qui sépare les pièces (mur). Deux pièces choisies pour être voisines, doivent avoir un mur entre elles. Aucune pièce ne peut se toucher directement.

2.3 SORTIE

Finalement, le programme doit afficher la carte, ligne par ligne, à l'écran. Une case vide est affichée avec une '.', une case pleine avec un 'O' (lettre 'o' majuscule). Une porte horizontale utilise un tiret '-' et une porte vertical utilise une bar verticale '|'.

```
OOOOOOOOOO
O....O...O
O....|...O
O....O...O
OOO-OOOOOO
O......OOO
O......OOO
OOOOOOOOOO
```

3 DIRECTIVES

Les sections suivantes décrivent les attentes et les éléments d'évaluation pour le devoir.

3.1 DIRECTIVES POUR LA CONSTRUCTION DU PROJET.

- Placez vos noms au début du fichier `Principal.java`.
- Lorsque vous ajoutez une méthode, vous devez l'ajouter dans la classe appropriée.
- Des tests vous seront donnés avec les résultats attendus.

3.2 DIRECTIVES POUR L'ÉCRITURE DU CODE.

1. Le TP est à faire seul ou en équipe de deux.
2. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaitre que dans les `switch`.

c. Un seul `return` par méthode.
d. Additionnez le nombre de `if`, `for`, `while`, `switch` et `try`. Ce nombre ne doit pas dépasser 7 pour une méthode.
e. Construisez des classes au besoin.

3. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
4. Commentaires
   - Commentez l'entête de chaque classe et méthode.
   - Une ligne contient soit un commentaire, soit du code, pas les deux.
   - Utilisez des noms d'identificateur significatif.
   - Une ligne de commentaire ou de code ne devrait pas dépasser 120 caractères. Continuez sur la ligne suivante au besoin.
   - Nous utilisons Javadoc :
     - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
       - Courte.
       - Complète.
       - Commencez la description avec un verbe.
       - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
     - Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
       - Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.
       - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
     - Ensuite, avant de placer les **tags,** placez une ligne vide.
     - Placez les **tag** @param, @return et @throws au besoin.
       - @param : décris les valeurs acceptées pour la méthode. Vous devez commenter les paramètres de vos méthodes.
     - Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
     - Écrivez les commentaires à la troisième personne.

3.3 Remise

Remettre le TP par l'entremise de Moodle. Placez vos fichiers `'*.java' seulement`, pas de sous répertoire dans un dossier compressé de **Windows**, vous devez remettre l'archive. Le TP est à remettre avant le 16 février 23 :59.

3.4 Évaluation

- Fonctionnalité (8 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre TP. Votre projet doit compiler sans erreur pour avoir ces points.
- Structure (2 pt) : veillez à placer correctement vos méthodes et à utiliser correctement le mécanisme d'héritage.

- Lisibilité (2 pts) : commentaire, indentation et noms d’identificateur significatif.