SCIENCES SORBONNE UNIVERSITÉ

TME 11

Éléments de programmation par objets avec Java

Présentation

Ce TME 11 a pour but la réalisation d'un programme mettant en œuvre la plupart des points du cours vus ce semestre. Il représente un travail devant prendre environ 2 à 3 heures. Le travail demandé est de proposer et d'implémenter un programme qui réalise une simulation se déroulant sur une grille représentée sous la forme d'un tableau à 2 dimensions dont le contenu doit être précisé.

Cette simulation a pour cadre la collecte de joyaux (diamant, rubis, opale,...) dans une grille où se trouve des gardiens. Joyaux et gardiens sont des `Contenu` qui se trouvent dans les cases de la grille. Les joyaux ont un prix évalué en nombre de pièces d'or (entre 1 et $X * 1000$ où X est votre numéro de groupe de TDTME). Les gardiens ont un nombre de points de vie (entre 0 et 200).

Le programme à réaliser doit respecter le cahier des charges suivant :

- la simulation se déroule sur une grille représentée à l'aide de la classe `Grille` qui est fournie.
- la simulation gére des `Contenu` (classe qui est aussi fournie) placés sur la grille, ces contenus doivent être placés dans une case (et une seule) ou retirés.
- votre simulation doit définir au moins 2 types de contenus. Les joyaux et les gardiens peuvent donner lieu à plusieurs classes selon des propriétés que vous voudrez leur donner, c'est à vous de choisir ce que vous souhaitez ajouter.

Les classes `Grille` et `Contenu` sont fournies (voir Annexe), accompagnées de leur documentation et de leur représentation UML client.

Important, certaines méthodes de la classe `Grille` peuvent lever une exception. Il y a 2 types d'exception possibles : `CoordonneesIncorrectesException` et `CaseNonPleineException`. Les classes de ces 2 exceptions sont représentées dans le diagramme UML donné en Annexe, mais c'est à vous de les écrire.

Feuille de route

1. Comprendre comment utiliser les classes `Grille` et `Contenu` en étudiant la documentation fournie et en expérimentant avec la classe `TestGrille` aussi fournie. Pour cela, les 2 classes d'exception doivent être définies avant tout.
2. Créer au moins 2 classes qui étendent la classe `Contenu`.
3. Donner le code la classe `AgentX` où `X` est le numéro de votre groupe de TD (donc `Agent1` pour le groupe 1, etc.). Cette classe possède deux attributs qui correspondent à la position de l'agent sur le terrain (numéros de ligne et de colonne), ainsi qu'un sac de joyaux qui est vide au départ. Remarque : les agents ne sont pas placés dans la grille, ils parcourent la grille en se déplaçant. Cette classe doit contenir les 4 méthodes suivantes (ainsi que tous les attributs et méthodes que vous jugerez utile de rajouter pour votre simulation) :
   - `seDeplacer(xnew,ynew)` qui change la position de l'agent sur la grille et le place en (xnew, ynew) si le déplacement est possible, sinon une exception `DeplacementIncorrectException` doit être levée. Il est nécessaire de définir vous-même cette exception. Une fois que l'agent arrive dans une case :
     - si la case où il arrive contient un joyau, il le ramasse (le joyaux est donc retiré de la grille et il le met dans son sac à joyaux.
     - si la case est occupée par un gardien, il perd tous les joyaux de son sac (qui est donc vidé).
   - `seDeplacer(xnew,ynew,f)` qui change la position de l'agent courant sur le terrain et le place en (xnew, ynew) si le déplacement est possible sinon une exception `DeplacementIncorrectException` doit être levé. Le troisième argument est un entier `f` (la force) qui est utilisé de la façon décrite ci-dessous. Une fois que l'agent arrive dans une case :

  - si la case où il arrive contient un joyau, il le ramasse.
  - si la case est occupée par un gardien, si son nombre de point de vie est inférieur ou égal à f, alors le gardien est retiré de la grille. Par contre, si le gardien possède un nombre de points de vie strictement supérieur à f alors l'agent perd tous les joyaux de son sac et le gardien voit son nombre de points de vie baisser de f points.
- `fortune()` qui donne la valeur en pièces d'or du sac de joyaux de l'agent.
- `contenuSac()` qui donne l'ensemble des joyaux contenus dans son sac.

4. Écrire la classe `Simulation` qui contient un agent, une grille, et un tableau de contenus. Cette classe possède un constructeur qui place aléatoirement m contenus sur la grille. Elle contient aussi une méthode `toString` qui permet d'avoir le contenu de la grille et les informations sur l'agent (position, fortune) et qui sera utilisée pour afficher l'état de la simulation. Cette classe doit contenir une méthode `lance(nbEtapes)` qui prend un entier naturel en argument et réalise le travail suivant :
   (a) choisir le déplacement de l'agent (avec ou sans force) par un choix aléatoire : une case juste à côté de sa position où il se trouve est choisie et l'agent à 30% de chance de faire un déplacement avec force (la valeur f est alors générée aléatoirement entre 10 et 100)
   (b) réaliser le déplacement de l'agent ;
   (c) afficher des informations sur ce qui s'est produit durant l'étape ;
   (d) recommencer à l'étape (a) un nombre `nbEtapes` de fois.
5. Écrire la classe `TestSimulation` contenant un `main()`, dans laquelle des essais de simulation sont effectués et qui produit un log à l'écran avec les différentes actions réalisées et leurs résultats. Les logs seront fournis dans le compte-rendu en les copier-collant dans le fichier à rendre.
6. Ajouter **au choix** l'une des fonctionnalités suivantes (et 1 seule) à votre programme :
   - la capacité d'écrire le log dans un fichier dont le nom est donné au lancement du programme. Votre fichier log doit avoir comme première ligne vos nom(s), prénom(s) et numéro de groupe de TDTME ;
   - la capacité de lire un fichier (dont le nom est donné en argument au lancement) ascii dont chaque ligne donne le type, la position, et le prix ou les points de vie des contenus à mettre sur la grille. les dimensions de la grille peuvent être données en argument du programme.
   - au moins 2 types d'agents ayant des stratégies de déplacement : complètement aléatoire, choix d'une direction, etc.
   - définir une interface `Teleportable` qui peut être implémentée par une sous-classe de la classe `Gardien` qui permet à un gardien de se téléporter dans une autre case de la grille (le choix de la téléportation se fait selon un % donné sur le modèle du choix de la force pour les agents).
   - toute fonctionnalité qui vous semble intéressante et dans l'esprit de cette simulation...

Modalités et compte-rendu

Ce TME 11 est à réaliser seul ou en binôme (2 étudiants max.) d'un même groupe de TDTME.

Remarque : dans le cas d'un travail en binôme, une seule remise sera faite sur un des 2 comptes Moodle, **les noms des 2 membres du binôme et le numéro du groupe doivent apparaître en commentaire dans toutes les classes écrites**.

A la fin de la séance de TME 11 de votre groupe, vous devez remettre les réponses aux points 1 à 4 de la feuille de route, c'est à dire les fichiers java des classes `AgentX`, des exceptions, et la version de base de la classe `Simulation`.

Le compte-rendu final sera à rendre au plus tard le vendredi 15 décembre, 18h. Il sera composé d'un seul fichier archive (zip ou tar) contenant les fichiers suivants :

- un fichier PDF basé sur le fichier libreoffice `description.odt` (ou sa version word) complété avec les informations demandées. Ce fichier se trouve dans l'archive `LU2IN002-2023-tme11.tgz`. Toutes les classes écrites dans le projet seront mises (copier/coller) dans ce fichier.
- un répertoire avec tous les fichiers java écrits pour ce projet (et seulement les fichiers java).

Annexe : classes fournies

Trois classes sont fournies :

- la classe `Grille` pour gérer une grille sous la forme d'un tableau d'éléments. Remarque : cette classe est finale.
  La grille est représentée dans cette classe par un tableau à 2 dimensions de `Contenu`.
- la classe `Contenu` qui sert à définir le contenu d'une case de la grille.
  Un contenu est caractérisé par un type et une quantité. Le type est mémorisé sous la forme d'une chaîne de caractères (par exemple : "Cèpe" ou "Pin") donnée lors de la création du contenu. La quantité est donnée sous la forme d'un entier naturel.
- la classe `TestGrille` qui fournit un exemple d'utilisation des 2 classes précédentes.

Les classes `Grille` et `Contenu` ne sont fournies que sous forme compilée (format .class). Leur documentation est fournie dans le répertoire `doc/` de l'archive `LU2IN002-2023-tme11.tgz`. Dans ce répertoire, il faut ouvrir le fichier `index.html` avec un navigateur internet.

Pour utiliser les 2 classes `Grille` et `Contenu`, leurs fichiers class doivent être mis dans le même répertoire que les fichiers java qui les utilisent.

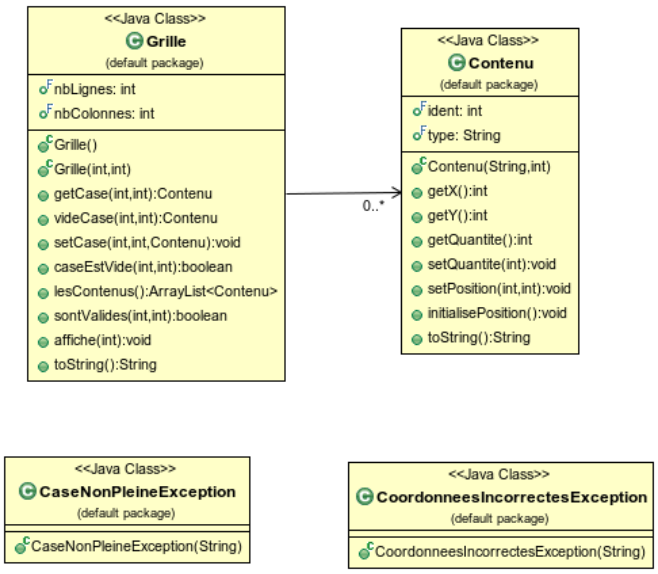

FIGURE 1 – Diagramme des classes (vision client) : Grille et Contenu sont fournies.

Annexe : exemple d'exécution

Voici ce que donne l'exécution de la classe `TestGrille` :

```
:--:--:--:--:--:
|  |  |  |  |  |
:--:--:--:--:--:
|  |  |  |  |  |
:--:--:--:--:--:
|  |  |  |  |  |
:--:--:--:--:--:
|  |  |  |  |  |
:--:--:--:--:--:

Informations sur la grille:
Grille de 4x5 cases: toutes les cases sont libres.
Erreur: coordonnees (7, 41) incorrectes !
```

```
Ajout de Cepe[id:0 quantite: 5]  en position (2, 3) valide !
:------:------:------:------:------:
|      |      |      |      |      |
:------:------:------:------:------:
|      |      |      |      |      |
:------:------:------:------:------:
|      |      |      |Cepe  |      |
:------:------:------:------:------:
|      |      |      |      |      |
:------:------:------:------:------:

Informations sur la grille:
Grille de 4x5 cases: il y a une case occupée.
Dans la case (1,4): null
Liste de tous les Contenus présents actuellement:
Cepe[id:0 quantite: 5]  en position (2, 3)
```