Projet pirates

Antoine Mosleh, Yvan Thai, Bastien Zalugas

Projet du cours de java avancé du S5

1 Table des matières

2 Lien du github

`https://github.com/AntoineMosleh/Projet_Java_S5`

3 Contexte

Après chaque pillage de navire, le capitaine d'un bateau pirate se sert en premier, puis répartit le reste du butin entre les membres de son équipage. Parmi toutes les richesses qu'il n'a pas gardées pour lui, il laisse un objet précieux à chaque pirate. Pour apaiser les tensions sur son navire, il demande à chaque pirate de lui donner ses préférences sur les objets à partager, et il essaye de les respecter au mieux. En plus de leurs préférences, il tient compte des relations au sein de l'équipage. Certains pirates s'apprécient peu, et s'il y a des sources de jalousies entre eux, cela pourraitmener à des disputes ou une mutinerie... Le capitaine pirate nous demande de l'aider, en développant un logiciel qui permet :

1. *de représenter les pirates et les relations entre eux;*
2. *de simuler le partage des ressources entre eux;*
3. *de calculer le coût d'une solution (le nombre de pirates jaloux), et de le minimiser.*

Nous réaliserons ces tâches de façon incrémentale, afin d'avoir à la fin du semestre un logiciel fonctionnel.

4 Utilisation du logiciel

Dans le dossier principal, taper dans un terminal la commande `make run` Si vous ne pouvez pas utiliser de Makefile :

1. compilez avec la commande

   `javac -d bin -cp src src/up/mi/yt_am_bz/partage_butin/*/*.java`

2. Lancement du programme

   - Cas classique : lancez le programme avec la commande
     `java -cp bin up/mi/yt_am_bz/partage_butin/programme.Main`
   - Deuxième possibilité : comme la première mais en ajoutant directement en argument le fichier à importer
     `java -cp bin up/mi/yt_am_bz/partage_butin/programme.Main nom_fichier`

5 Précisions internes au code

5.1 Premier rendu

5.1.1 Représentation des objets

- Pirates avec lettres de l'alphabet (26 pirates max au début)
- Trésors numérotés de 1 à n (n = nb de pirates)

5.1.2 Implémentation algorithme naïf de résolution du problème

5.1.3 L'utilisateur peut modifier la solution proposée pour essayer de faire une solution optimale

5.2 Second rendu

5.2.1 Représentation des objets

- Les pirates ont désormais des **noms quelconques** (représentés par des String).
  - Nous avons ajouté dans la classe Pirate un **attribut représentant l'ordre d'enregistrement du pirate dans l'équipage** afin de pouvoir gérer la matrice d'adjacence des pirates de la même manière que dans le premier rendu.
- **Les trésors sont maintenant une classe** et possèdent des **noms quelconques** aussi.

5.2.2 Implémentation d'un algorithme d'approximation d'une solution

Il nous a été demandé d'implémenter un algorithme permettant de se **rapprocher** d'une solution optimale pour l'équipage. Nous avons commencé par implémenter l'algorithme proposé dans le sujet puis nous l'avons amélioré.

En effet, l'algorithme proposé choisit **deux pirates** de manière aléatoire et **échange leurs objets**. Ensuite, il **calcule le cout de cette nouvelle solution par rapport à la meilleure solution déjà trouvée** et **si cette nouvelle solution est meilleure alors elle devient la solution de référence**. Le problème, c'est qu'il était possible de **tester plusieurs fois la même solution**, et comme il y a un nombre limité d'essais, cela "grillait" des essais avec potentiellement plusieurs fois la même solution testée.

Nous avons donc amélioré l'algorithme en faisant en sorte qu'une solution ne soit testée qu'une seule fois, sauf si l'aléa ne trouve pas assez vite une nouvelle solution.