Programmation Java
Ninon Devis
Philippe Esling

Mystic tarot

Lire votre futur en Java

2020-11-09
ninon.devis@ircam.fr
philippe.esling@ircam.fr

Le but du projet **Mystic tarot** est de créer un logiciel avec interface graphique permettant de lire les cartes du tarot divinatoire. Dans un premier temps, nous modéliserons un système de carte, permettant d'ajouter, modifier et chercher dans notre collection. Par la suite nous proposerons une interface graphique permettant d'effectuer toutes ces opérations. Enfin nous implémenterons le système de divination.

1. Modalités d'évaluation

Date de rendu ..... 10 décembre 2020
Groupes ..... Effectuer le projet *seul* et rendre par dépôt `GitHub`

Système de notation

Rendu attendu

Code source (`Java`) 50%
Rapport (`LaTeX`) 20%
Documentation (`Javadoc`) 20%

Points bonus

Code et documentation en anglais ..... **+1**
Rapport en anglais ..... **+1**

Distribution des points

Système de carte basique ..... **4 pts**
Extension et recherche ..... **6 pts**
Sauvegarde ..... **3 pts**
Interface graphique ..... **5 pts**
Lecture du futur ..... **4 pts**

Critères d'évaluation

Clarté du code (découpage en petites méthodes, bon nommage des variables) .....
Redondance du code (pas de répétition). .....
Extensibilité (ajout facile de fonctionnalités). .....
Justesse du code. .....

2. Préliminaires

Quelques notes générales à suivre pour le projet.

- Ne vous sentez pas obligés d'utiliser le polymorphisme ou l'héritage.
  Ce sont des outils qu'il faut garder en tête.
  Une mauvaise utilisation ou non-justifiée est identique à une absence d'utilisation justifiée.
- Préférez implémenter moins de fonctionnalités mais bien.
- Facilitez la vie de l'utilisateur final
  Si vous hésitez entre deux choix, préférez celui qui est ergonomique pour l'utilisateur.
  Si ils sont équivalents, choisissez ce qui est le plus simple à implémenter et relire.

1) Mailing list

Une *mailing list* est un email auquel vous pouvez vous abonner. A chaque fois qu'un utilisateur envoie un email sur cette liste, tous les abonnés reçoivent le mail. Pour vos questions concernant ce projet, ou le cours en général, vous pouvez utiliser la mailing list mise en place grâce aux commandes suivantes:

- **Inscription** : Envoyer un mail a `java-l3-2020+subscribe@googlegroups.com`
- **Désinscription** : Envoyer un mail a `java-l3-2020+unsubscribe@googlegroups.com`
- **Envoyer un message** : Comme d'habitude mais en utilisant l'adresse `java-l3-2020@googlegroups.com`

Vous pouvez répondre directement à l'adresse du groupe, et n'hésitez pas à vous entraider sur celui-ci.

2) Gestionnaire de version

Pour ce projet, nous vous demandons d'utiliser un gestionnaire de version. Comme vu en cours, nous allons utiliser le service de `github`. Nous rappelons ici les étapes principales à suivre pour celui-ci.

1. Créer un compte sur `http://www.github.com`.
2. Vous pouvez dès lors créer un répertoire à versionner.
3. Lors de la création, ajoutez une licence a votre projet (MIT ou LGPL).
4. Envoyer l'adresse de ce répertoire par mail à `esling@ircam.fr`, vous serez noté grâce à celui-ci.

Vous pouvez installer sur Windows l'exécutable `http://msysgit.github.io/`. Nous rappelons ici quelques commandes principales a connaître, mais vous pouvez suivre un tutoriel plus fourni, comme par exemple `http://git-scm.com/docs/gittutorial`

Cloner un répertoire Permet de télécharger un dépôt, créer le dossier associé, et mettre en place le système de versionnage

git clone `https://github.com/esling/atiam_ml.git`

Ajouter un fichier au projet

git add Tarot.java

Faire un commit l'option `-a` met à jour tous les changements des fichiers qui ont été ajoutés.

git commit -a -m Ïnitial commit"

Upload des changements locaux (via commit sur le serveur distant)

git push

Download des changements du serveur sur votre machine locale. A priori, seulement utile si vous travaillez sur plusieurs machines ou à plusieurs (pour synchroniser votre pc portable et le pc fixe).

git pull

Il y a des plugins spécifiques pour chaque IDE qui incorporent cet outil directement. Nous vous conseillons néanmoins de vous familiariser d'abord avec la ligne de commande pour saisir les concepts généraux de `git`. Ensuite vous devez respecter ces règles

- Effectuer des petits commit réguliers (un pour chaque fonctionnalité et résolution de bugs).
- Le message dans un commit doit être clair, explicite et relativement court.
- Vous ne devez effectuer un commit que sur du code compilable (pas d'erreur).
- Faites fréquemment des push, au moins une fois par jour, utile en cas de crash et pour les retours.
- Vous ne devez pas versionner les binaires (pas de .class ou .jar par exemple).
- Soyez cohérent, si vous codez en anglais, les messages des commit seront en anglais.

3. Mystic tarot

Le but de ce projet est d'implémenter un jeu de *tarot divinatoire*, c'est-à-dire un logiciel permettant de gérer ses cartes de tarot, pour un jeu minimal tel que décrit dans https://fr.wikipedia.org/wiki/Tarot_divinatoire. Cette page énonce les propriétés principales des cartes de tarot, mais libre à vous d'ajouter des variantes et propriétés.

1) Analyse et modélisation

Réfléchissez bien à cette partie avant d'attaquer les autres questions.

Vous devez analyser le domaine de ce problème (les cartes de tarot) et trouver une représentation informatique vous permettant de le simuler. Concrètement, vous pouvez commencer par réaliser un diagramme de classes en notant les différentes classes que vous pensez devoir créer, ainsi que leurs attributs et méthodes. Par exemple, commencez par faire un menu avec les différentes options. Vous implémenterez chaque fonctionnalité une par une par la suite. Dans votre rapport, vous présenterez votre projet de façon à ce qu'une personne tierce puisse comprendre votre architecture et ajouter de nouvelles fonctionnalités facilement. Vous pouvez inclure un diagramme de classe ou tout autre diagramme si vous pensez que cela aide ou clarifie la présentation. Le but de cette partie est que les choix architecturaux de votre projet soient clairs. Il s'agit ainsi de justifier vos choix de structure, et éventuellement démontrer que les autres solutions posent problème.

2) Système de carte basique

Ce logiciel devra nous permettre tout d'abord d'effectuer deux actions de base:

1. Ajouter la description d'une nouvelle carte.
2. Supprimer une carte.

La représentation d'une carte sera importante ici. Vous pouvez notamment réfléchir à l'ensemble des *propriétés* des cartes et leurs différentes *relations*. Justifiez vos choix dans votre analyse que vous décrirez dans le rapport.

3) Extension et recherche

On veut maintenant ajouter quatre nouvelles fonctionnalités

1. Mettre à jour une carte.
2. Consulter sa collection de cartes.
3. Rechercher des cartes sur différents critères, avec deux critères minimum (par exemple le numéro de carte ou le nom de la carte).
4. Ajouter une image correspondante à la carte

Discuter dans votre rapport de la manière dont vous avez étendu votre projet par rapport au système de carte basique.

4) Sauvegarde

Le problème de notre système de paquet de tarot est qu'il ne dispose pas de fonction de sauvegarde. Donc, l'utilisateur doit ré-entrer toutes ses cartes à chaque démarrage du logiciel, ce qui est peu pratique. Nous allons donc permettre a notre utilisateur de *sauvegarder* ses cartes.
Ajoutez une fonctionnalité permettant de sauvegarder les cartes dans un fichier grâce à la *serialization*. Pour cela, utilisez l'interface `Serializable`, voir le tutoriel http://ydisanto.developpez.com/tutoriels/java/serialisation-binaire/.
Bonus. La serialization telle que présentée est *binaire*, donc le fichier créé n'est pas lisible à l'oeil nu et difficilement ré-utilisable par d'autres applications. On veut donc utiliser un format texte comme XML ou JSON. Nous vous conseillons d'utiliser le format JSON (qui est légèrement moins verbeux et plus facile à comprendre), vous pouvez trouver un tutoriel sur l'utilisation d'une librairie JSON à http://www.mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/.

5) Interface graphique

Nous allons maintenant ajouter une interface graphique permettant d'effectuer toutes les opérations proposées par les questions précédentes.

1. Créer une interface graphique générique
2. Permettre l'affichage d'une carte et toutes ses propriétés
3. Ajouter toutes les fonctionnalités précédentes (création, modification)
4. Ajouter une interface de recherche
5. Ajouter un menu de gestion de l'application

En termes de modélisation, il est possible de découpler les fonctionnalités (ajout d'une carte, opérations de recherche) de l'aspect d'interfaçage (console ou `Swing`). Ce découplage est important, car il donne l'intuition nécessaire pour comprendre plus facilement l'interaction entre interfaces et code sous-jacent.
Le but de ce découplage est que les classes dites *métiers*, par exemple la classe représentant la collection de cartes de Tarot, n'utilisent plus du tout les aspects d'interface (elles n'affichent rien, ni ne demandent rien).

On laissera ce travail aux classes de l'*interface* utilisateur. Concrètement, vous avez une classe d'interface graphique qui affiche des aspects et attend de voir ce que l'utilisateur choisit. En fonction des choix de l'utilisateur, cette classe peut

- Afficher un sous-menu (par exemple pour simplifier la recherche).
- Appeler une des classes métiers pour demander un service (ajouter une carte par exemple).

On en déduit que ce sont les classes de l'interface utilisateur qui contiennent celles métiers.
Bonus. Lorsque l'utilisateur fait un choix dans le menu, vous ne devrez plus tester celui-ci explicitement (via `if` ou `switch`) mais implicitement, et que l'action correspondante soit appelée implicitement.
Indice. Utiliser la classe `HashMap` où la clé est l'identifiant du menu (ce que l'utilisateur entre pour naviguer dans un menu) et la valeur est une classe abstraite `Action` dont hérite chacune des actions correspondantes (ajouter, supprimer, etc...).

6) **Bonus** - Lecture du futur

Si vous avez implémenté tous les points précédents, vous pouvez coder un sous-ensemble des règles du jeu de divination. Un exemple du mécanisme complet est disponible sur https://www.tirage.net/. Expliciter dans le rapport quel sous-ensemble des mécanismes du tarot vous implémentez et discutez du design de l'application.