Programmation Orientée Objet
M2 CCI 2008–2009
hoenen@icps.u-strasbg.fr

Université de Strasbourg
Département d'informatique
2 février 2009

Mini-projet : le pendu

Le but de ce mini-projet est de réaliser un jeu simple du *pendu*, en mode ligne de commande et en manipulant chaînes de caractères et tableaux. N'oubliez pas de garder un œil sur l'API java, notamment sur la classe `String` et ses méthodes.

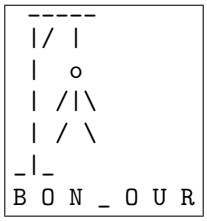

FIG. 1 – Affichage du pendu en mode ligne de commande

Détails pratiques :

- Les différentes classes implantées doivent toutes appartenir à un paquetage `cci.projet.nom` (remplacez `nom` par votre nom).
- Des commentaires doivent être présent pour expliquer les classes et les méthodes implantées, et la génération d'une documentation à l'aide de l'outil `javadoc` serait un plus appréciable.
- Vous pouvez utiliser des méthodes et des classes qui diffèrent par rapport à celles demandées par le sujet (que se soit pour des raisons de praticité d'implantation ou pour sauter une partie que vous ne savez pas traiter), mais les commentaires doivent être alors d'autant plus précis.
- Vous prendrez garde à utiliser les bons modificateurs d'accessibilité selon les situations.
- Votre code source étant placé dans un répertoire qui porte votre nom, vous pourrez en créer une archive compressée en vous plaçant dans ce répertoire `nom` et en tapant la commande :

  ```
  tar -cvzf nom.tgz ../nom
  ```

  Le fichier `nom.tgz` généré devra m'être envoyé par mail pour le vendredi 20 février en indiquant `[pooCCI]` dans le sujet du mail.

1 But du jeu

Le but du jeu est de trouver un mot dont on ne connait initialement que le nombre de lettre, par exemple 7 lettres pour le mot "bonjour". Ce mot est présenté par des traits soulignés pour chaque lettre manquante, soit `_ _ _ _ _ _ _` pour `B O N J O U R`. Le joueur dispose d'un certain nombre de tours de jeu pour trouver le mot en proposant des lettres qui peuvent appartenir au mot. À chaque tour de jeu, le joueur propose une lettre :

- soit la lettre appartient au mot et elle s'affiche aux bonnes positions en complétant le mot à trous (si la lettre se trouve plusieurs fois dans le mot, alors elle s'affiche à chaque occurence),
- soit la lettre n'appartient pas au mot et le dessin du pendu se complète d'une partie, se rapprochant de plus en plus du dessin complet signifiant la perte de la partie.

2 Partie règles du jeu

Vos allez d'abord créer une classe `ReglePendu` pour gérer les informations du jeu et ses mécanismes : l'affichage du mot à trous, la recherche d'un caractère dans le mot, la détection de fin de partie, etc... Cette classe hérite de la classe `DessinPendu` qui est téléchargeable à l'adresse suivante :

http://icps.u-strasbg.fr/~hoenen/cours/2008-2009/poo/index.html

1. Surchargez la méthode `afficher` pour lui permettre de prendre en paramètre un entier représentant le nombre de traits à afficher dans le dessin.
2. Implantez la classe `ReglePendu`, avec comme attributs le mot à trouver, le nombre d'essais autorisés, le nombre d'erreurs commises et un tableau de booléens permettant de savoir quelles lettres ont déjà été trouvées dans le mot. Cette classe doit avoir un constructeur unique qui prend en paramètres le mot à trouver et le nombre d'essais autorisés.
3. Redéfinissez la méthode `afficher()` pour augmenter de manière constante le nombre de traits affichés dans le dessin à chaque faute. Par exemple, si on autorise 2 essais, il faut afficher la moitié des traits à la première faute, et tout le reste à la deuxième faute.
4. Ajoutez une méthode qui affiche le mot à trous (selon l'état d'avancement de la partie).
5. Ajoutez une méthode qui prend un caractère en paramètre et renvoie un booléen indiquant si oui ou non le caractère appartient au mot recherché. Cette méthode doit modifier l'état du tableau représentant le mot à trous.
6. Ajoutez une méthode qui teste l'état du jeu et détermine si oui ou non on a trouvé toutes les lettres (donc si on a gagné).
7. Ajoutez une méthode `boolean jouer(BufferedReader cin) throws IOException` qui prend en paramètre un flux de caractères pour récupérer les caractères à tester. Cette méthode effectue un tour de jeu : elle affiche le mot à trous puis demande au joueur d'entrer un caractère pour le compléter. Ensuite elle teste la présence du caractère dans le mot puis teste la fin du jeu (un message est affiché si la partie est gagnée ou perdue). Enfin elle renvoie un booléen qui vaut `true` si le jeu continu (il reste des lettres à découvrir) et `false` si le jeu s'arrête (gagné ou perdu).

3 Partie interface joueur

1. Faire une classe `JeuPendu` qui contient la méthode `main` dans laquelle une partie est initialisée en prenant le mot à trouver et le nombre d'essais autorisés en arguments de la ligne de commande.
2. Ensuite, l'entrée standard est enveloppée dans un flux de caractères bufferisé et passée en argument de la méthode `jouer`. Cette méthode est appelée tant qu'elle renvoie `true`.
3. La classe `java.io.Console` de l'API Java 1.6 (voir http://java.sun.com/javase/6/docs/api/) permet notamment de rentrer des mots de passe (désactive l'echo et stocke le mot dans un tableau de caractères plutôt que dans un String). Dans le cas où aucun argument n'est passé en ligne de commande, demandez à l'utilisateur de rentrer au clavier le mot à trouver par la méthode `readPassword`.

4 Partie libre

Initialisation On souhaite gérer un troisième type d'initialisation pour le jeu du pendu : le mot doit être choisi au hasard à partir d'un fichier contenant (notamment) une liste de mots. Pour cette partie, vous pouvez implanter ce choix comme bon vous semble, notamment en ce qui concerne l'organisation du fichier (cela peut par exemple être un objet Java sérialisé) et le choix du nombre d'essais autorisés.

Mise à jour du fichier Enfin, on souhaite que le fichier qui permet d'initialiser le jeu avec un mot tiré au hasard soit mis à jour à la fin d'une partie lancée en rentrant un mot en mode standard (ligne de commande ou `Console`).