TP long : Un projet mal implémenté...

I) Contexte

Vous travaillez en tant que développeurs dans une entreprise qui crée des jeux éducatifs. Suite à la demande d'un client, un stagiaire a développé une version basique du jeu du pendu en Java, mais le code ne suit pas les bonnes pratiques et est difficilement maintenable.
La mission de votre équipe est de refactoriser ce programme et d'améliorer sa structure. Vous ajouterez également de nouvelles fonctionnalités pour vous rapprochez des demandes du client. Plus votre équipe est nombreuse, plus le nombre de fonctionnalités attendues par le client est grand.

Prise en main du programme

Il s'agit d'un jeu du pendu : un mot est choisi par l'ordinateur, le nombre de lettres est indiqué au joueur, le joueur doit proposer des lettres et deviner le mot. Lorsque la lettre proposée par le joueur est dans le mot, elle s'affiche à sa place dans le mot, sinon, le joueur perd un essai. Le joueur a le droit à 6 essais, ils correspondent aux traditionnels membres du pendu : la tête, le corps, les bras, les jambes.
La première version du projet est disponible sur Moodle, dans le dossier TP.

- Analysez le code pour en comprendre la structure.
- Identifiez les mauvaises pratiques.

II) Consignes

Voici les consignes que vous devez prendre en compte durant toute l'implémentation, ces critères compteront dans la qualité de votre projet. *Chacun des points évoqués dans les consignes sera pris en compte dans la notation.*

Méthode de travail

Vous devrez travailler dans un répertoire git, le tout premier commit contiendra le code tel qu'il vous a été transmis. Vous devrez maintenir des branches cohérentes. Les messages des commits devront avoir du sens.

Refactorisation

Organisez le code, en créant davantage de classes et de fichiers. La structure pourra évoluer au fur et à mesure des fonctionnalités implémentées (nouvelles classes, etc.). L'utilisation de patrons de conception peut être utile. Ils ne sont obligatoires que lorsqu'ils sont précisés.

Documentations et tests

Chaque classe doit être testée. Le code doit être documenté, par des commentaires dans le code et par un README dans le git qui décrira les fonctionnalités disponibles et comment y avoir accès.

Build

Le projet doit pouvoir être exécuté et testé avec maven.

Rendu

La version du projet qui sera consultée sera celle disponible sur la branche *main* à la fin de la dernière séance, le 6 mai. Vous veillerez à ce qu'un accès (permettant de voir l'intégralité de l'évolution du code sur les différentes branches) à votre répertoire soit accordé à votre correctrice au plus tard à cette date. Un rapide rapport (max 2 pages) pourra être utile pour justifier les choix techniques, il pourra également contenir un diagramme décrivant la structure du projet.

III) Fonctionnalités

Le client a demandé plus de fonctionnalités que celles déjà implémentées par le stagiaire. En voici la liste, par ordre d'importance. *Le nombre de fonctionnalités et leur validité sera pris en compte dans la notation.*

a) Gestion des erreurs

1. Les lettres majuscules et minuscules doivent être considérées de façon équivalente, les accents doivent également être ignorés. Par exemple, si le mot est : "mât", la lettre 'A' entrée par le joueur sera acceptée comme appartenant au mot.
2. Une entrée non valide, c'est-à-dire une séquence de lettres ou un chiffre par exemple, ne doit pas être comptabilisée (aucun essais de perdu) et le joueur doit être prévenu de son erreur.

b) Mots et lettres

1. La liste des lettres déjà proposées doit être affichée.
2. Le mot à deviner : il doit être choisi au hasard dans une liste de mot.
3. Cette liste peut être gérée dans un fichier externe. Le mot peut également être entré par un ami du joueur avant le début du jeu. Implémenter ces différentes façon d'obtenir un mot et utilisez une factory method pour gérer ces différentes sources.

c) Score

1. Le score doit être affiché en fin de partie. Il sera calculé en fonction du temps pris pour trouver la solution et du nombre d'essais utilisés. Plus exactement, il correspond à la formule (nombre d'essais restants * 100) - (temps en secondes). Par exemple, si je trouve le mot en 1 minute et qu'il me reste 2 essais, j'ai un score de 2*100 - 60 = 140.
2. Si le score est assez haut, il doit être enregistré à la fin de la partie dans un fichier, avec le nom du joueur. À la fin de chaque partie, les 3 meilleurs scores sont affichés.

d) Affichage

1. Le client regrette de ne pas voir le pendu se dessiner au fur et à mesure des essais. Ajoutez un affichage pour le contenter.
2. Proposez une version avec interface graphique.
3. Laissez le choix au joueur d'utiliser la version graphique ou terminale.