Labo 4 – Choix des Collections

Objectifs (2h)

Au terme de ce laboratoire, vous serez capable de :

→ Justifier le choix d'une interface de Collection du Java Collections Framework (List, Queue, Set et Map) sur base de leurs principales fonctionnalités ;

→ Justifier le choix de l'implémentation concrète d'une Collection du Java Collections Framework (List, Queue, Set et Map) sur base des complexités des principales méthodes utilisées par un programme ;

→ Améliorer un programme en modifiant les choix d'interfaces et d'implémentation qui ont été réalisés.

Préambule

Ce laboratoire te propose de déposer ton travail à la fin des deux heures. Les responsables de laboratoire le corrigeront et tu recevras un feedback. C'est un excellent entrainement pour l'activité intégrative : lors de celle-ci, tu seras évalué sur tes choix de collections !

Description du problème (15 min explications + 15 min en individuel)

L'archive `algo.labo04` contient un programme permettant de jouer une ou plusieurs parties de « Pendu ».

Au démarrage du programme, l'utilisateur sélectionne son choix dans un menu :

1. Démarrer une partie ;
2. (Dés)Activer le mode « `debug` » ;
3. Quitter.

Mode « `debug` »

Le mode « `debug` » vous permet simplement d'afficher la réponse au début de la partie de « Pendu ».

Démarrer une partie

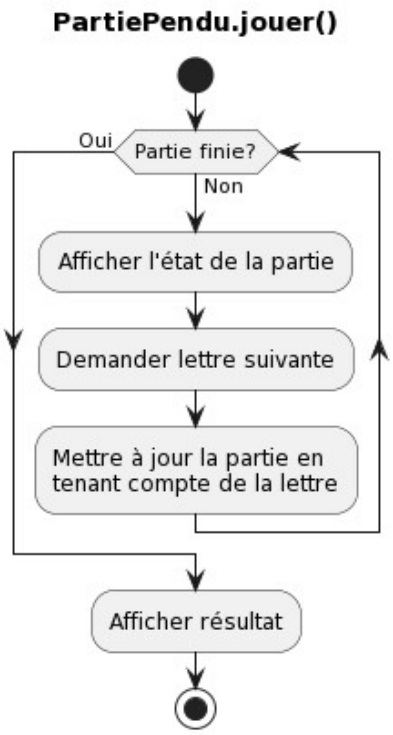
Le démarrage d'une nouvelle partie consiste à effectuer les étapes suivantes :

1. Choisir un mot au hasard parmi ceux qui ont été lu dans un fichier. La lecture dans le fichier est possible via la fonction :
   ```
   LecteurFichier.depuis("data/mots.txt").dans(collectionMots);
   ```

Le chemin du fichier est déjà encodé pour vous. La référence `collectionMots` est une `Collection<String>` de votre choix. On extrait ensuite un mot au hasard depuis cette collection.

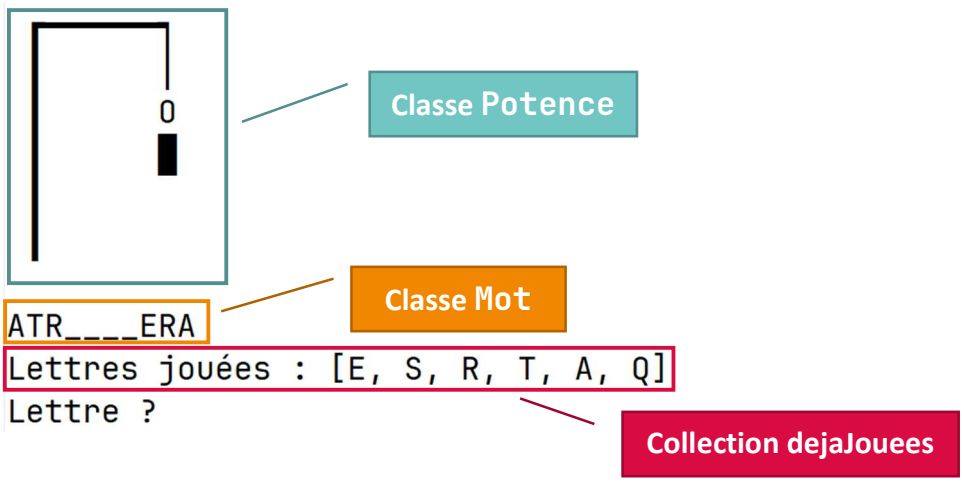
2. On crée une instance de la `PartiePendu` grâce à la méthode de fabrique statique « `nouvellePartie(String reponse)` » ;
3. On lance la partie proprement dite à l'aide de l'appel de la méthode jouer().

Comme vous pouvez le voir dans la méthode `PartiePendu.jouer()`, une partie de pendu se déroule comme suit :



À chaque itération de la boucle de jeu, voici ce qu'il se passe :

1. On affiche l'état de la partie, comme ci-dessous :



   a. La potence est dessinée pour représenter le pendu. Une classe « `Potence` » est responsable de savoir retourner les représentations des différents stades du jeu.
   b. Le mot en train d'être deviné est affiché. On utilise des « _ » pour symboliser les lettres non devinées. La classe « `Mot` » est responsable de la manipulation correcte des lettres et de la représentation de ce mot.
   c. Les lettres déjà jouées sont aussi mentionnées.
2. On demande une nouvelle lettre à l'utilisateur
3. Sur base de cette lettre, plusieurs choses peuvent se passer :
   a. Si la lettre n'a pas encore été proposée et est dans le mot à deviner, le programme l'affiche à chaque position correcte dans le mot ;
   b. Si la lettre a déjà été proposée, le programme affiche un message et ne fait rien d'autre ;
   c. Si la lettre n'a pas été proposée et n'appartient pas au mot, le programme incrémente le malus du joueur et affiche la potence correspondante.
4. Comme vous pouvez vous en douter, la partie se termine si le mot mystère est complètement révélé ou si la Potence atteint son dernier stade (le personnage pendu).

Évaluation des choix de collection et du code fournit (30 min)

Dans l'archive disponible sur HELMo LEARN, tu trouveras trois grilles qui justifient le choix de collections :

1. La collection présente dans la classe « Mot ». Cette grille est **correcte** et t'est fournie en exemple.
2. La collection qui contient les mots dans laquelle on sélectionne le mot à faire deviner (voir la classe « Pendu »).
3. La collection qui permet d'identifier les lettres déjà jouées (voir la classe « PartiePendu »).

Ta **première tâche** consiste à prendre connaissance de ces grilles et à évaluer les grilles « Pendu – ChoixCollection – A évaluer » et « PartiePendu – ChoixCollection – A évaluer » à l'aide d'un questionnaire disponible sur Helmo LEARN.

Amélioration des choix de collection et du code (30 min)

Lors de la phase précédente, tu as évalué les deux grilles. Tu as peut-être remarqué des erreurs ou des imperfections dans l'une ou l'autre grille ? Les choix ou leurs justifications étaient-ils tous corrects ?

Dès lors, ta **deuxième tâche** consiste à améliorer ces grilles. Pour ce faire, édite directement les fichiers Word des grilles.

Si tu as modifié les grilles, c'est peut-être que tu as considéré que certains choix d'interfaces ou de collections n'étaient pas correct ou optimaux !

Par conséquent, ta **troisième tâche** consiste à corriger le code du programme, conformément aux grilles que tu as améliorées.

Dépôt du code et des justifications (5 min)

Crée une archive contenant tes grilles et ton projet java. Dépose-là sur HELMo LEARN. Les enseignants de laboratoire corrigeront les dépôts et tu recevras un feedback. Ce feedback sera d'ailleurs similaire à celui que tu recevras lors de l'Activité Intégrative.

Questionnaires

Après ton dépôt, sur HELMo LEARN, sur l'espace dans lequel tu as rempli l'enquête préliminaire, tu dois encore répondre à une ou deux enquêtes (si tu ne les trouves pas, demande à ton responsable de laboratoire).
Nous t'invitons à répondre à ces enquêtes le plus complètement possible.