JAVA SE

PROJET « Jeux d'enfants »

Table des matières

1 Présentation

Vous devez réaliser un programme « multi-jeux » à destination des jeunes enfants. Ce programme propose deux niveau (facile et difficile) et trois activités :

- Une « ardoise magique » pour dessiner à volonté.
- Des exercices de calcul.
- Un pendu.

2 Contraintes techniques

Le programme est entièrement graphique. Il se tiendra dans une fenêtre unique qui proposera quatre activités :

- La première activité (*Dessin*), affichera l'ardoise magique avec une barre d'outils proposant un bouton pour tout effacer, une gomme et une sélection de couleurs :
  - Les couleurs rouge, vert et bleu au niveau *facile*.
  - Une palette de couleur complète au niveau *difficile*.
- La seconde activité (*Calcul*), proposera un calcul aléatoire à résoudre, un champ de saisie pour la réponse (champ ayant automatiquement le focus lors de l'affichage du calcul proposé) et des boutons « *Vérifier* », « *Solution* » et « *Nouveau* ».
  - Les calculs de niveau *facile* seront des additions ou des soustractions de nombres à 1 chiffre, le résultat devant être positif mais pouvant être à 2 chiffres.
  - Les calculs de niveau difficile seront des additions ou des soustractions de nombres pouvant aller jusqu'à 3 chiffres avec un résultat qui peut être négatif ou à 4 chiffres, et des multiplications de nombres à 1 chiffre.
- La troisième activité (*Pendu*), proposera de trouver un mot tiré aléatoirement parmi ceux présents dans un dictionnaire, un clavier virtuel proposant les 26 lettres de l'alphabet sous forme de boutons et un espace avec un dessin représentant « l'avancée » du pendu : chaque erreur fera avancer vers la pendaison totale. Ce jeu n'utilise pas de niveau.
- La quatrième activité (*Administration*), sera invisible par défaut. Elle ne sera visible que si l'on donne le mot de passe administrateur du logiciel. Cette activité doit permettre de modifier ou d'ajouter un mot au dictionnaire. Elle affichera donc :
  - La liste de tous les mots, sélectionnables pour modification.
  - Un champ d'édition du mot, prérempli dans le cas d'une modification ou vide dans le cas d'un nouveau mot.
  - Un bouton « *Enregistrer* » qui enregistre le nouveau mot ou la modification, bouton qui ne doit être cliquable que si le champ contenant le mot est rempli.
- Le mot de passe de l'administrateur ne doit pas pouvoir être lu directement dans le code source, ni même au niveau de la machine : il s'agira donc de le stocker **haché** dans un *fichier de configuration*.

La fenêtre principale proposera également un *menu* contenant :

- Une entrée *Activité* proposant les trois activités. Le clic sur l'une d'elle sélectionnera l'activité correspondante.
- Une entrée *Niveau* proposant les 2 niveaux de difficulté. Le choix d'un niveau provoque la réinitialisation des activités utilisant le niveau (dessin et calcul).

- Une entrée *Administration* proposant la modification du dictionnaire. L'accès à ce menu sera soumis à l'entrée d'un mot de passe. L'accès, une fois autorisé, affiche la quatrième activité dans la fenêtre.

Les exercices de calculs seront créés à la demande et dynamiquement par le programme lui-même suivant les conditions énoncées plus haut.

Les mots du dictionnaire seront stockés dans une base de données textuelle éditable via une interface graphique (la quatrième activité). La base de données se réduit donc à un fichier texte qu'il faudra gérer tant en lecture qu'en écriture.

3 Livrables attendus

Les codes sources bien sûr, correctement rangés dans des packages adéquats, et correctement présentés (nom du codeur en commentaire, indentation correcte et commentaires au minimum, *javadoc* bienvenue).

Le plan de développement. Je ne vous demande pas de me rendre un devoir de gestion de projet, mais au moins le résultat de vos réflexions initiales présentant les choix faits, l'explication de ce choix, et les solutions techniques envisagées. Si les solutions techniques changent au cours du projet (ça arrive fréquemment), il faut expliquer pourquoi et les avantages de la nouvelle solution par rapport à l'ancienne… et prouver que ça n'affecte pas le reste du projet.

Bref, je veux voir les documents inhérents à la gestion de projet. Si en plus vous savez faire un Gantt et un prévisionnel de temps, ça n'en est que mieux, mais encore une fois ce n'est pas le but.

4 Pour aller plus loin

Si le projet vous semble trop court, vous pouvez améliorer les choses. Voici quelques idées :

- Programmer des touches de raccourci pour les différents menus et items de menu.
- Pour le dessin, rajouter le choix de la taille du trait.
- La possibilité de changer le mot de passe de l'administrateur.
- La possibilité de sauvegarder, charger et/ou redessiner un dessin.

Ce ne sont que quelques idées, les vôtres seront les bienvenues. Aucune obligation de réaliser ces idées, ce n'est que si vous désirez aller plus loin ET si vous en avez le temps !

5 Memento - Aide technique

Organisation

- Un projet nécessite un *chef de projet*. Choisissez de façon collégiale celui qui va gérer le projet. Le rôle d'un chef de projet n'est pas d'être un dictateur mais un facilitateur. Il est celui qui s'assure que toute l'équipe travaille dans le même sens, avec les mêmes objectifs. Il est celui qui résout les conflits pouvant surgir dans l'équipe. Il est aussi celui qui aura le dernier mot en cas de choix difficile. Il est également celui qui reçoit les lauriers de la victoire… et qui doit assumer l'échec ! Ce n'est pas un rôle facile, mais ce sera le vôtre dans un futur proche, autant vous y accoutumer dès maintenant.
- Un projet est une succession de problèmes à résoudre, de TP à réaliser. La première chose à faire est donc de réaliser un plan de développement indiquant des solutions techniques. Chaque membre de l'équipe sera chargé de produire une partie du projet tirée de ce plan de développement. Il est donc nécessaire que chacun apporte sa pierre à l'édifice, tant au niveau des idées de développement que des solutions techniques.

- Prenez un temps pour transmettre votre savoir et vos découvertes aux autres membres du projet. Le partage est la meilleure façon d'apprendre.
- Si vous souhaitez utiliser un système de fenêtre à onglets, allez voir la documentation de la classe JTabbedPane.

Partage de code

Afin de simplifier les échanges de données et les progressions de chacun, il est fréquent d'utiliser un logiciel de versionning. Le plus utilisé à ce jour est Git, nous avons fait un TP dessus. Vous devrez donc utiliser un dépôt Git partagé. Pour éviter les erreurs fréquentes dues à l'utilisation de Git, suivez ce plan :

- Le chef de projet :
  - Crée le dépôt de référence sur github/gitlab/bitbucket/autre.
  - **Clone** le dépôt sur sa machine.
  - Crée le projet sur sa machine (dans le répertoire cloné, évidemment !).
  - Créer le fichier « *.gitignore* » (je vous le mets en ressource du projet) à la racine du projet pour éviter tout problème si vous utilisez des IDE différents.
  - Fait le premier **commit** (pour rappel, les répertoires vides ne sont pas commit).
  - **Push** le tout.
  - Donne les droits d'administration aux autres membres de l'équipe en les invitant.
- Les autres membres de l'équipe vont alors :
  - **Cloner** le dépôt sur leur propre machine.
  - **Créer leur branche** de développement.
  - Développer sur leur branche.
  - **Commit** multiples sur leur branche.
  - Quand le développement de la fonctionnalité est terminé **checkout master (**ou **main** selon le nom de la branche principale créée**)**, **pull** la branche depuis le dépôt de référence, **merge** leur branche, et **push** sur le dépôt de référence.
  - Avertir les autres qu'il y a eu **push** pour que les collègues fassent un **pull** s'ils en ont besoin.
  - **Créer une nouvelle branche** pour le prochain développement.

Si vous suivez ces conseils, il n'y aura (presque) aucun problème.

MAIS cela nécessite que tout le monde s'accorde sur le nommage des fichiers. Si chacun choisit de nommer ses packages ou ses fichiers n'importe comment, il ne peut en résulter d'un chaos total et un KO définitif lorsqu'il faudra récupérer le code des autres ! Ça fait partie de la gestion du projet : s'assurer que tout le monde respecte la nomenclature (pour rappel, Java est sensible à la casse mais Windows ne l'est pas, ce qui peut compliquer les choses si vous avez la sale manie de mettre des majuscules là où il n'en faut pas et vice-versa…).

6 Contraintes de la présentation

- Prenez le temps de préparer votre présentation. Un bon *PowerPoint* est toujours appréciable.
- Le chef de projet fera la présentation globale du projet.
- Chaque membre du groupe devra présenter sa partie. Donc tout le monde va parler…
- La présentation durera au plus 40 minutes tout compris. Je compte :
  - 5 minutes maximum d'installation et de préparation.
  - 15 minutes de présentation orale.

    - 15 à 20 minutes d'échanges avec moi d'abord dans le rôle du patron, puis un jeu de questions-réponses avec les autres spectateurs (les autres élèves) ensuite, ceci afin d'éclaircir des points techniques ou des choix effectués.
- Comme il est hors de question de vous laisser aller voir le client directement, la présentation sera faite comme si votre interlocuteur était votre patron. Vous devrez donc défendre votre projet tant au niveau de l'ergonomie que de la programmation (choix de développement, méthode de travail utilisée, concepts mis en œuvre, exemples de code…). Ne vous enfoncez pas dans le code avec une analyse ligne à ligne, ça ne présente aucun intérêt à part indiquer que vous ne savez pas quoi dire sur votre propre travail… Sans compter que votre patron n'est pas nécessairement un as du code. La seconde partie de la présentation, le jeu de questions-réponse, permettra de s'enfoncer dans la technique.