Travail pratique #3

Ce travail peut être fait individuellement ou en équipe de deux.

AUCUN RETARD PERMIS

Notions mises en pratique : Respect d'un ensemble de spécifications, et implémentation d'une interface graphique.

1. Description et exigences logicielles

Ce travail consiste à concevoir une petite application graphique permettant de créer de petits tests à choix multiples sur des sujets divers, et de passer les tests créés. Un test possède un nom (indiquant son sujet) ainsi qu'une liste de questions. Chaque question est composée d'un énoncé, de quatre choix de réponses, et de la bonne réponse parmi les 4 choix (il ne doit y avoir qu'une seule bonne réponse parmi les choix de réponses). Votre programme doit enregistrer les tests créés avec l'application, dans le fichier **tests.txt**, dans le format de fichier texte expliqué ci-dessous. Le fichier tests.txt doit être **enregistré à la racine de votre projet**.

Types de séparateurs utilisés pour séparer les différentes sections dans le fichier tests.txt :

SÉPARATEUR_TESTS = "====="
SÉPARATEUR_QUESTIONS = "-----"
SÉPARATEUR_CHOIX_REPONSES = "<>"

Chaque test, dans le fichier tests.txt, est séparé par le SÉPARATEUR_TEST. Pour chaque test, la première ligne indique le nom du test, la seconde ligne indique le nombre de questions du test, et la troisième ligne est un SÉPARATEUR_QUESTION. Par exemple :

```
Les couleurs        // le nom du test
15                  // le nombre de questions dans ce test
-----               // SÉPARATEUR_QUESTION
```

Ensuite, les questions du test sont inscrites, une à la suite de l'autre, dans l'ordre (question 1, question 2, question 3, etc.), dans le format suivant :

- L'énoncé de la question *=> peut faire plusieurs lignes*
- Un SÉPARATEUR_QUESTIONS *=> seul sur la ligne, au début de la ligne*
- Les 4 choix de réponses inscrits sur une seule ligne, séparés par 3 SÉPARATEUR_CHOIX_RÉPONSES
- Un SÉPARATEUR_QUESTIONS *=> seul sur la ligne, au début de la ligne*
- Le numéro de la bonne réponse parmi les choix de réponses donnés *=> entre 0 et 3 inclusivement, 0 correspondant au premier choix, 1 au deuxième choix, etc.*
- Un SÉPARATEUR_QUESTIONS (ou un SÉPARATEUR_TESTS si c'est la dernière question du test) *=> seul sur la ligne, au début de la ligne.*

Par exemple :

```
Quelle couleur donne le mélange de rouge et bleu ?
-----
Orange<>Bleu<>Violet<>Turquoise
-----
2       // indique que la bonne réponse à cette question est Violet
-----   // ou  =====  si c'est la dernière question du test

// etc…
```

Le fichier tests.txt ne contient aucune ligne blanche, sauf, possiblement, dans l'énoncé des questions ou à la toute fin du fichier. Les tests sont enregistrés l'un à la suite de l'autre dans le fichier.

Mélanie Lord
lord.melanie@uqam.ca

Un fichier exemple, tests.txt, vous est fourni avec l'énoncé de ce TP. Celui-ci contient 3 tests. ASSUREZ-VOUS que votre programme puisse lire (et créer) le fichier tests.txt EXACTEMENT dans ce format, car ce sera ce format de fichier qui sera utilisé dans les tests de votre application. DE PLUS, ASSUREZ-VOUS que le fichier est bien enregistré à la racine du projet, lorsqu'il existe. Notez que si le fichier n'existe pas, on considère qu'il n'y a aucun test de créé.

Votre application (la méthode `main`) DOIT SE TROUVER dans une classe nommée `GenerateurTests`. C'est cette classe qui sera exécutée lors des tests de votre application. Cependant, vous pouvez faire d'autres classes pour modulariser votre code (comme une classe pour la fenêtre #2, une autre pour la fenêtre #3, une classe pour conserver les données d'un test, etc.). **Vous pouvez remettre un maximum de 5 classes différentes** (en comptant `GenerateurTests`).

1.1 ASPECT VISUEL DE L'INTERFACE GRAPHIQUE

1.1.1 Fenêtres de l'application

L'interface graphique est composée de 3 fenêtres différentes (3 `JFrame`) : fenêtre #1, fenêtre #2, et fenêtre #3.

Fenêtre #1	- Fenêtre principale de l'application. - Au démarrage de l'application, c'est cette seule fenêtre qui apparaît. - Dimensions : largeur = **380**, hauteur = **360** - Lorsqu'on clique le bouton de fermeture de la fenêtre [x], la fenêtre disparaît, et l'application se termine.
Fenêtre #2	- Fenêtre permettant de créer un nouveau test. - N'est pas visible au démarrage de l'application - Dimensions : largeur = **550**, hauteur = **540** - Lorsqu'on clique le bouton de fermeture de la fenêtre [x], la fenêtre disparaît, mais l'application ne se termine pas.
Fenêtre #3	- Fenêtre permettant de passer un test existant. - N'est pas visible au démarrage de l'application - Dimensions : largeur = **550**, hauteur = **540** - Lorsqu'on clique le bouton de fermeture de la fenêtre [x], la fenêtre disparaît, mais l'application ne se termine pas. Cette fenêtre possède deux vues différentes (A et B). La vue A est la vue qui permet de naviguer dans les questions du test, et d'y répondre tandis que la vue B est la vue qui affiche le résultat du test, après une demande de correction.

De plus :

- Chaque fenêtre est non redimensionnable.
- Chaque fenêtre doit s'ouvrir au centre de l'écran (peu importe la résolution de l'écran).

Les 3 figures suivantes montrent les 3 fenêtres.

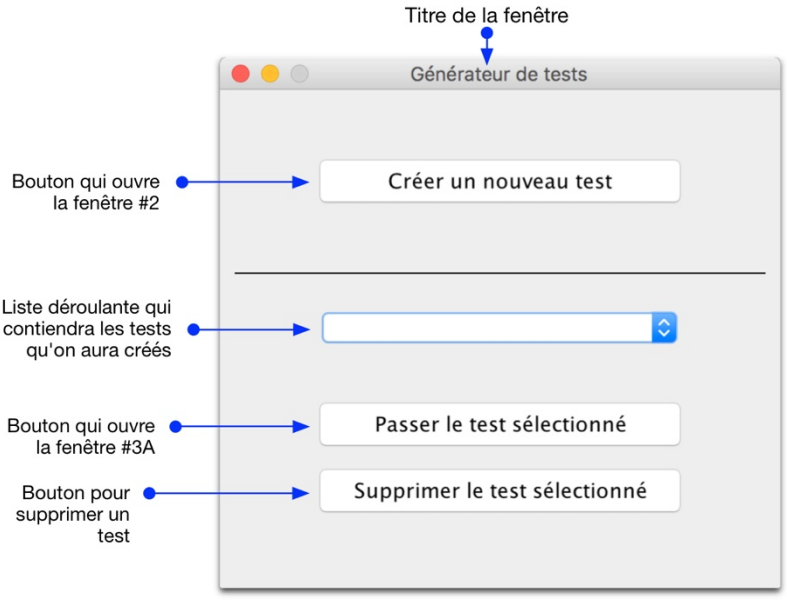


Figure 1. Illustration de la fenêtre #1

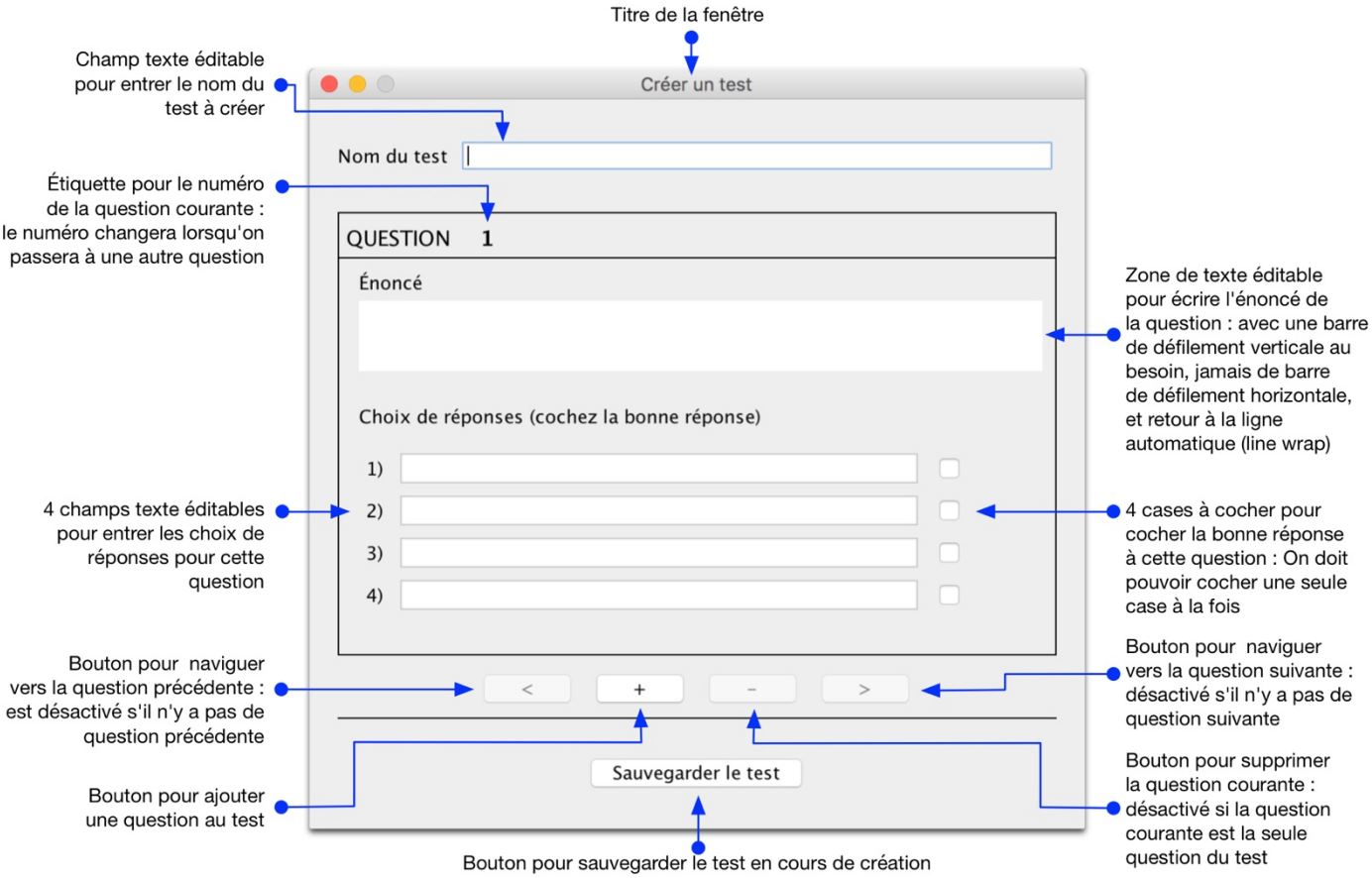


Figure 2. Illustration de la fenêtre #2

Mélanie Lord
lord.melanie@uqam.ca

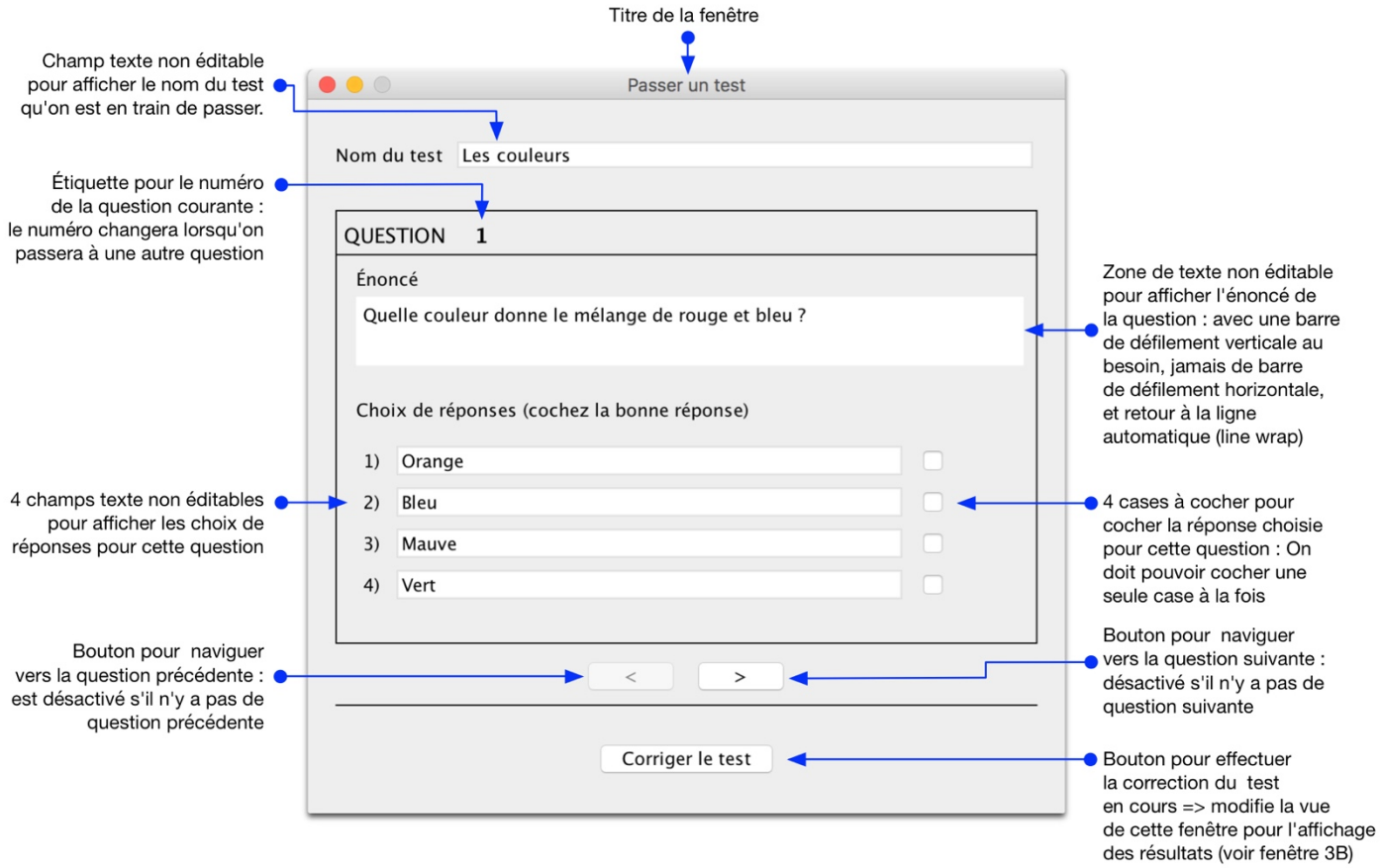


Figure 3 : Illustration de la fenêtre #3 (vue A)

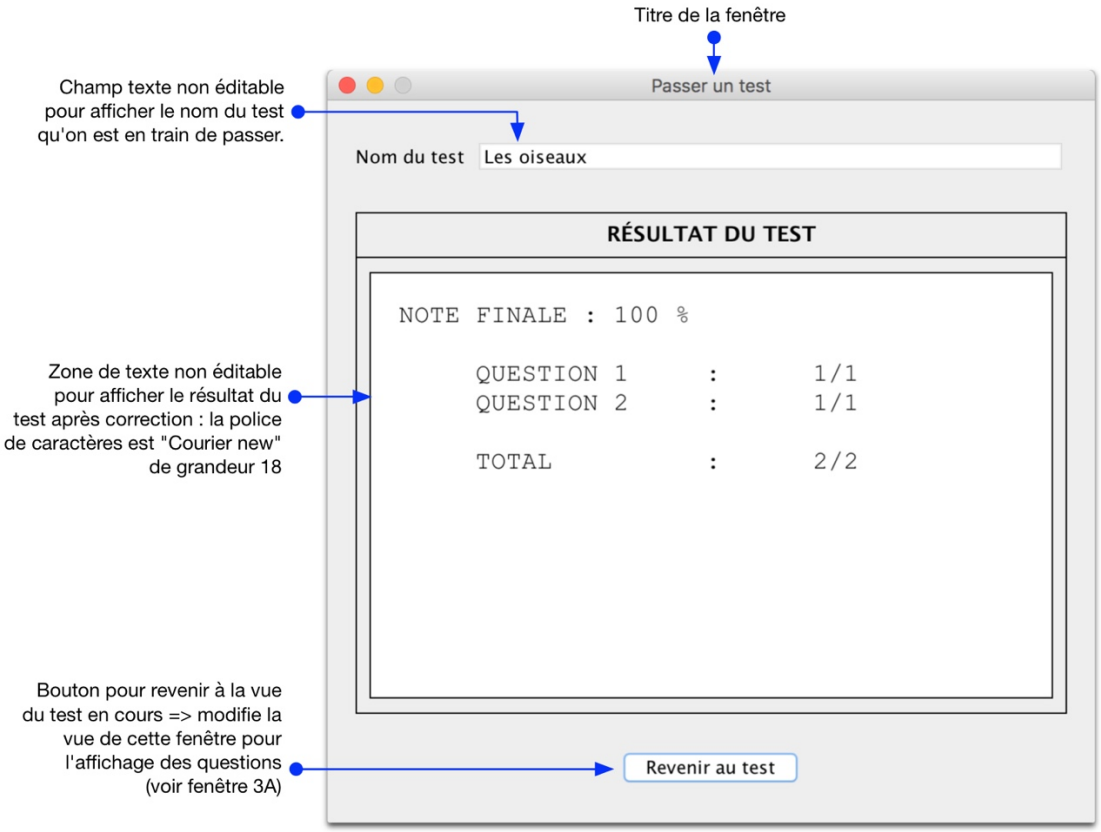


Figure 4 : Illustration de la fenêtre #3 (vue B)

1.2 FONCTIONNALITÉS DE L'INTERFACE

1.2.1 Fenêtre #1

Composant	Fonction
Bouton [Créer un nouveau test]	Lorsqu'on clique sur ce bouton, l'application ouvre la fenêtre #2, centrée dans l'écran, par-dessus la fenêtre #1. La fenêtre #2 affiche la question numéro 1 pour ce nouveau test, tous ses champs sont vides, et ses cases à cocher sont toutes décochées.
Bouton [Passer le test sélectionné]	Lorsqu'on clique sur ce bouton : 1) Si la liste des tests est vide, l'application affiche un message d'erreur dans une fenêtre surgissante (`JoptionPane`), centrée au milieu de la fenêtre #1. 2) Si la liste des tests n'est pas vide, l'application ouvre la fenêtre #3 (vue A), qui affiche alors le nom, et la première question du test sélectionné dans la liste.
Bouton [Supprimer le test sélectionné]	Lorsqu'on clique sur ce bouton : 1) Si la liste des tests est vide, l'application affiche un message d'erreur dans une fenêtre surgissante (`JoptionPane`), centrée au milieu de la fenêtre #1. 2) Si la liste des tests n'est pas vide, l'application supprime le test sélectionné de la liste déroulante (et le supprime aussi du fichier tests.txt).

Voir l'exemple d'exécution de la fenêtre #1 comme spécifications complémentaires, et pour voir les messages à afficher.

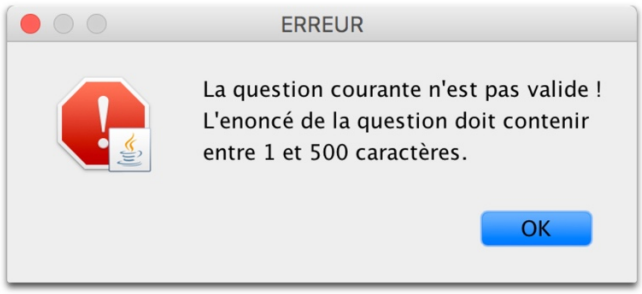
1.2.2 Fenêtre #2

La fenêtre #2 sert à créer un nouveau test.

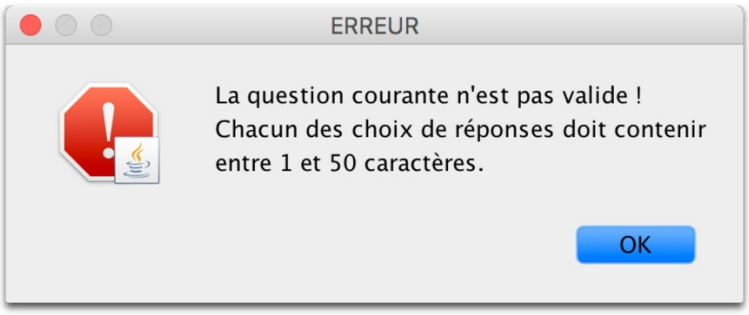
Composant	**Fonction**
Bouton [<] :	Ce bouton sert à revenir à la question qui précède la question courante. **Il doit être désactivé lorsque la question courante est la première question de ce test**. Lorsqu'on clique sur ce bouton (s'il est actif) : 1) Si les champs de la question courante sont valides, on affiche la question précédente. 2) Si les champs de la question courante sont invalides, on affiche un message d'erreur dans une fenêtre surgissante, et la question courante reste inchangée.
Bouton [>]	Ce bouton sert à aller à la question qui suit la question courante. **Il doit être désactivé lorsque la question courante est la dernière question de ce test**. Lorsqu'on clique sur ce bouton (s'il est actif) : 1) Si les champs de la question courante sont valides, on affiche la question suivante. 2) Si les champs de la question courante sont invalides, on affiche un message d'erreur dans une fenêtre surgissante, et la question courante reste inchangée.
Bouton [+]	Ce bouton sert à ajouter une question au test courant. Lorsqu'on clique sur ce bouton : 1) Si les champs de la question courante sont valides, on affiche une nouvelle question (dont tous les champs sont vides et les cases décochées) tout de suite après la question courante (le numéro de toutes les questions suivantes augmente de 1, sauf si la question courante était la dernière, et que dans ce cas, on ajoute à la fin). 2) Si les champs de la question courante sont invalides, on affiche un message d'erreur dans une fenêtre surgissante, la question courante reste inchangée, et l'ajout est annulé.
Bouton [-]	Ce bouton sert à supprimer une question. **Il doit être désactivé lorsqu'il n'y a aucune ou seulement une question dans le test courant.** Lorsqu'on clique sur ce bouton (s'il est actif), la question courante est supprimée, et l'on affiche la question qui venait juste après la question supprimée. Aussi, le numéro des questions qui venaient après la question supprimée diminue de 1.
Bouton [Sauvegarder le test]	Ce bouton sert à sauvegarder le test courant. Lorsqu'on clique sur ce bouton : 1) Si le nom du test n'est pas valide, ou si la question courante n'est pas valide, le programme affiche un message d'erreur dans une fenêtre surgissante,selon le cas, l'enregistrement est annulé, et la fenêtre demeure inchangée. Notez que le programme n'affiche qu'un seul message d'erreur, dans cet ordre : 1) nom du test invalide, et 2) champ(s) de la question courante invalide(s) (voir précision #5 pour les champs de la question invalides). 2) Si le nom du test, et la question courante sont valides : – Le test est sauvegardé dans le fichier tests.txt ainsi que dans la liste déroulante de la fenêtre #1. – La fenêtre #2 se ferme. – Un message de confirmation de l'enregistrement est affiché.

Précisions

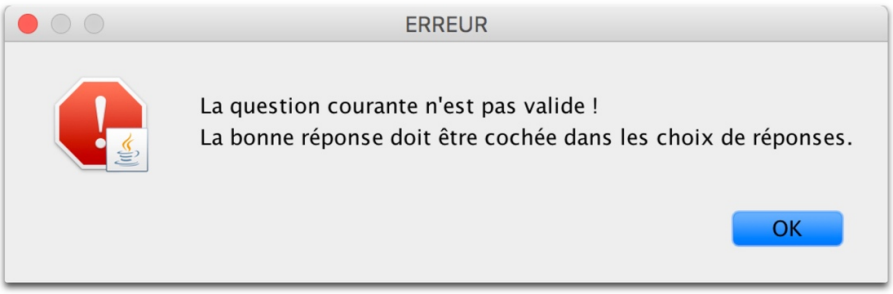
1. Un nom de test valide doit contenir entre 1 et 50 caractères (qui ne sont pas que des espaces).
2. Les champs de la question courante sont considérés comme valides si :
   - Le champ "Énoncé" de la question contient entre 1 et 500 caractères (qui ne sont pas que des espaces).
   - Chacun des 4 choix de réponses contient entre 1 et 50 caractères (qui ne sont pas que des espaces).
   - La bonne réponse, parmi les choix de réponses, est cochée.
3. Notez que pour chacun des 5 boutons, lorsque les champs de la question courante ne sont pas valides, le programme affiche le message approprié et l'opération est annulée.
4. En tout temps, on peut modifier les champs de la question courante, et ces modifications doivent être conservées (si valides).
5. Si plusieurs champs de la question courante sont invalides, le programme ne doit afficher qu'un seul **message d'erreur** : celui de la première erreur rencontrée, dans cet ordre :
   1) Énoncé invalide



   2) Choix de réponses invalides



   3) Bonne réponse non cochée



Voir les exemples d'exécution de la fenêtre #2 comme spécifications complémentaires, et pour voir tous les messages à afficher.

1.2.3 Fenêtre #3 (vue A)

La fenêtre #3 (vue A) sert à passer un test existant.

Composant	Fonction
Bouton [<] :	Ce bouton sert à revenir à la question qui précède la question courante. **Il doit être désactivé lorsque la question courante est la première question de ce test**. Lorsqu'on clique sur ce bouton (s'il est actif) : 1) Si la réponse est cochée, on affiche la question précédente. 2) Si la réponse n'est pas cochée, on affiche un message d'erreur à cet effet, dans une fenêtre surgissante, et la question courante reste inchangée.
Bouton [>]	Ce bouton sert à aller à la question qui suit la question courante. **Il doit être désactivé lorsque la question courante est la dernière question de ce test**. Lorsqu'on clique sur ce bouton (s'il est actif) : 1) Si la réponse est cochée, on affiche la question suivante. 2) Si la réponse n'est pas cochée, on affiche un message d'erreur à cet effet, dans une fenêtre surgissante, et la question courante reste inchangée.
Bouton [Corriger le test]	Ce bouton sert à faire la correction du test en cours. Lorsqu'on clique sur ce bouton : 1) Si la réponse de la question courante n'est pas cochée, on affiche un message d'erreur à cet effet, dans une fenêtre surgissante. Si elle est cochée, on vérifie alors si l'on a répondu à toutes les questions du test, et si ce n'est pas le cas, on affiche un message d'erreur, dans une fenêtre surgissante, qui indique le plus petit numéro de question parmi les questions qui n'ont pas de réponse. Lorsqu'un message d'erreur s'affiche, la fenêtre reste inchangée. 2) Si l'on a répondu à toutes les questions du test, le programme calcule le résultat au test, et l'affiche dans la vue B.

Précisions :

1. Si l'on passe le test une autre fois, celui-ci est complètement réinitialisé : les résultats des tests passés ne sont pas conservés.

1.2.4 Fenêtre #3 (vue B)

La fenêtre #3 (vue B) sert à afficher la correction du test.

Composant	Fonction
Bouton [Revenir au test]	Ce bouton sert à revenir à la vue A qui affiche les questions du test. L'utilisateur peut ainsi en profiter pour tenter de corriger les questions qu'il a manquées.

Précisions pour le calcul des points :

1. Chaque question vaut 1 point.
2. On calcule le total de bonnes réponses sur le total de questions dans le test, puis on reporte la note sur 100. Le programme **arrondit la note sur 100 à l'entier le plus près** avec la méthode `Math.round(…)`.

Voir l'exemple d'exécution de la fenêtre #3 comme spécifications complémentaires, et pour voir les messages à afficher.

1.2.5 Notes diverses sur l'implémentation

Au démarrage de l'application, votre programme doit lire tous les tests se trouvant dans le fichier tests.txt, s'il existe. Ceux-ci doivent alors apparaître dans la liste déroulante de la fenêtre #1. Si le fichier n'existe pas, ou s'il est vide, la liste déroulante demeure vide.

Assurez-vous que lorsque l'application se termine, les tests qui sont dans le fichier tests.txt sont exactement ceux qui sont indiqués dans la liste déroulante de la fenêtre #1 au moment de la fermeture. Et lors du redémarrage de l'application, ce sont ces mêmes tests qui apparaissent de nouveau dans la liste déroulante de la fenêtre #1. Il faut donc mettre à jour le fichier à chaque suppression et ajout de tests.

Pour centrer une fenêtre (`JFrame`) au milieu de l'écran : `fenetre.setLocationRelativeTo(null);`

Pour fermer une fenêtre (`JFrame`) par programmation : `fenetre.dispose();`

2. Documents fournis

1) **tests.txt** : exemple de fichier que votre programme doit pouvoir lire et créer. Il contient 3 tests.
2) **exemplesExec.zip** : dossier zippé qui contient des fichiers vidéos qui montrent l'utilisation de l'application. Ces exemples d'exécution doivent être considérés comme des spécifications à respecter, au même titre que ce qui est écrit dans ce document.

3. Précisions et contraintes d'implémentation

1) La classe obligatoire à remettre, et qui doit contenir la méthode `main` de l'application, doit se nommer `GenerateurTests`. Vous pouvez cependant remettre 4 classes supplémentaires si vous le désirez.
2) Toutes les fenêtres surgissantes doivent apparaître centrées au milieu de la fenêtre de l'application.
3) Les exemples d'exécution fournis avec l'énoncé du TP doivent être considérés comme des spécifications complémentaires à respecter.
4) Votre/vos classe/s à remettre doit/doivent se trouver dans le **paquetage par défaut**.
5) Il ne doit y avoir aucun affichage à la console.

Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.

4. Détails sur la correction

Votre travail sera noté sur le respect des spécifications concernant 1) l'aspect visuel de l'interface graphique, et 2) les fonctionnalités de l'application.

5. Date et modalités de remise

5.1 REMISE

Date de remise : **25 avril 2018** avant minuit (**AUCUN RETARD PERMIS**).

Fichier obligatoire à remettre : `GenerateurTests.java`. NE **PAS** REMETTRE VOS FICHIERS DANS UNE ARCHIVE (zip, rar, …). Les remettre séparément.

Remise via Moodle uniquement.

Vous devez remettre (téléverser) votre/vos fichier/s sur le site du cours (Moodle). Vous trouverez la boîte de remise dans la section **BOÎTES DE REMISE - TRAVAUX PRATIQUES / REMISE DU TP3**.

POUR LA REMISE D'UN TRAVAIL FAIT EN ÉQUIPE :

AVANT la date de remise du TP3, assurez-vous que votre équipe soit bien inscrite dans la liste des équipes présente sur Moodle, dans la section CONTENU DU COURS / Travaux / TP3.

Écrivez-moi un courriel me spécifiant les membres de votre équipe (**en mettant l'autre coéquipier en copie**). Le courriel doit être formé comme suit :

SUJET du message : **INF1120-GROUPE : Équipe TP3 (**où **GROUPE** doit être remplacé par votre groupe).
CORPS du message : vos **2 codes permanents** doivent être dans le corps du message.

J'inscrirai les équipes dans le fichier sur Moodle à mesure que je recevrai vos courriels.

VOUS DEVEZ remettre vos fichiers sur **UN SEUL compte Moodle**, celui du membre de l'équipe dont le code permanent est le plus petit (alphabétiquement parlant).

5.2 POLITIQUE CONCERNANT LES RETARDS

AUCUN RETARD NE SERA ACCEPTÉ

5.3 REMARQUES GÉNÉRALES

- **N'oubliez pas d'écrire (entre autres) votre/vos nom/s complet/s et votre/vos code/s permanent/s dans l'entête de la classe à remettre (et toutes les autres classes, s'il y a lieu).**
- Aucun programme reçu par courriel ne sera accepté (à moins d'une entente préalable).
- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Dropbox, Google Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir du temps supplémentaire pour la remise de votre travail.
- N'attendez pas à la dernière minute pour commencer le travail, vous allez fort probablement rencontrer des problèmes inattendus!

Ce travail est strictement individuel. Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !