M2GL - V & V
TP5 & TP6

Romain LE HÔ & Yves DEMIRDJIAN

14 novembre 2012

Chapitre 1

TP5 - Test d'IHM SWING

1.1 Notes

Toutes modifications apportées dans le code sont précédées d'un commentaire informant que ces dites modifications proviennent de nous.
Les tests de ce tp se trouvent dans le fichier *WTTests.java*dans le pacquage *test*.

1.2 Tests proposés dans l'énoncé du TP

1.2.1 All text fields shoud be disabled at application startup.

Au lancement manuel de l'application, les champs textes semblent Disabled. Un test nous montre cependant qu'à part deux composants, tous sont des LabeledText, qui ont bien leur état isEnabled à False. Mais State et ZIP, ne sont pas des LabeledText, mais des JTextComponent.

Ceux-ci sont Disabled, cependant l'inspection ne trouve pas de label pouvant les lier au champ texte correspondant ("ZIP" ou "State"). Cela a pour conséquence de faire échouer les tests JUnit quand on les lance une fois qu'ils ont été générés.

D'après le code (AddressPanel) nous montre que les labels des différentes rubriques sont toujours déclarées avant la boite de texte liée, sauf pour ZIP et State. On déplace ceux-ci, et relançons les inspections : cette fois ZIP et State sont considérés comme des LabeledText et leur état isEnabled à False est testable grâce à JUnit.

1.2.2 All text fields shoud be enabled after clicking the new button.

Ce test fonctionne parfaitement des la première exécution. Tous les champs de texte sont Enabled après qu'on ait cliqué sur New.

1.2.3 The button for delete should be enabled when there's a selected item in the list.

Ce test, fonctionne, une fois qu'on prend garde de tester juste, dans JListLocator(), le "Nom, prénom" en paramètre et sans le numéro d'index à la suite (comme initialisé automatiquement). (voir code)

1.2.4 The Save button should not be enabled right after clicking the New button.

Le test manuel montre que le bouton Save est bien Enabled après un New, ce qui n'est pas conforme à ce qu'on attend. Il faut donc corriger l'application pour que seule une modification entraîne l'activation du bouton Save et rendre celui-ci désactivé par défaut.

Ceci est fait en ajoutant des KeyListeners dans l'AddressPanel et une fonction addListenersToTextFields(), de manière à ce qu'à chaque pression d'une touche dans un JTextField, on informe AddressActionPanel. Pour simplifier, seuls les champs FirstName, LastName sont obligatoires pour activer le bouton.

1.2.5 The button for Delete should not be enabled where there isn't a selected item in the list.

Le test regarde si le bouton Delete est Disabled au démarrage. Le test échoue en première exécution, car Delete est toujours activé.

De nouveau, on ajoute un addListenersToAddressList à AdressListPanel (fonction addListenerToList() dans AddressListPanel), afin d'être notifié lorsqu'un élément est sélectionné ou non.

1.3 Tests supplémentaires

1.3.1 The button for edit should not be enabled where there isnt a selected item in the list.

De la même manière que pour le test précédent avec Delete, Edit ne doit être possible que si un item de la List est séléctionné. Nous avons donc rajouté les quelques lignes manquantes dans le listener que nous avons précédemment créé afin de gérer l'état Enable du bouton.

1.3.2 The button for edit should be enabled where there is a selected item in the list.

Nous ajoutons une personne, la sauvons et nous vérifions qu'après sélection de celle-ci le bouton Edit est Enabled. Le test a été concluant immédiatement grâce au rajout du code du test précédant.

1.3.3 The button for cancel should not be enabled when we are not editing an address

Le test a tout d'abord échoué, car le bouton Cancel était toujours activé. Nous l'avons donc désactivé par défaut puis dans la fonction setEditable de AddressPanel il nous a suffit de changer l'état Enabled du bouton en fonction du paramètre passé à la fonction.

1.3.4 The button for cancel should not be enabled when we are not editing an address

Le test consiste à vérifier que le bouton est désactivé avant l'appui sur le bouton New ou après l'appui sur le bouton Cancel. Le test a tout d'abord échoué, car le bouton Cancel était toujours activé. Nous l'avons donc désactivé par défaut puis dans la fonction setEditable de AddressPanel il nous a suffit de changer l'état Enabled du bouton en fonction du paramètre passé à la fonction.

1.3.5 The button for cancel should be enabled when we are editing an address

Le test a immédiatement fonctionné suite aux modifications précédemment apportées. Le test consiste à simuler l'appui sur le bouton New et vérifier que le bouton Cancel est bien activé.

Chapitre 2

TP6 - Test daccès à une base de donnée

2.1 Notes

Toutes modifications apportées dans le code sont précédées d'un commentaire informant que ces dites modifications proviennent de nous.
Les tests de ce tp se trouvent dans le fichier *DBTests.java*dans le pacquage *test*.

2.2 Tests proposés dans l'énoncé du TP

2.2.1 Créer deux contacts avec le même tuple (LASTNAME, FIRSTNAME, MIDDLENAME, EMAIL) et vérifier que le contact n'est pas dupliqué.

Pour tester le doublon (méthode testAddDuplicatedContactData), nous avons créé une boucle enregistrant les mêmes données à la suite. La fonction getRowCount() de IDatabaseConnection est très utile pour l'assertion. Le test à d'abord échoué.
Afin d'y remédier, nous avons créé notre propre opération de comptages de lignes dans le fichier AddressDao.java.
Ensuite nous avons vérifié à la fonction saveRecord et editRecord qu'il n'existait pas précédemment de lignes dont le nom, le prénom, le deuxième prénom et l'adresse e-mail était identique à l'enregistrement en cours d'insertion. Le test à été concluant par la suite.

2.2.2 Vérifier qu'un nouveau contact dont tous les champs sont vides ne peut pas être créé.

Pour tester que les champs ne soit pas tous vides (méthode testAddContactWithEmptyTextFields), nous avons utilisé WindowTester pour qu'il rentre un texte vide dans le champs du nom afin d'activer le bouton save. La fonction getRowCount() de IDatabaseConnection nous a permis de tester si le nombre de lignes était identique avant et après l'insertion. L'insertion devant échouer si au moins un des champs n'est pas rempli. Le test à d'abords échoué, donc une insertion a bien eu lieu.
Afin d'y remédier, nous avons complété notre condition du précédent test dans le fichier AddressDao et la fonction saveRecord afin de tester la valeur de chacun des champs.

2.3 Tests supplémentaires

2.3.1 Vérifier que la suppression d'un contact entraîne bien la suppression dans la base de donnée.

Méthode testDeleteContact. Il nous a suffit de tester le nombre de lignes avant et après la suppression. Puis de tester qu'il s'agit du bon contact ayant été supprimé. Le test a tout de suite fonctionné.