UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

IUT DE VÉLIZY

Projet : Simplexe

DUT Informatique 2ème année
Formation initiale
IUT de Vélizy
Université de Versailles-St-Quentin-en-Yvelines

Étudiant(s) :

Claire BAUCHU, Louis BIZOT, Damien CHANCEREL, Victor CHARDERON, Martin NIOMBELA, Adam SERRAKH

Enseignant(s) :

M. AUGER, M. MARTEL

Dossier :

Dossier de test du projet tutoré

Sommaire

1 Introduction

Ce document contient l'ensemble des tests à mener sur les différentes parties de notre programme pour s'assurer de sa fiabilité avant de présenter le produit fini à nos clients.

Ces tests sont cruciaux et sont déterminés avant le début du développement pour permettre une meilleure appréciation des critères d'acceptation. Cela nous permet donc, en plus du dossier d'analyse des besoins et du dossier concepteur de déterminer le plus précisément possible les fonctionnalités et l'apparence générale du produit fini.

Pour effectuer ces tests, nous avons choisi des méthodes et des outils précis. Ces éléments seront développés dans ce document, ainsi que la liste précise des tests que nous allons effectuer avant de présenter le produit fini au client.

2 Procédures

2.1 Type de test

Les tests que nous allons effectuer sur le modèle et la vue seront dits en « boîte noire ». En effet, nous allons établir des matrices de tests avec différents types de données « entrées » et comparer les sorties avec les sorties attendues. Dans ce document, les sorties attendues seront mentionnées, ce qui nous permettra lors des tests de conclure sur la validité du modèle.

Utiliser des tests en boîte noire nous permet de déterminer les cas pouvant mener à des erreurs (des cas dits « limites », par exemple une division par 0). Cela nous semble être le plus judicieux pour pouvoir mettre plus tard notre programme à l'épreuve de ces tests, et ainsi empêcher un maximum d'erreurs sur le produit final.

2.2 Outils

Nous allons utiliser JUnit, un *framework* de test unitaire pour Java. Nous allons donc créer un package dans notre application comportant des tests implémentés grâce à JUnit. Ce package s'appellera `TestSimplexe` et comprendra une classe de test par classe implémentée dans notre application.

3 Tests du modèle

3.1 Fraction

Entrées	½	½	½	½	1	½			
	½	- ½	0	- ½	½	-2			
Additionner	1	0	½	-1	$^{3}/_{2}$	$^{-3}/_{2}$	Sorties Fraction + Fraction		
Diviser	1	-1	Erreur	1	2	¼	Sorties Fraction / Fraction		
Multiplier	¼	-¼	0	¼	½	-1	Sorties Fraction * Fraction		

3.2 Monôme

Entrées	-2x	3x	¼x	0x	-2x	2x	½ x	5x	
	2x	-2x	¾ x	4x	-1	-1	2	0	
Additionner	0	x	x	4x					
Multiplier					2x	-2x	X	0	

3.3 Contrainte explicite

Entrées	x1+2x2	x1+2x2	X3=x1+2x2	X3=x1+2x2	X3=x1+2x2	X3=x1+2x2
	-4x3	+0x3	« x2 »	« x3 »	« X2 », X2=3x1	« X1 », X1=-2x2
AjouterMonome	x1+2x2-4x3	1x1+2x2				
RentrerBase			X2=-(1/2)x1+(1/2)x3	Erreur		
Echanger					X3=7x1	X3=0

3.4 Fonction économique

Entrées	x1+2x2	x1+2x2	X3=x1+2x2	X3=x1+2x2
	-4x3	+0x3	« X2 », X2=3x1	« X1 », X1=-2x2
AjouterMonome	x1+2x2-4x3	1x1+2x2		
Echanger			X3=7x1	X3=0

3.5 Simplexe

Entrées	x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -1x1 -1x2 -2x3 -2x4 x7 = 24 -1x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4	x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -1x1 -1x2 -2x3 -2x4 x7 = 24 -1x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4
	« x1 », « x6 »	« x3 », « x5 »
Echanger	x5 = 8 -2x2 -1x3 -3x4 + 2x6 x1 = 17 -1x2 -2x3 -2x4 -1x6 x7 = 7 -1x2 -1x3 -1x4 + 1x6 z = 119 + 2x2 + 4x3 + 3x4 - 7x6	x3 = 42/5 -2/5x1 -4/5x2 -7/5x4 -1/5x5 x6 = 1/5 -1/5x1 + 3/5x2 + 4/5x4 + 2/5x5 x7 = -6/5 + 1/5x1 + 2/5x2 + 6/5x4 + 3/5x5 z = 756/5 -1/5x1 -27/5x2 -41/5x4 - 18/5x5

3.6 Historique

Entrées	`Historique = LinkedList(simplexe1, simplexe2)`	`Historique = LinkedList(simplexe1, simplexe2, simplexe3)`
	Simplexe3	
AjouterSimplexe	`Historique.size() = 3`	
SimplexePrécédent		`Historique.size()=2` `Historique = LinkedList(simplexe1, simplexe2)`

4 Tests de la vue

Les tests dans la vue sont en boîte noire également. Ainsi, nous n'observons que la situation initiale, puis les éventuels changements effectués dans la vue. Cela nous permet donc de vérifier le fonctionnement du Contrôleur (qui fait le lien entre le Modèle et la Vue) par la même occasion.

4.1 PanelCréerSimplexe

Ce panel est un formulaire en deux parties demandant d'abord le nombre de contraintes et de monômes souhaité, puis un formulaire avec autant de ligne et de colonnes que les nombres entrés précédemment apparaît, où l'utilisateur doit entrer des `int` correspondant aux coefficients des monômes. Une fois validé, le simplexe est affiché dans la vue et l'historique créé contient le nouveau simplexe.

Entrées (Nombre de contraintes et nombre de monômes)	2,2	2,2
Valeurs des coefficients	X3 =-2x1 -4x2 X4 =-1x1 -1x2 z = 7x1 + 9x2	X3 =- Ax1 +/*x2 X4 =- x1 *1x2 z = 7x1 + 9x2
AjouterSimplexe	OK. La vue affiche le simplexe ainsi créé. `Historique.size()=1`.	Erreur. **Pop-up d'erreur** : « Saisie incorrecte »

4.2 PanelChargerSimplexe

Le fichier choisi pour charger le simplexe doit être compatible avec l'application et contenir un simplexe préalablement enregistré.

Entrées	Clic sur Parcourir	« simplexe.txt »	« »	« eclipse.exe »	« rapportStage.txt »
Clic sur Parcourir	Affichage d'un explorateur de dossier				
Sélection d'un dossier		Ok. La vue affiche le simplexe chargé.	Erreur. **Pop-up d'erreur** « Veuillez sélectionner un fichier »	Erreur. **Pop-up d'erreur** « Veuillez sélectionner un fichier compatible »	Erreur. **Pop-up d'erreur** « Veuillez sélectionner un fichier contenant un simplexe à charger »

4.3 PanelGénéralSimplexe

Entrées	x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -1x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4	x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4
Clic sur une inconnue dans PanelSimplexe	Clic sur « x1 » (deuxième ligne)	Clic sur « x3 » (troisième ligne)
	Echange de x1 et x6. La vue affiche le nouveau simplexe : x5 = 8 -2x2 -x3 -3x4 + 2x6 x1 = 17 -x2 -2x3 -2x4 -x6	Echange de x3 et x7. La vue affiche le nouveau simplexe : x5 = 2 -1/3x1 -2/3x2 -2x4 + 5/3x7 x6 = 1 -1/3x1 + 1/3x2 0x4 + 2/3x7

	x7 = 7 -x2 -x3 -x4 + x6 z = 119 + 2x2 + 4x3 + 3x4 -7x6 Le `panelHistorique` affiche tous les simplexes précédents et le nouveau simplexe à la fin	x3 = 8 -1/3x1 -2/3x2 -1x4 -1/3x7 z = 144 + 1x1 -3x2 -1x4 -6x7 Le `panelHistorique` affiche tous les simplexes précédents et le nouveau simplexe à la fin
Entrées	x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4	x5 = 8 -2x2 -x3 -3x4 + 2x6 x1 = 17 -x2 -2x3 -2x4 -x6 x7 = 7 -x2 -x3 -1x4 + x6 z = 119 + 2x2 + 4x3 + 3x4 -7x6
Clic sur le bouton « ? » du LabelIndication	Affiche « Echange de x1 et x6 » dans le `LabelIndication`	Affiche « Echange de x3 et x7 » dans le `LabelIndication`
Entrées	`Historique={simplexe1, simplexe2}` Simplexe 1 = x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -1x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4 [Echange de x1 et x6] Simplexe2= x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4 [La vue affiche Simplexe2]	`Historique={simplexe1}` Simplexe 1 = x5 = 42 -2x1 -4x2 -5x3 -7x4 x6 = 17 -x1 -1x2 -2x3 -2x4 x7 = 24 -x1 -2x2 -3x3 -3x4 z = 7x1 + 9x2 + 18x3 + 17x4
Clic sur « retour en arrière »	`Historique={simplexe1}` [La vue affiche Simplexe1]	`Historique={simplexe1}` [La vue affiche Simplexe1] **Pop-up** : « pas d'action à annuler »

4.4 JMenuBar

Entrées	Aucun simplexe chargé dans l'application	Un simplexe est chargé dans l'application (chargé depuis un fichier ou créé)
Clic sur Quitter	Affichage d'un pop-up demandant de confirmer	Affichage d'un pop-up demandant de confirmer
Clic sur « ? »	Affichage du mode d'emploi	Affichage du mode d'emploi
Clic sur « Créer Simplexe »	Affichage du formulaire dans `panelCréerSimplexe`	**Pop-up** : « Cette action entraînera la perte du simplexe actuel s'il n'est pas enregistré. Voulez-vous continuer ? Oui - Non»

Clic sur « Charger Simplexe »	Affichage du formulaire dans `panelChargerSimplexe`	**Pop-up** : « Cette action entraînera la perte du simplexe actuel s'il n'est pas enregistré. Voulez-vous continuer ? Oui - Non»
Clic sur « Enregistrer »	**Pop-up** : « Il n'y pas de simplexe à enregistrer. Veuillez charger ou créer un simplexe ».	Enregistrement du simplexe. La vue ne change pas

5 Conclusion

Ce dossier constitue donc un guide utile pour notre travail de développeurs tant en amont qu'en aval du code.

En amont, il nous permet de déterminer les cas devant attirer notre attention durant le code, et ainsi de mieux appréhender la charge de travail que représentent les différentes classes. Cela nous permettra également de perdre moins de temps durant le code, puisque les cas « limites » seront déjà pris en compte, ou au moins connus du développeur.

En aval, ce dossier va nous permettre de valider notre projet. En effet, après le code de chaque classe, cette dernière passera les tests énoncés dans ce dossier. Par ailleurs, ces tests pourront être effectués par une autre personne que celle qui a codé la classe en question, puisque les tests sont en boîte noire : il n'y a pas besoin de connaître le code source (ni même de le lire) pour pouvoir valider ces tests.

Ce dossier nous permet donc d'éviter et de supprimer les erreurs éventuelles dans notre application.