Cours Tests et validation des Logiciels

Mise en oeuvre des tests avec JUnit

Décembre 2008

Présentation de l'étude

Supposons que l'on ait à tester une fonction *Lendemain* qui calcule le lendemain d'une date passée en paramètre et définie par trois entiers : Jour, Mois, et Année. On considère (arbitrairement) que l'année doit être plus grande que 1000 et plus petite que 3000. On doit prendre en compte les années bissextiles.

Construisons maintenant des jeux de tests pour la fonction *Lendemain* en utilisant la technique des classes d'équivalence. Pour cela on commence par prendre en compte les contraintes élémentaires sur les paramètres en entrées. Puis on affinera ces contraintes pour enfin prendre en compte les contraintes liant les entrées et les sorties.

Classes d'équivalences valides	Clases d'équivalences invalides
Jour ∈ [1, 31]	Jour < 1 Jour > 31
Mois ∈ [1, 12]	Mois < 1 Mois > 12
Année ∈ [1000, 3000]	Année < 1000 Année > 3000

Tableau 1: Premier partitionnement

Après ce premier partitionnement élémentaire, on applique la règle générale qui dit qu'une classe complexe pouvant faire apparaître des comportements distincts doit être fractionnée en classes plus petites. Ainsi, il paraît utile de distinguer les mois à 30 jours des mois à 31 jours et mettre à part le mois de février. De même, nous pouvons penser que le traitement des années bissextiles n'est pas le même que celui des années non bissextiles et que l'an 2000 est un cas particulier intéressant à différencier. La séparation des classes n'entraîne pas nécessairement la création de nouvelle classe invalide car celles-ci coïncident avec des classes existantes (par exemple, un mois qui n'est pas un mois à 30 jours, sera soit un mois à 31 jours soit le mois de février soit un mois invalide déjà pris en compte).

Classes d'équivalences valides		
Jour ∈ [1, 31]		
~~Mois ∈ [1, 12]~~	Mois ∈ {1, 3, 5, 7, 8, 10, 12}	
	Mois ∈ {4, 6, 9, 11}	
	Mois = 2	
~~Année ∈ [1000, 3000]~~		Année bissextile
		Année non bissextile
		Année = 2000

Tableau 2 : On affine les partitions

On remarque sur cet exemple qu'à moins de compliquer la définition des classes, certaines valeurs peuvent apparaître dans deux classes ; par exemple la valeur d'année 2000 appartient à deux classes.

Si l'on s'arrête ici, nous avons bien pris en compte les contraintes sur les paramètres en entrée mais nous n'avons pas inclus les contraintes liant les entrées et les sorties (et donc l'aspect fonctionnel du composant). Il s'agit en effet ici de tester une fonction qui calcule la date du lendemain et donc, les valeurs d'entrée caractérisant une fin de mois ou une fin d'année donneront des valeurs de sortie

particulières (début de mois ou début d'année). Il est donc intéressant d'intégrer ces contraintes et donc, en partitionnant des classes existantes, de faire apparaître de nouvelles classes d'équivalence valides :

Classes d'équivalences valides		
~~Jour ∈ [1, 31]~~	Jour ∈ [1, 28]	
	Jour = 29	
	Jour = 30	
	Jour = 31	
~~Mois ∈ [1, 12]~~	~~Mois ∈ {1, 3, 5, 7, 8, 10, 12}~~	Mois ∈ {1, 3, 5, 7, 8, 10}
		Mois = 12
	Mois ∈ {4, 6, 9, 11}	
	Mois = 2	
~~Année ∈ [1000, 3000]~~		Année bissextile
		Année non bissextile
		Annéee = 2000

Tableau 3 : On affine de nouveau les partitions

Nous avons donc quatre classes d'équivalence valides pour les jours, quatre pour les mois et trois pour les années. Nous avons deux classes d'équivalence invalides pour les jours, deux pour les mois et deux également pour les années.

Du fait que les paramètres en entrée sont liés, certaines combinaisons de valeurs valides conduisent à des entrées invalides ; par exemple, le 29 février 2004 ou le 31 avril 1997 sont deux dates invalides alors que chaque valeur prise individuellement est valide. Les spécifications permettent ici de détecter ce problème très facilement ; ce n'est pas malheureusement toujours le cas.

Nous pouvons maintenant générer les jeux de tests. Si l'on se contente du critère « All Singles », quatre valeurs de test permettent de tester toutes les classes valides :

Jeux de valeurs	Résultat attendu	Clases d'équivalences couvertes
(14, 7, 2008)	(15, 7, 2008)	Jour ∈ [1, 28] Mois ∈ {1, 3, 5, 7, 8, 10} Année bissextile
(29, 12, 1997)	(30, 12, 1997)	Jour = 29 Mois = 12, Année non bissextile
(30, 4, 2000)	(31, 4, 2000)	Jour = 30 Mois ∈ {4, 6, 9, 11} Année = 2000
(31, 12, 1999)	(1, 1, 2000)	Jour = 30 Mois = 12 Année non bissextile

Tableau 4 : Premiers jeux de tests pour la fonction Lendemain

A cela, il faut ajouter des jeux de tests correspondant aux classes invalides ; on prend garde ici à ne couvrir qu'une seule classe invalide afin de bien tester individuellement l'effet de l'entrée invalide.

Jeux de valeurs	Clases d'équivalences couvertes
(-10, 2, 2000)	Jour < 1
(33, 12, 2001)	Jour > 31
(2, 0, 1999)	Mois < 1
(3, 14, 1997)	Mois > 12
(1, 1, 100)	Année < 1000
(1, 1, 4000)	Année > 3000

Tableau 5 : Prise en compte des classes invalides

Si l'on veut tester plus à fond la fonction Lendemain le testeur peut choisir la stratégie « All Pairs » pour générer les jeux de valeurs de test. Cela conduit à générer 4*4 = 16 jeux de valeurs distincts :

Classes d'équivalence « Jour »	Classes d'équivalence « Mois »	Classes d'équivalence « Année »	Jeux de valeur
Jour ∈ [1, 28]	Mois ∈{1, 3, 5, 7, 8, 10}	Année bissextile	(14, 5, 2004)
Jour = 29	Mois = 12	Année non bissextile	(29, 12, 1999)
Jour = 30	Mois ∈ {4, 6, 9, 11}	Année = 2000	(30, 6, 2000)
Jour = 31	Mois = 2	Année bissextile	(31,2, 1006)
Jour ∈ [1, 28]	Mois = 2	Année non bissextile	(15, 2, 1789)
Jour = 29	Mois ∈{1, 3, 5, 7, 8, 10}	Année = 2000	(29, 12, 2000)
Jour = 30	Mois = 12	Année bissextile	(30, 12, 1992)
Jour = 31	Mois ∈ {4, 6, 9, 11}	Année non bissextile	(31, 9, 1995)
Jour ∈ [1, 28]	Mois ∈ {4, 6, 9, 11}	Année = 2000	(7, 11, 1967)
Jour = 29	Mois = 2	Année bissextile	(29, 2, 1964)
Jour = 30	Mois ∈{1, 3, 5, 7, 8, 10}	Année non bissextile	(30, 7, 1903)
Jour = 31	Mois = 12	Année 2000	(31, 12, 2000)
Jour ∈ [1, 28]	Mois = 12	Année bissextile	(8, 12, 1888)
Jour = 29	Mois ∈ {4, 6, 9, 11}	Année non bissextile	(29, 9, 1805)
Jour = 30	Mois = 2	Année = 2000	(30, 2, 2000)
Jour = 31	Mois ∈{1, 3, 5, 7, 8, 10}	Année bissextile	(31, 3, 2012)

Tableau 6 : Jeux de tests selon le critère "All Pairs" pour la fonction Lendemain

A ces jeux de tests il convient d'ajouter ceux du tableau concernant la prise en compte des classes invalides. On obtient alors une qualité des jeux de tests qui laisse peu d'espoir à une erreur de rester inaperçue ! On remarquera que le jeu de test correspond au calcul du lendemain du 31 décembre 2000 est bien construit par cette méthode mais qu'il faudrait rajouter manuellement le test du lendemain des 28 et 29 février 2000.

Travail à faire

On suppose que l'on ait la classe MyDate.java (voir données annexes). Écrire des tests pour les méthodes MyDate(), checkData() et nextDay(), cette dernière méthode étant à écrire. Vous prendrez garde que certaines méthodes sont « statiques » c'est à dire qu'elles s'utilisent indépendamment de tout objet. Vous utiliserez les données prédéterminées par la méthode des partitions, ce qui pourra vous conduire à définir plusieurs méthodes de test pour une même méthode à tester.