Deuxième étude de cas

« Tests et classes d'équivalences »

On se place dans le contexte où il faut tester une fonction `Lendemain` qui calcule le lendemain d'une date passée en paramètre. La date est définie par 3 entiers : `jour`, `mois`, et `année`. On considère que l'année doit être plus grande que 1000 et plus petite que 3000. On doit prendre en compte les années bissextiles.

Question 1) Est-il nécessaire d'avoir le code pour construire des jeux de tests pertinents ? Comment s'appelle l'approche où les tests sont construits avant le code ? Est-ce une technique intéressante ? pourquoi ?

Réponse :

Non

Test Driven Development (TDD)

Exigences bien définies, qualité de code, réduction de bugs probables

Question 2) En quoi les stratégies de tests basés sur les limites et sur les classes d'équivalence sont-elles complémentaires ?

Réponse :

Coouverture des cas critiques + couverture des cas généraux -> réduction du nombre de cas

à tester

On va maintenant construire des jeux de tests pour la fonction `Lendemain` en utilisant la technique des classes d'équivalence et la technique des tests aux limites. Pour cela on va partir des contraintes élémentaires sur les paramètres en entrées, affiner ces contraintes et enfin prendre en compte les contraintes liant les entrées et les sorties (et donc intégrer des aspects fonctionnels).

Question 3) Définir 2 classes d'équivalences simples pour la fonction `Lendemain` basées sur la notion de date valide.

Classes d'équivalences valides	Clases d'équivalences invalides
jour E [1, 31] mois E [1, 12] année E [1000, 3000]	jour < 1 ll jour > 31 mois < 1 ll mois > 12 année < 1000 ll année > 3000

Question 5) Raffiner la classe d'équivalence invalide en classes plus fines.

Clases d'équivalences invalides
jour < 1
jours > 31
mois < 1
mois > 12
annee < 1000
annee > 3000

Question 4) Découper la classe d'équivalence valide en intégrant progressivement des contraintes plus fines sur les dates ; pour chaque classe valide, n'oubliez pas de définir une classe « invalide ».

Classes d'équivalences valides	Clases d'équivalences invalides
Mois à 30 jours mois E {4, 6, 9, 11} 1 < jour < 30	mois E {4, 6, 9, 11} jour < 1 ou jour > 30
Mois à 31 jours mois E {1, 3, 5, 7, 8, 10, 12 } 1 < jour < 31	mois E {1, 3, 5, 7, 8, 10, 12 } jour < 1 ou jour > 31
Mois de février année bissextile mois = 2 + année bissextile 1 < jour < 29	mois = 2 + année bissextile jour < 1 ou jours > 29
Mois de février année non bissextile mois = 2 + année non bissextile 1 < jour < 28	mois = 2 + année non bissextile jour < 1 ou jours > 28

Question 5) Intégrez maintenant des contraintes liées au calcul effectué par la fonction pour raffiner les classes d'équivalence.

Classes d'équivalences valides	Clases d'équivalences invalides
Fin de mois à 30 jours mois E {4, 6, 9, 11} jour = 30	
Fin de mois à 31 jours mois E {1, 3, 5, 7, 8, 10, 12 } jour = 31	
Fin de mois février année bissextile mois = 2 & ab jours = 29	
Fin de mois février année non bissextile mois = 2 & !ab jours = 28	
Fin d'année non bissextile mois = 12 & jour = 31 et !ab	
Fin d'année bissextile mois = 12 & jour = 31 et ab	

Question 6) Générer les jeux de test pour couvrir les classes d'équivalences.

Jeux de valeurs	Clases d'équivalences couvertes
(30 / 4 / _)	
(29 /4 / _)	
(31 / 1 / _)	
(30 / 1 / _)	
(29 / 2 / ab)	
(28 / 2 / ab)	
(28 / 2 / !ab)	
(27 / 2 / !ab)	
(31 / 12 / !ab)	
(31 / 12 / ab)	