TP spécification et test

Exercice 1 :

Le code est il facilement vérifiable par le test unitaire ? Quel est le problème?

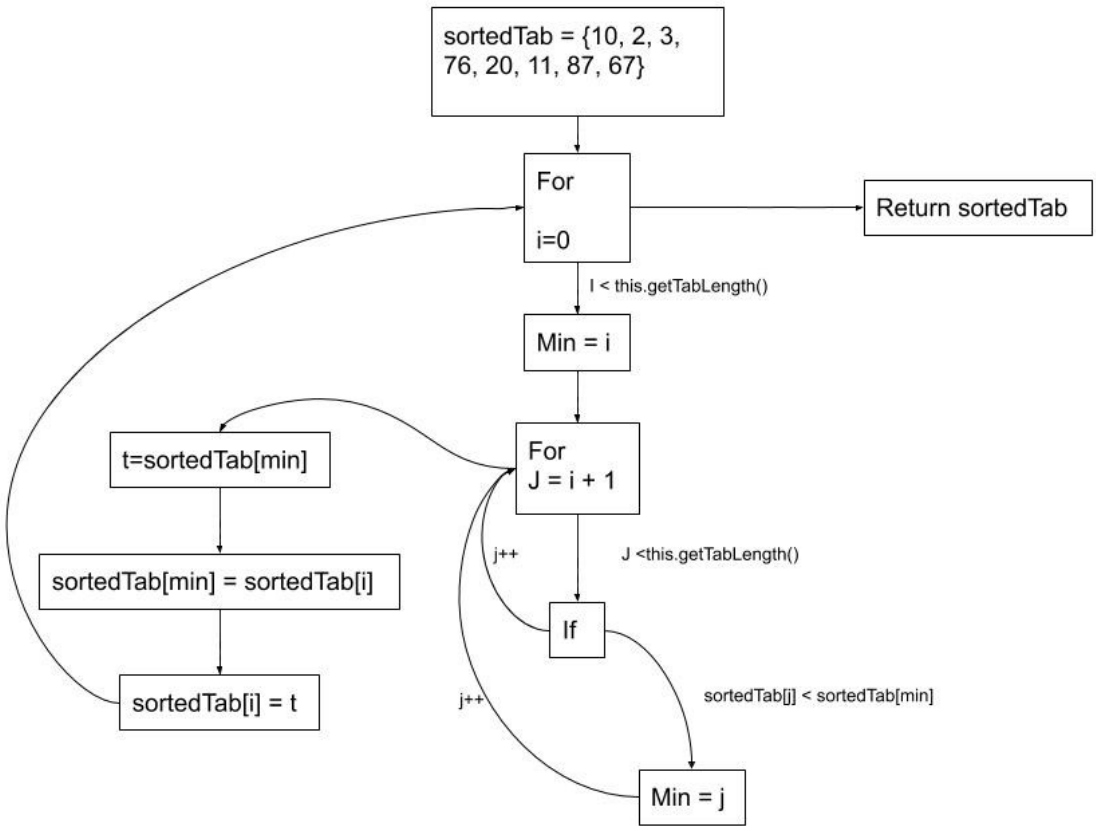
Le code n'est pas facilement vérifiable car toutes les méthodes sont des procédures (elles ne retournent pas de résultat). Il faut donc appelé la variable a, qui est le tableau trié à chaque fois que l'on veut tester le tableau.
De plus, étant donné que toutes les méthodes et variables sont « static », il faut nécessairement rappelé la méthode 'initialisation(String[] args)' à chaque fois que l'on veut modifier le tableau. Comme les méthodes de tri font toujours un tri croissant, on ne peut pas en tester deux sans appeler la méthode initialisation entre les deux.

Quelle est la couverture de la méthode triSelection? Est ce suffisant ?

La cocuverture des instructions est de 100 % grâce au tableau donnée précédemment. Cependant, le taux de couverture émis par jacoco ne concerne pas les chemins. Pour obtenir une couverture à 100 %, il faut commencer par calculer le nombre de McCabe (4 dans notre cas). Celui-ci nous informe qu'il y a 4 chemins différents pour retourner un resultat. Pour obtenir une couverture totale, il faut donc réaliser tester la fonction avec 4 tableaux différents pour passer par tout les chemins.

triSelection()	100,0 %	50	0

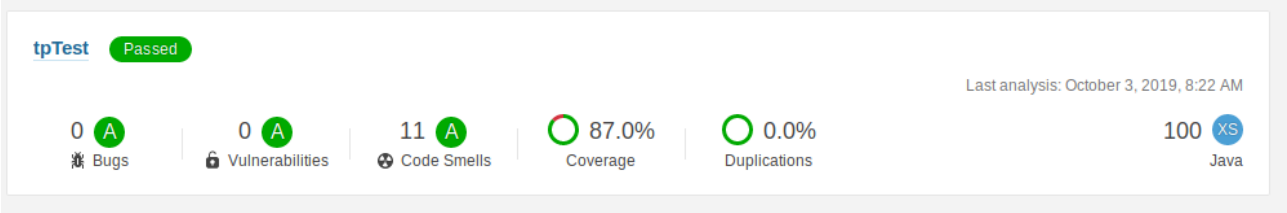
Construisez le graphe de contrôle de la méthode :



Notre tableau ne contient que 3 test différents car il est impossible de ne pas rentrer dans la 2ème boucle ‘for’. Il est donc impossible d’obtenir une couverture des chemins à 100 %.

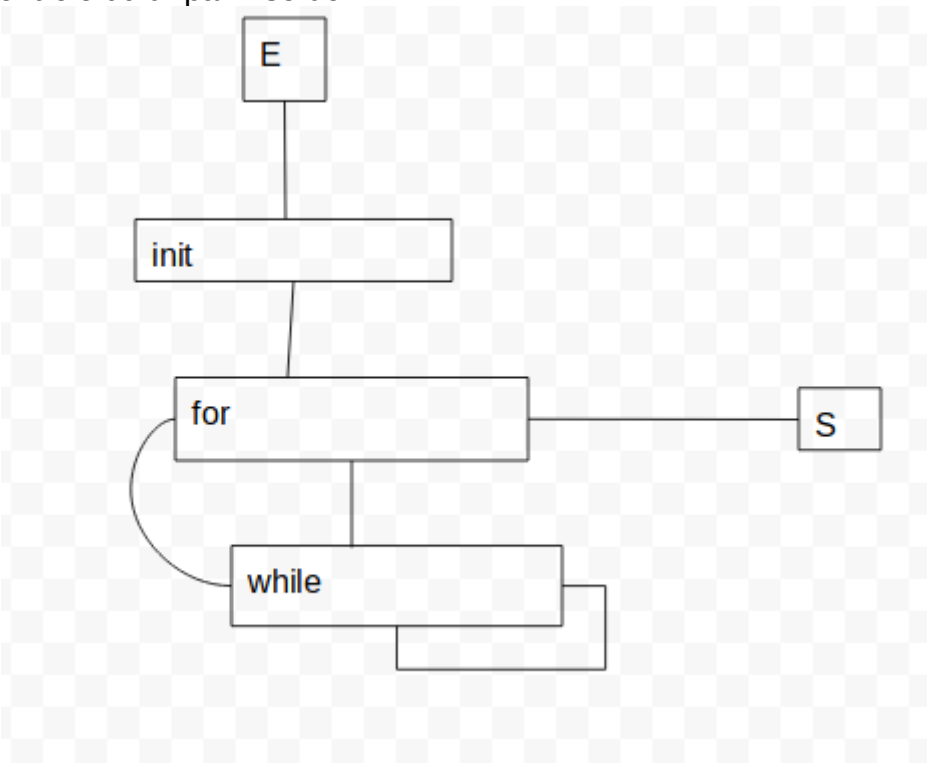
entré	sortie	résultat attendu	PASS ?
{}	{}	{}	V
{1,5,8,3,10}	{1,3,5,8,10}	{1,3,5,8,10}	V
{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	V

Données de couvertures accessible dans sonarqube :



Exercice 2 :

Graphe de contrôle du tri par insertion :



Nombre de Mc Cabe : N=5 ; A=6 ; Mc Cabe=3

Il faut donc réaliser 3 tests pour vérifier tout les chemins :

```
@Test
public void testTriInsertion() {
    Assert.assertArrayEquals(this.sortedValues, tritableau.triInsertion());
    int[] tab0= {};
    int[] tab1= {2,3,1,4};
    int[]tab2= {1,2,3,4};
    tritableau.setTab(tab0);
    Assert.assertArrayEquals(tab0, tritableau.triInsertion());
    tritableau.setTab(tab1);
    Assert.assertArrayEquals(tab2, tritableau.triInsertion());
    tritableau.setTab(tab2);
    Assert.assertArrayEquals(tab2, tritableau.triInsertion());
}
```