

Les Tests des composants

Table des matières

I. Que sont les tests de composants ?	3
II. Exercice : Quiz	4
III. Quand et comment réaliser ces tests ?	5
IV. Exercice : Quiz	6
V. Auto-évaluation	7
A. Exercice	7
B. Test.....	7
Solutions des exercices	8

I. Que sont les tests de composants ?

Durée : 1 h 30

Prérequis : tests Unitaires, Java

Environnement de travail : VSCode

Contexte

La confusion entre les tests unitaires et les tests de composants est parfois bien pratique. Néanmoins il faut avoir en tête que cette simplification peut mener à des confusions et des erreurs dans certains contextes. Dans ce cours, vous découvrirez ce que sont les tests de composants, comment les réaliser, et à quel moment du développement ils doivent être créés et utilisés. En tant que professionnel, il est important de bien faire la différence entre ces différents tests.

Fondamental

Un test unitaire permet de tester une fonction en isolation totale du reste du programme.

Définition Composant

Un composant est un bloc de code représentant une des fonctionnalités du programme final, décrites dans les spécifications techniques de ce programme.

Un composant peut prendre de multiples formes : module, classe, programme, base de donnée.

Il peut être dépendant ou non d'autres composants du programme, dans ce cas la méthode de test diffère légèrement.

Dans les deux cas, la finalité est identique : vérifier qu'un composant possède toutes les fonctionnalités requises, et que leur fonctionnement soit en accord avec les spécifications du logiciel, définies en début de projet.

Définition Test de composant

Un test de composant a pour but de vérifier qu'un composant fonctionne selon les caractéristiques définies au début du projet.

Remarque

Les tests de composants sont parfois confondus avec les tests unitaires, car, comme ces derniers, ils peuvent être réalisés en isolation du reste.

Cependant, ils en sont différents de par la nature des données d'entrée (réelles / construites pour les composants, fixes pour les tests unitaires), ainsi que pour leur cible (les tests unitaires ne servent à tester que des fonctions dans un environnement contrôlé, contrairement aux tests de composants qui s'occupent de l'interaction d'un ensemble de fonctions sur des données).

Il est possible de réaliser ces tests de deux manières différentes :

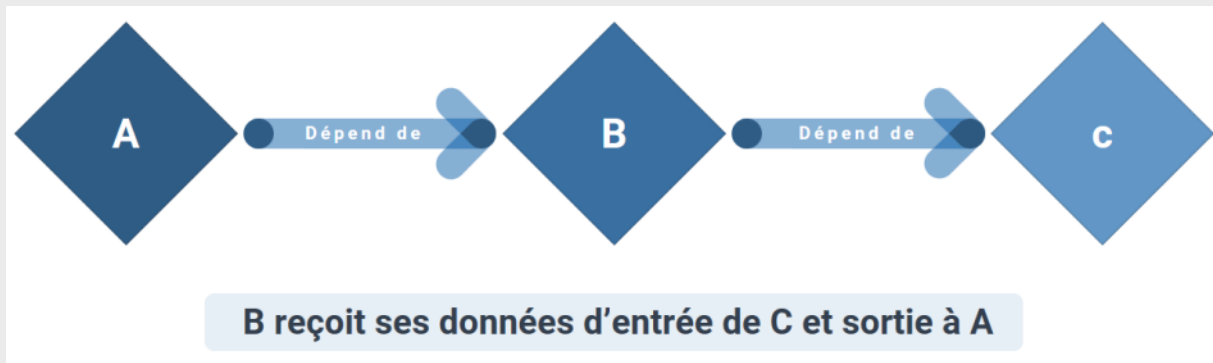
- **CTIS (Component Testing In Small)** : le composant est testé en isolation du reste du programme. Dans ce cas, les tests de composants sont parfois confondus avec les tests unitaires, car la méthode est très similaire.
- **CTIL (Component Testing In Large)** : le composant est testé avec les composants dont il dépend. Cela s'apparente aux tests d'intégration, qui servent à tester le bon fonctionnement de modules entre eux. La différence est qu'en test de composant, les dépendances sont remplacées par des objets « vides ».

Exemple CTIS

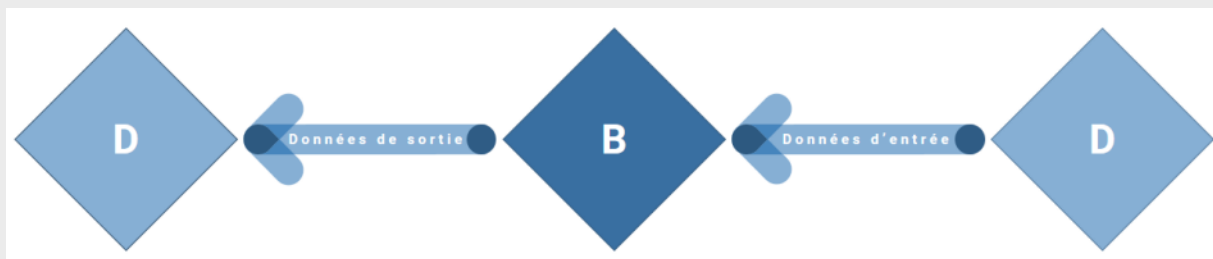
Considérons un site Internet constitué de 10 pages, considérées indépendantes. Chaque page peut être assimilée à un composant, et sera testée séparément des autres en raison de leur absence de liens.

Exemple CTIL

Prenons un programme constitué de trois composants, A, B et C, avec les dépendances suivantes :



Prenons le cas où le composant B est créé et est en attente de tests. Problème : A et C ne sont pas finis. Il est possible de contourner le problème en les remplaçant par un objet D, dont le comportement est basique : il retourne quelque chose de simple, sans opérations complexes. On peut alors tester le comportement de B.



Exercice : Quiz

[solution n°1 p.9]

Question 1

Un composant ne peut pas dépendre d'autres composants pour être testable.

- ☐ Vrai
- ☐ Faux

Question 2

Les tests de composants sont des tests unitaires.

- ☐ Vrai
- ☐ Faux

Question 3

Que signifie le sigle CTIL ?

- ☐ Clear Test In Language
- ☐ Component Testing In Large
- ☐ Component Teasing In Length

Question 4

Un composant peut être testé sans ses dépendances.

- ☐ Vrai
- ☐ Faux

Question 5

Un composant représente une fonctionnalité.

- ☐ Vrai
- ☐ Faux

III. Quand et comment réaliser ces tests ?

Les tests de composants doivent être effectués une fois que les tests unitaires du composant en question ont été réalisés. Cela permet de s'assurer qu'un problème découvert pendant cette phase ne vient pas d'une fonction mal codée, mais bien d'un problème de fonctionnalité.

Il y a 2 cas de figure à identifier au moment de réaliser ces tests :

- Si le composant est un **élément de code**, il suffit d'écrire une série de fonctions de tests utilisant un générateur de données (plutôt que des données fixes) pour les donner au composant. Ainsi, le test simule une situation réelle afin d'analyser le comportement du composant.
- Si le composant est un **élément plus complexe**, par exemple une partie d'une interface utilisateur, il est compliqué, voire impossible, d'automatiser le processus, il faut donc réaliser ces tests à la main, ce qui est plus imprécis et peut mener à des erreurs.

Si les tests unitaires ont bien été écrits, le composant ne devrait pas avoir de problème à recevoir des données erronées. Dans le cas contraire, cela signifie que les tests unitaires ont été mal réalisés, ou que la liste de fonctions de test est incomplète. Souvent, un problème émerge sur les cas limites des données d'entrée (par exemple un objet vide, ou un nombre trop grand).

Remarque

Ces tests sont rarement automatisables complètement, il est très souvent nécessaire d'introduire un mécanisme de récolte de données durant le test, pour procéder à une analyse du déroulement du test et décider de valider ou non le fonctionnement du composant.

Exemple

Un site marchand veut tester que les promotions de certains articles sont bien prises en compte au moment du calcul du prix d'un panier.

On ne s'intéresse ici pas à la manière dont ces éléments sont implémentés, seulement aux résultats qu'ils retournent.

```

1 public class TestsPanierClient
2 {
3     private PanierClient panier;
4
5     public TestsPanierClient(PanierClient p) //Initialise la session de test
6     {
7         this.panier = p;
8     }
9
10    public void Test_Promotion2Plus1() //Promotion 2 achetés 1 gratuit
11    {
12        Article article = GenererArticle();
13        this.panier.AjoutArticle(article, 3); //Ajoute 3 fois l'article créé
14        System.out.println("Coût attendu : " + article.cout * 2);
15        System.out.println("Coût réel : " + panier.Cout());
16    }
17
18    public Article GenererArticle()
19    {
20        //Génère et renvoie un nouvel article
21    }
22 }

```

Ce code ressemble fortement à un test unitaire, les différences notables étant la fonction de génération de données (ici `GenererArticle`) et les appels à plusieurs fonctions de l'objet « *panier* ».

Ces similitudes sont partagées entre les différents types de tests du code.

Note : ici, les données à analyser sont simplement affichées dans la console, pour simplifier le code. Une application réelle utiliserait d'autres méthodes, par exemple écrire dans un fichier.

Ci-dessous une capture d'écran de la barre d'outils de Google Docs :



Il est ici compliqué, voire impossible, de tester les boutons avec uniquement du code, il faudra donc qu'un testeur le fasse manuellement.

Cependant, les boutons n'ayant pas d'effets (seule l'interface est présente ici), il faudra implémenter un effet déclenché par l'appui de chaque bouton.

Exercice : Quiz

[solution n°2 p.9]

Question 1

Les tests de composants sont réalisés après les tests unitaires.

- ☐ Vrai
- ☐ Faux

Question 2

Les tests d'une interface utilisateur peuvent être automatisés.

- ☐ Vrai
- ☐ Faux

Question 3

Les composants ne changent pas de comportement face à des données erronées.

- ☐ Vrai
- ☐ Faux

Question 4

Les tests de composants doivent être validés manuellement.

- ☐ Vrai
- ☐ Faux

Question 5

Les données d'entrée des tests sont fixes.

- ☐ Vrai
- ☐ Faux

V. Auto-évaluation

A. Exercice

Vous êtes chargé de tester le bon fonctionnement d'une page web contenant plusieurs boutons ayant des effets différents.

Question

[solution n°3 p.10]

Décrivez votre démarche de test.

B. Test

Exercice 1 : Quiz

[solution n°4 p.11]

Question 1

Que signifie le sigle CTIS ?

- ☐ Console Team In Suburbs
- ☐ Component Tester Init Something
- ☐ Component Testing In Small

Question 2

Les tests de composants sont réalisés avant les tests unitaires.

- ☐ Vrai
- ☐ Faux

Question 3

Les tests de composants sont toujours automatisés.

- ☐ Vrai
- ☐ Faux

Question 4

Les composants possédant des dépendances peuvent être testés sans ces dernières.

- ☐ Vrai
- ☐ Faux

Question 5

Les données d'entrée des tests peuvent être des données réelles.

- ☐ Vrai
- ☐ Faux


Solutions des exercices

Exercice p. 4 Solution n°1**Question 1**

Un composant ne peut pas dépendre d'autres composants pour être testable.

☐ Vrai

☒ Faux


 Un composant peut être dépendant ou non d'autres composants du programme, mais la méthode de test diffère.

Question 2

Les tests de composants sont des tests unitaires.

☐ Vrai

☒ Faux

 Les tests de composants sont parfois confondus avec les tests unitaires, car, comme ces derniers, ils peuvent être réalisés en isolation du reste. Cependant, ils en sont différents de par la nature des données d'entrée et leur cible.

Question 3

Que signifie le sigle CTIL ?

☐ Clear Test In Language

☒ Component Testing In Large


☐ Component Teasing In Length

Question 4

Un composant peut être testé sans ses dépendances.

☒ Vrai

☐ Faux


 Lors d'un test sur un composant, ses dépendances sont remplacées par des objets « vides » ayant un comportement simple qui ne doit pas pouvoir amener de nouveaux problèmes.

Question 5

Un composant représente une fonctionnalité.

☒ Vrai

☐ Faux

 Un composant est la plus petite unité logicielle définissant une fonctionnalité. Les fonctions définissent des opérations, et n'en font donc pas partie.


Exercice p. 6 Solution n°2

Question 1

Les tests de composants sont réalisés après les tests unitaires.

☒ Vrai

☐ Faux


 Les tests unitaires sont réalisés en premier, pour garantir que les problèmes relevés par les tests de composants ne sont pas liés à un code défectueux.

Question 2

Les tests d'une interface utilisateur peuvent être automatisés.

☐ Vrai

☒ Faux


 Les tests liés à une interface utilisateur ne sont pas automatisables, il faut les réaliser manuellement.

Question 3

Les composants ne changent pas de comportement face à des données erronées.

☒ Vrai

☐ Faux


 Des données d'entrée erronées ne peuvent pas engendrer un comportement imprévu du composant. Dans le cas contraire, les tests unitaires ont été mal réalisés.

Question 4

Les tests de composants doivent être validés manuellement.

☒ Vrai

☐ Faux


 Ces tests sont rarement automatisables complètement, il est très souvent nécessaire d'introduire un mécanisme de récolte de données durant le test, pour procéder à une analyse du déroulement du test et décider de valider ou non le fonctionnement du composant.

Question 5

Les données d'entrée des tests sont fixes.

☐ Vrai

☒ Faux

 Les données d'entrée sont créées par un générateur de données, afin de simuler une situation réelle pour analyser le comportement du composant.

La première étape est d'identifier les éléments à tester, puis se référer aux spécifications écrites au début du projet pour connaître les résultats attendus.

Ensuite, pour chaque bouton :

- S'il est dépendant d'autres composants, créez des objets temporaires pour votre test, et identifiez-les au bouton que vous testez.
- Utilisez le bouton comme le ferait un simple utilisateur.
- Consignez le résultat et passez au suivant.

Une fois tous les boutons testés, comparez les résultats obtenus aux résultats attendus. Si tout correspond, le test est fini. Sinon, il faut analyser les erreurs pour trouver la source du problème, et répéter la séquence de test jusqu'à faire disparaître toutes les erreurs.

Exercice p. 7 Solution n°4

Question 1

Que signifie le sigle CTIS ?


- ☐ Console Team In Suburbs
- ☐ Component Tester Init Something
- ☒ Component Testing In Small

 CTIS signifie Component Testing In Small, ce test prend chaque composant en isolation des autres.

Question 2

Les tests de composants sont réalisés avant les tests unitaires.


- ☐ Vrai
- ☒ Faux

 Les tests unitaires sont réalisés en premier, pour garantir que les problèmes relevés par les tests de composants ne sont pas liés à un code défectueux.

Question 3

Les tests de composants sont toujours automatisés.

- ☐ Vrai
- ☒ Faux

 Il n'est par exemple pas possible d'automatiser les tests d'interface utilisateur.

Question 4

Les composants possédant des dépendances peuvent être testés sans ces dernières.

- ☒ Vrai
- ☐ Faux

- Q On peut tester un composant en remplaçant ses dépendances par des objets temporaires qui en jouent le rôle, en attendant que les autres composants soient terminés et testés.

Question 5

Les données d'entrée des tests peuvent être des données réelles.

☒ Vrai

☐ Faux

- Q On peut également générer des données proches du réel pour simuler une grande quantité de scénarios.