Université Abou Beckr Belkaid
Master 2 Génie Logiciel
Année Universitaire 2018/2019

Examen Final

—

Validation et vérification logiciel

—

Remarques :

— Les documents ne sont pas autorisés ainsi que les appareils électroniques (PC, Tablette, téléphone, etc).
— La clarté des réponses et du code est très importante.

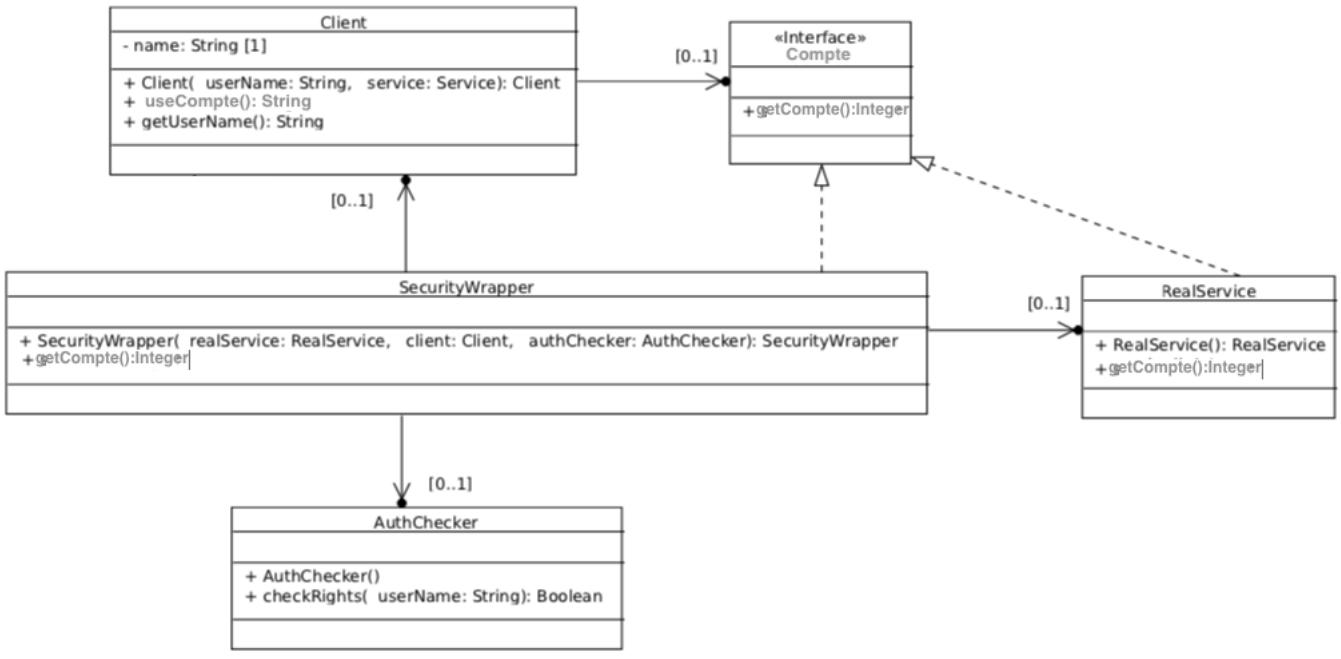
Partie A Questions de cours :

1 donner la définition des tests unitaires? Donner un exemple simple en utilisant Junit.
2 Quelle est la différence entre les tests en boite noire et en boite blanche?
3 les classes d'équivalence sont souvent utilisées durant les tests, dans quel cas? en donnant un exemple simple.

Partie B

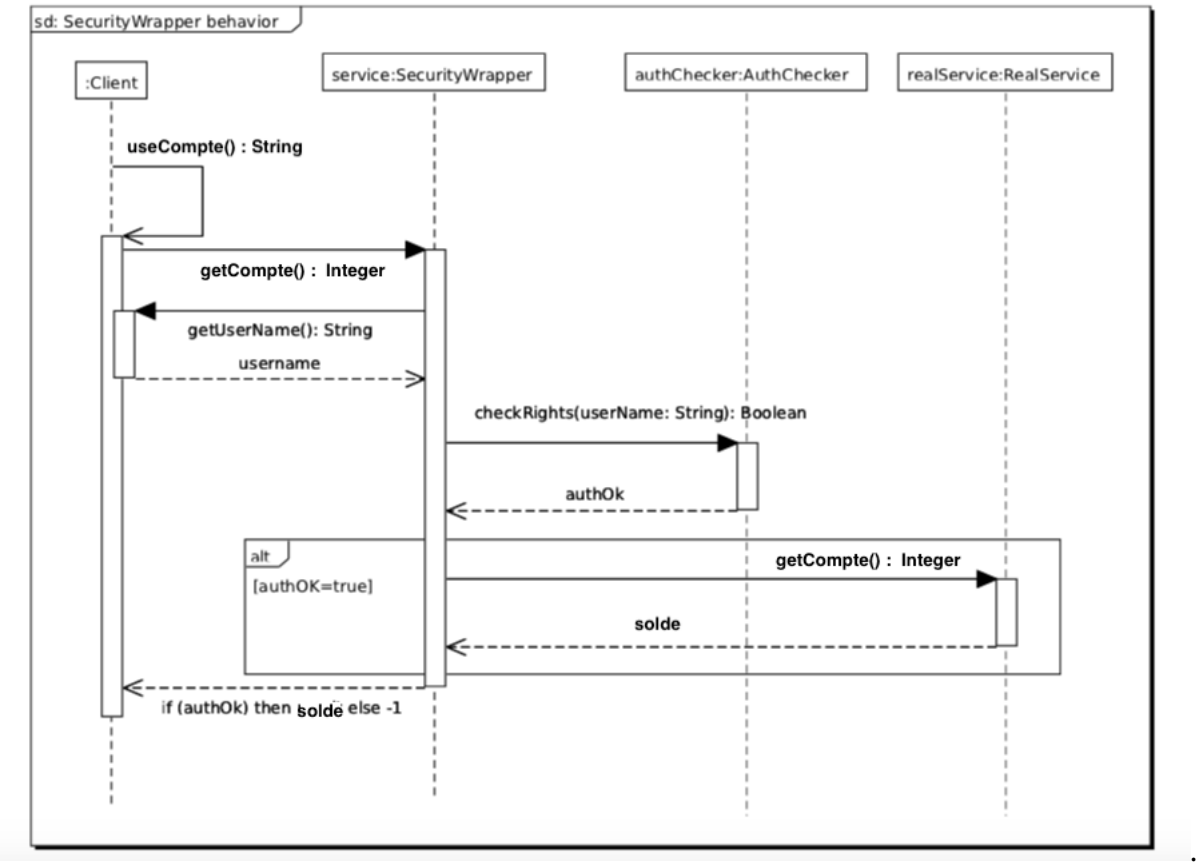
Exercice 1 :

Un système est mis en place afin de sécuriser l'accès à un service, ce système est modélisé comme suite :



.

La méthode *useCompte()* fait appel à la méthode *getCompte()* et retourne le nom du client suivi de son solde si le client à un compte et "Compte inexistant" sinon. Afin d'accéder au compte un client devra passer la barrière de sécurité en s'authentifiant. Le diagramme de séquence correspondant est donné comme suite :



Questions :

Nous souhaitons tester le comportement de *getCompte()*, du point de vue du *Client* et cela en utilisant les tests d'integration.

1 Proposer une stratégie d'integration liée à ce système, en justifiant votre choix.

2 En utilisant Mockito et en se basant sur le diagramme de séquence donné, donner l'implémentation de la classe des tests associés, et cela en testant :
   — la classe Client en isolation.
   — les interactions entre classes.

Solution :

1 Pour proposer une stratégie d'intégration, il faut construire le graphe de dépendance de test puis choisir un ordre d'intégration.

2
```
public class CompteTest(){
    private Compte mockCompte;
    private AuthChecker mockChecker;
    private RealService mockRealCompte;
    // Création du mock de la persistance et injection dans l'instance de la classe à tester
    @Before
    public void setUp() throws Exception {
        mockCompte = mock(SecurityWrapper.class);
        mockChecker = mock(AuthChecker.class);
        mockRealCompte = mock(RealService.class);
    }
    /** Les tests de la classe en isolation */
@Test
    public final void testCompteOK(){
     when(mockCompte.getCompte()).thenReturn(100000);
     Client c = new Cient("Amine",mockCompte);
    String s= c.useCompte();
        assertEquals("Amine 100000", s);
    }
@Test
    public final void testCompteKO(){
     when(mockCompte.getCompte()).thenReturn(-1);
     Client c = new Cient("Amine",mockCompte);

    String s= c.useCompte();
        assertEquals("Compte inexistant", s);
    }
```

```java
/**
   Les tests de l'interaction entre classes
 */
@Test
public final void testCompteInteraction(){

 Client c = new Cient("Amine",mockCompte);
 InOrder ordreExecution1 = inOrder(mockCompte,mockChecker);
 InOrder ordreExecution2 = inOrder(mockCompte,mockRealCompte);
 InOrder ordreExecution3 = inOrder(mockChecker,mockRealCompte);

String s= c.useCompte();
verify(mockCompte).getCompte();

ordreExecution1.verify(mockCompte).getCompte();
ordreExecution1.verify(mockChecker).checkRights();

ordreExecution2.verify(mockCompte).getCompte();
ordreExecution2.verify(mockRealCompte).getCompte();

ordreExecution3.verify(mockChecker).checkRights();
ordreExecution3.verify(mockRealCompte).getCompte();

}
```

.

Exercice 2 :

On souhaite tester, à l'aide d'une table de décision, un module qui contrôle la mise en route d'une climatisation à l'aide d'un capteur, il présente une valeur de signal entre 0 et 10. La climatisation possède un indicateur interne compris entre 1 et 3 qui vaut initialement 1. Si cet indicateur vaut 1 et que le capteur présente une valeur de signal comprise entre 1 et 5, la climatisation passe son indicateur au niveau 2 et se met en marche en mode « lent ». Si le capteur présente une valeur de signal plus grande que 5 et que l'indicateur interne vaut 2 alors la climatisation passe en mode « rapide » et passe son indicateur au niveau 3.

Questions :

1. Donnez les actions et les conditions associées à chaque action.
2. Construire la table de décisions permettant de tester cette fonction. Vous expliquerez les simplifications que vous effectuerez. Est ce que cette construction à mis en évidence des défauts de spécifications? Si oui, dites lesquels.
3. En utilisant cette table, donnez des jeux de valeurs de test pertinents pour cette fonction.

Solution :

1. lent $\Rightarrow$ indicateur = 1 and ($1 \leqslant$ capteur $\leqslant 5$) : return indicateur = 2
   Rapide $\Rightarrow$ indicateur = 2 and capteur>5 : return indicateur = 3

ind=1	ind=2	ind=3	1⩽capteur⩽5	capteur>5	Lent	Rapide
V	V	V	V	V		
V	V	V	V	F		
V	V	V	F	V		
V	V	V	F	F		
V	V	F	V	V		
V	V	F	V	F		
V	V	F	F	V		
V	V	F	F	F		
V	F	V	V	V		
V	F	V	V	F		
V	F	V	F	V		
V	F	F	V	V		
2 V	F	F	V	F	V	F
V	F	F	F	V	F	F
V	F	F	F	F	F	F
F	V	V	V	V		
F	V	V	V	F		
F	V	V	F	V		
F	V	F	V	V		
F	F	V	V	V		
F	F	V	V	F	F	F
F	F	V	F	V	F	F
F	F	F	V	V		
F	F	F	V	F		
F	F	F	F	F		

Bon courage et bonne continuation.