

TP 2 et 3 : Test d'intégration

1 Travail à réaliser

Le but est de tester et de corriger un logiciel de ventes aux enchères. Pour cela, vous devez d'abord faire un plan d'intégration, et tester unitairement chacune des classes avant de les intégrer au système. Vous utiliserez *JUnit* et *EasyMock* pour cela. Il vous est demandé de ne tester que la partie **métier**. Vous ne devez pas utiliser l'interface utilisateur pour le test.

Le diagramme de classes de la partie métier du système vous est donné en annexe.

Pour ce TP vous pouvez travailler par groupe de 4 (2 binômes). Les résultats devront être envoyés par mail (benoit.combemale@irisa.fr) avant le 8 janvier 2010 et avec comme objet « [TQL-TP2-3] Nom1-Nom2-Nom3-Nom4 ».

2 Première phase : intégration au cours du développement

Phase préliminaire

Préparez à partir du diagramme de classes un plan d'intégration.

Une fois que vous aurez créé un projet à partir des sources du système, vous pouvez relire le code afin de le comprendre et de corriger un certain nombre d'erreurs (les erreurs détectées par relecture devront quand même faire l'objet de test les mettant en évidence).

Mise en œuvre de l'intégration

Le plan d'intégration détermine l'ordre dans lequel les classes seront testées unitairement et intégrées. Certaines interdépendances et l'absence d'implémentation de certaines classes/méthodes imposent l'écriture de `stubs`.

La procédure à suivre est de commencer par tester unitairement les classes et méthodes qui ne font pas partie d'interdépendances. Dans un deuxième temps il faut écrire les `stubs` ou `mock` dans notre cas nécessaires puis tester les méthodes qui utilisent ces `stubs`. Ce travail est à réaliser.

Travail à rendre pour la première phase

- Le plan d'intégration.
- Les tests pour les classes et méthodes pouvant être testées sans utilisation de `stubs`.

3 Deuxième phase : exploitation du code achevé

Vous pourrez finir d'intégrer le système suivant votre plan d'intégration. Pour implanter les stubs ou bouchons, deux techniques sont possibles. Dans le premier cas, il est possible d'implanter/d'étendre les interfaces/classes devant être stubbées. Dans le deuxième cas, il est possible d'utiliser *EasyMock* vu en cours. Nous choisirons cette deuxième solution. *EasyMock 2* est une librairie fournissant un moyen simple d'utiliser des Mock Objects pour une interface donnée. *EasyMock 2* est disponible sous [licence MIT](#).

Les Mock Objects simulent le comportement du code métier et sont capables de vérifier s'il est utilisé comme prévu. Les classes métier peuvent être testées de façon isolée en simulant leurs objets liés par des Mock Objects.

Avantages d'EasyMock 2

- Pas d'écriture manuelle des Mock Objects.
- Supporte le refactoring sur les Mock Objects : le code de test ne sera pas cassé au runtime lors du renommage de méthodes ou de la réorganisations de paramètres
- Supporte les valeurs de retour et les exceptions.
- Supporte la vérification de l'ordre d'appel des méthodes, sur un ou plusieurs Mock Objects.

Inconvénients d'EasyMock 2

Par défaut, EasyMock supporte la génération de Mock Objects uniquement pour les interfaces. Pour ceux souhaitant générer des Mock Objects pour des classes, il existe une extension disponible sur la page d'accueil d'EasyMock.

Installation

1. Java 2 (minimum 5.0) est requis.
2. Le répertoire easymock 2.4 contient la documentation et l'API d'easymock. Ajoutez le jar d'EasyMock (`easymock.jar`) de ce répertoire dans votre classpath (clic droit sur le projet -> properties -> Java build path, onglet « Libraries »).

Utilisation

La documentation est fournie à l'adresse :

http://easymock.org/EasyMock2_5_2_Documentation_fr.html

Travail à rendre pour la deuxième phase

Pour les classes à stubbé, vous devez implanter dans votre cas de test junit la création du mock afin de grouper l'ensemble des informations liées aux tests.

Vous devez rendre une description de votre plan de test ainsi qu'un rapport de test : ensemble des cas de tests utilisés et liste des erreurs trouvées (en précisant par quel test) et corrigées. La chronologie de votre travail de test (avec entre autre la différence entre les tests sans et avec la classe `Auction` complète) devra apparaître clairement dans votre compte-rendu.

Si besoin, vous pouvez utiliser les outils vus lors des précédentes séances pour établir des critères de test.

