Ingénierie logicielle
Master Ingénierie des Contenus Numériques
Université de la Rochelle
Armelle Prigent

2019-2020

TP Spécification et Test

Le rapport de TP devra contenir toutes les réponses aux questions, les codes (intégrés au format textuel), les codes des séquences de test. Vous réaliserez le TP en binôme.
Le rendu doit se faire EXCLUSIVEMENT sous la forme d'un PDF dont le nom de fichier est votreNom1_Nom2.pdf.

Etape préliminaire - mise en place de l'environnement de test

Pour travailler, vous allez utiliser les outils suivants :

- Eclipse (en environnement JDK 1.8)
- Junit
- Java Code Coverage (plugin pour Eclipse)

Installation Junit et Java Code Coverage

- Créez un nouveau projet
- Ajouter la librairie JUnit au build path de votre projet. La librairie JUnit existe par défaut dans les versions « développeur java » d'eclipse. Si vous n'utilisez pas cette version, il suffit de l'ajouter comme un « external jar ».
- Installez le plugin eclipse : EclEmma Java Code Coverage.
- Ajoutez une méthode simpliste dans votre classe (qui réalise la somme de deux entiers et retourne la valeur).

```
public class somme {
public int somme(int a,int b){
    return a+b;
}
}
```

- Ajoutez une classe de test du type :

```
import static org.junit.Assert.*;


public class testSomme {

    @Test
    public void test() {
        somme CS = new somme();
        int s = CS.somme(40,2);
        assertEquals(s,42);
    }

}
```

- Exécutez le test Junit et vérifiez que le test passe avec succès.
- Dans le menu Run / coverage, calculez la couverture de test pour la classe somme.

Votre environnement est maintenant prêt.

Exercice 1 : Test d'une classe de tri de tableau

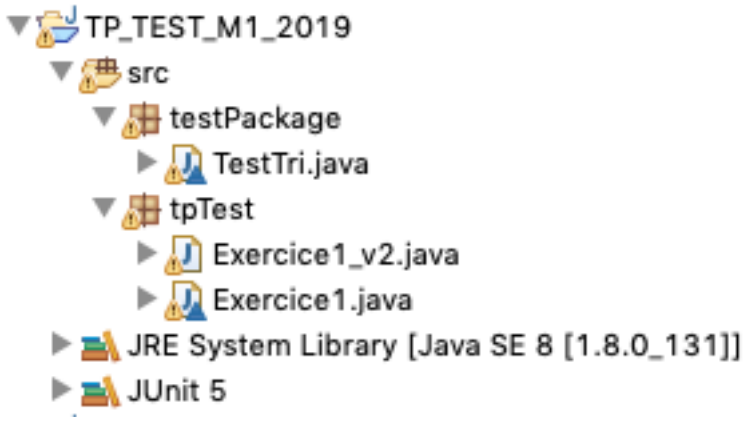
On souhaite à présent tester le code de la classe de tri de tableau disponible sur Moodle.

Etape 1

- Le code est il facilement vérifiable par le test unitaire ? Quel est le problème?
- Modifiez cette classe pour faciliter la procédure de test (vous fournirez le code modifié en annexe de votre rapport)

Etape 2 : on souhaite écrire une classe de test pour la méthode de tri par sélection

- Ajoutez un package dédié au test dans votre projet Eclipse



- Ajoutez une classe de test et un premier cas de test pour le tableau 10 2 3 76 20 11 87 67 et vérifiez que le tri fonctionne. Dans votre test, vous pouvez utiliser l'assertion : `Assert.assertArrayEquals(tab1, tab2);`

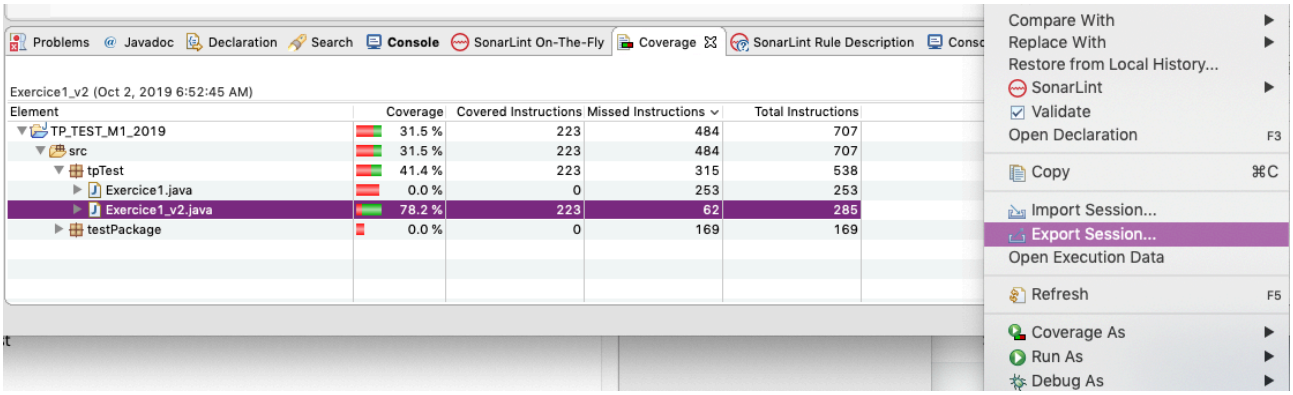
Etape 3 : couverture du test

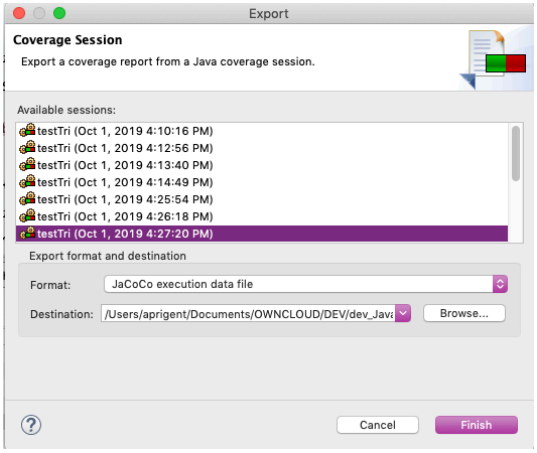
- Exécutez votre test avec Java Code Coverage
- Quelle est la couverture de la méthode triSelection? Est ce suffisant
- Construisez le graphe de contrôle de la méthode
- Donnez le tableau de test permettant d'atteindre une couverture de chemins à 100%
- Complétez votre classe de test avec ces données (un @Test par cas de test)

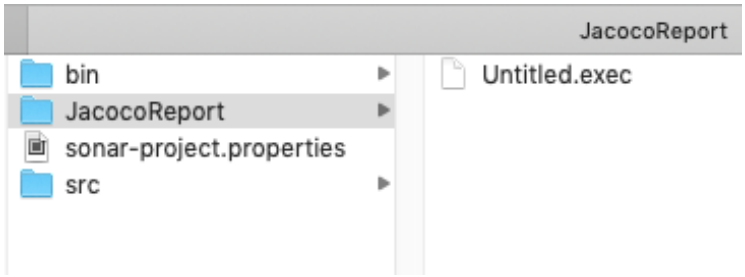
Ingénierie logicielle
Master Ingénierie des Contenus Numériques
Université de la Rochelle
Armelle Prigent

Etape 4 : Intégration de la couverture à SonarQube

- Faites un export des données de couverture dans un répertoire de votre projet







- Ajoutez un fichier sonar-project.properties dans le dossier de votre projet et configurez le chemin vers les test de la manière suivante :

sonar-project.properties

```
1  sonar-project.properties
2  # must be unique in a given SonarQube instance
3  sonar.projectKey=my:project
4
5  # --- optional properties ---
6
7  # defaults to project key
8  sonar.projectName=Tri
9  # defaults to 'not provided'
10 #sonar.projectVersion=1.0
11
12 # Path is relative to the sonar-project.properties file. Defaults to .
13 sonar.sources=src/tpTest
14 sonar.java.binaries=bin
15
16 # Tests
17 sonar.tests=src/testPackage
18 sonar.jacoco.reportPaths=JacocoReport/Untitled.exec
19 sonar.java.coveragePlugin=jacoco
20 # Encoding of the source code. Default is default system encoding
21 #sonar.sourceEncoding=UTF-8
22
```

- Lancez le serveur SonarQube puis SonarScanner dans votre répertoire. Vérifiez que vous avez accès aux données de couverture dans SonarQube.
- A partir des données produite dans SonarQube, ajoutez des tests (dans une classe de test à part) pour obtenir une couverture à 100% du tri par insertion.
- Pour ce tri, quel serait les tests permettant d'obtenir la couverture « toutes les conditions multiples » ?

Exercice 2 : Spécification et développement dirigé par le test

On souhaite développer une application permettant de calculer le nombre de jours ouvrés entre deux dates. Vous pourrez trouver un fichier csv contenant les jours fériés à l'adresse : https://www.data.gouv.fr/fr/datasets/jours-feries-en-france/#_

Etape 0 : Préparation

Créez un gitlab (https://gitlab.univ-lr.fr/) pour le développement et ajoutez en « master » les enseignants :

- noudehouenou.houssou1@univ-lr.fr
- aprigent@univ-lr.fr
- arevel@univ-lr.fr

Etape 1 : Spécification

Donnez une spécification fonctionnelle de votre application.

Etape 2 : Construction de la liste des tests

En utilisant une technique de test partitionnel, donnez sous la forme d'un tableau la liste des tests que votre système doit passer.

Etape 3 : développement dirigé par le test

Utilisez une technique de développement dirigé par le test pour implémenter votre application.

Ici, la méthode compte davantage que le résultat. A chaque nouvelle itération du DDT, vous ferez un push sur le git (en commentant avec le test concerné)
Vous fournirez dans le rapport, la liste des étapes suivies (association test - code)