1. Introduction

In this post, we will discuss the basics of setting up your JUnit Test cases. We’ll go step by step in creating test cases as we go along with creating our application. Before we dive into it though, why do we even need to create test cases? Isn’t it enough to just create the implementation since it’s what we are delivering anyway?

Although the actual implementation is part of the package, the JUnit Test case is a **bullet proof evidence** that what we wrote is what the actual requirements or functions will do. It is the concrete basis of a specific unit/function which does what it needs to do.

**Knowing the impact in the stability of the application.** The JUnit Test cases defines the stability of an application even after several extensions to it. If done correctly, it guarantees that the extension made to the system will not break the entire system as a whole. How does it prevent it? If the developers write clean unit and integration tests, it will report any side effects via the reporting plugins that the application uses.

**Regression and Integration Testing.** The effort of testing is relative to the applications size and changes done. By creating JUnit Test cases, regression and integration tests can be automated and can definitely save time and effort.

Overall, creating JUnit Test cases are definitely a must do by all developers, sadly there are still who don’t uses it’s power to it’s full extent and some just doesn’t do it. It’s sometimes a shame to think that one of the purest way of developing bullet proof code is not done by the developers. It can be because of the lack of training, experience or just pressure of not delivering the actual value (which is a problem within itself since although not part of the implementation, it’s a most valuable component of your code) but thats not an excuse especially that software are now globally taking over most of the major systems (medical, auto, planes, buildings) in the world. The stability of these systems rely on the stability of the unit test cases.

So as a precursor to being a skilled full blown developer that loves to do unit test, let’s dive into some of the beginners guide into doing it.

2. Tools

For this example, I’ll be using Java as the platform, Eclipse as the IDE, and Maven as the project management tool. If you’re not yet familiar with these tools, please visit the Java, Eclipse IDE and Maven site.

3. Step by Step Guide

3.1 Create your project

Let’s create a project first.

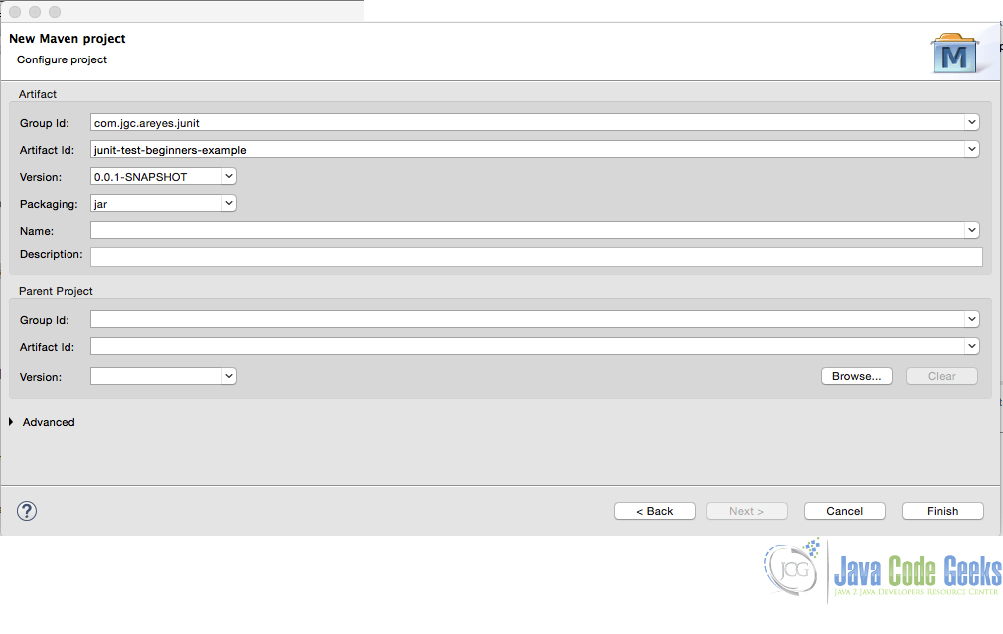
[](http://examples.javacodegeeks.com/wp-content/uploads/2015/10/new_maven_junit_tst.jpg)

Figure 1.0 New Maven Project

After creating the project, you’ll be shown a project like the one below:

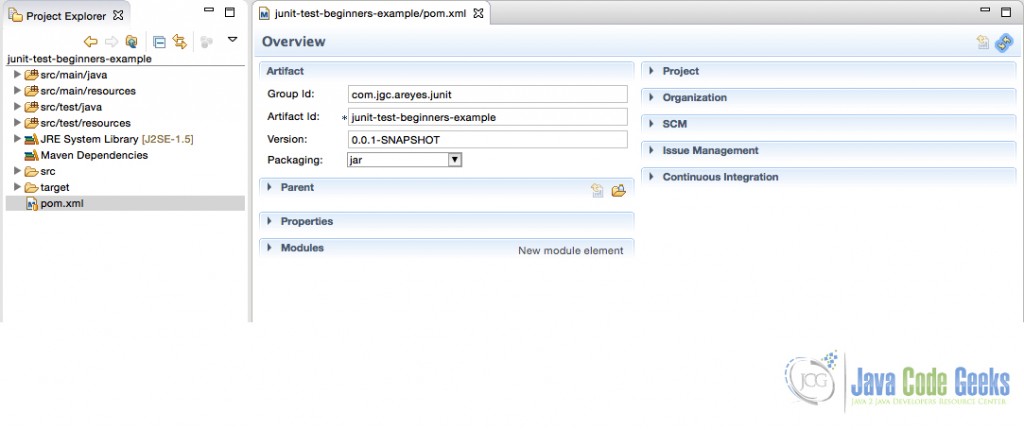
[](http://examples.javacodegeeks.com/wp-content/uploads/2015/10/new_project_maven.jpg)

Figure 2.0 New Maven Project .xml

Make sure you include the Junit Library to your dependency list.

*pom.xml*

|  |  |  |
| --- | --- | --- |
| 01 | <project xmlns="<http://maven.apache.org/POM/4.0.0>"xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> <http://maven.apache.org/xsd/maven-4.0.0.xsd>"> | |
| 02 | | <modelVersion>4.0.0</modelVersion> |

|  |  |
| --- | --- |
| 03 | <groupId>com.jgc.areyes.junit</groupId> |
| 04 | <artifactId>junit-test-beginners-example</artifactId> |

|  |  |
| --- | --- |
| 05 | <version>0.0.1-SNAPSHOT</version> |
| 06 |  |

|  |  |
| --- | --- |
| 07 |  |
| 08 | <dependencies> |

|  |  |
| --- | --- |
| 09 | <dependency> |
| 10 | <groupId>junit</groupId> |

|  |  |
| --- | --- |
| 11 | <artifactId>junit</artifactId> |
| 12 | <version>4.12</version> |

|  |  |
| --- | --- |
| 13 | <scope>test</scope> |
| 14 | </dependency> |

|  |  |
| --- | --- |
| 15 |  |
| 16 | </dependencies> |

|  |  |
| --- | --- |
| 17 |  |
| 18 |  |

|  |  |
| --- | --- |
| 19 | </project> |

3.2 Create the Service Class

Majority of the developers I know starts first with creating the implementation rather than the JUnit Test case, it isn’t a bad practice at all but wouldn’t it be more concrete if we create the JUnit Test cases first based on the design then create the implementation to pass all JUnit Test case? This is the case in the TDD, one of the most successful schemes of actual software development.

Assuming we are creating a service to manage an account. We need to introduce the following service methods:

* Create a new Account
* Update an Account
* Remove an Account
* List All Account Transactions

We have an OOP design that will handle this service and so we introduce the following classes (Class Diagram).

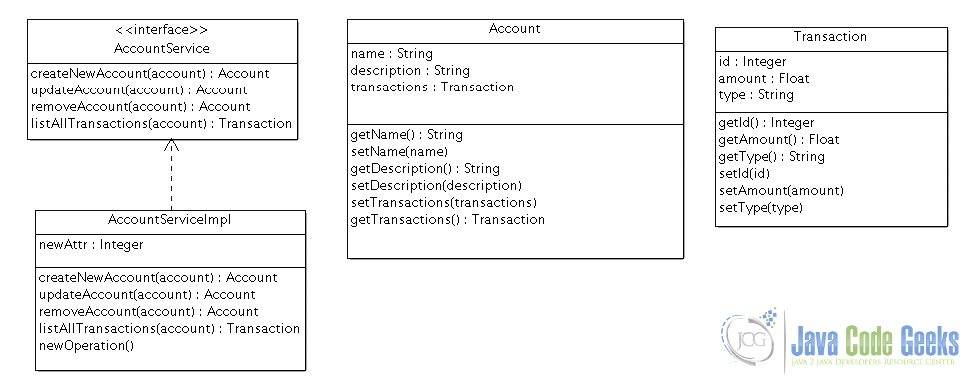
[](http://examples.javacodegeeks.com/wp-content/uploads/2015/10/cdiag1.jpg)

Figure 3.0 Class Diagram

Here is the actual class that doesn’t have any implementation yet. We will create the implementation after creating the test cases for this class.

*AccountServiceImpl.java*

|  |  |
| --- | --- |
| 01 | package com.areyes1.jgc.svc; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | import java.util.List; |
| 04 |  |

|  |  |
| --- | --- |
| 05 | import com.areyes1.jgc.intf.AccountService; |
| 06 | import com.areyes1.jgc.obj.Account; |

|  |  |
| --- | --- |
| 07 | import com.areyes1.jgc.obj.Transaction; |
| 08 |  |

|  |  |
| --- | --- |
| 09 | public class AccountServiceImpl implements AccountService { |
| 10 | public Account createNewAccount(Account account) { |

|  |  |
| --- | --- |
| 11 | // TODO Auto-generated method stub |
| 12 | return null; |

|  |  |
| --- | --- |
| 13 | } |
| 14 |  |

|  |  |
| --- | --- |
| 15 | public Account updateAccount(Account account) { |
| 16 | // TODO Auto-generated method stub |

|  |  |
| --- | --- |
| 17 | return null; |
| 18 | } |

|  |  |
| --- | --- |
| 19 |  |
| 20 | public Account removeAccount(Account account) { |

|  |  |
| --- | --- |
| 21 | // TODO Auto-generated method stub |
| 22 | return null; |

|  |  |
| --- | --- |
| 23 | } |
| 24 |  |

|  |  |
| --- | --- |
| 25 | public List listAllTransactions(Account account) { |
| 26 | // TODO Auto-generated method stub |

|  |  |
| --- | --- |
| 27 | return null; |
| 28 | } |

|  |  |
| --- | --- |
| 29 | } |

3.3 Create JUnit Test Cases

Now that we have the service placeholder, let’s create the Junit Test case for the AccountServiceImpl class. The Test cases will be the basis of your class functional aspect so you as a developer should write a solid and good test case (and not just fake it to pass).

When you create a test case, it will initially look like this:

*AccountServiceImplTest.java*

|  |  |
| --- | --- |
| 01 | /\*\* |
| 02 | \* |

|  |  |
| --- | --- |
| 03 | \*/ |
| 04 | package com.areyes1.jgc.svc; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | import static org.junit.Assert.\*; |

|  |  |
| --- | --- |
| 07 |  |
| 08 | import org.junit.After; |

|  |  |
| --- | --- |
| 09 | import org.junit.Before; |
| 10 | import org.junit.Test; |

|  |  |
| --- | --- |
| 11 |  |
| 12 | /\*\* |

|  |  |
| --- | --- |
| 13 | \* @author alvinreyes |
| 14 | \* |

|  |  |
| --- | --- |
| 15 | \*/ |
| 16 | public class AccountServiceImplTest { |

|  |  |
| --- | --- |
| 17 |  |
| 18 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 19 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#createNewAccount(com.areyes1.jgc.obj.Account)}. | |
| 20 | | \*/ |

|  |  |
| --- | --- |
| 21 | @Test |
| 22 | public void testCreateNewAccount() { |

|  |  |
| --- | --- |
| 23 | fail("Not yet implemented"); |
| 24 | } |

|  |  |
| --- | --- |
| 25 |  |
| 26 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 27 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#updateAccount(com.areyes1.jgc.obj.Account)}. | |
| 28 | | \*/ |

|  |  |
| --- | --- |
| 29 | @Test |
| 30 | public void testUpdateAccount() { |

|  |  |
| --- | --- |
| 31 | fail("Not yet implemented"); |
| 32 | } |

|  |  |
| --- | --- |
| 33 |  |
| 34 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 35 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#removeAccount(com.areyes1.jgc.obj.Account)}. | |
| 36 | | \*/ |

|  |  |
| --- | --- |
| 37 | @Test |
| 38 | public void testRemoveAccount() { |

|  |  |
| --- | --- |
| 39 | fail("Not yet implemented"); |
| 40 | } |

|  |  |
| --- | --- |
| 41 |  |
| 42 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 43 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#listAllTransactions(com.areyes1.jgc.obj.Account)}. | |
| 44 | | \*/ |

|  |  |
| --- | --- |
| 45 | @Test |
| 46 | public void testListAllTransactions() { |

|  |  |
| --- | --- |
| 47 | fail("Not yet implemented"); |
| 48 | } |

|  |  |
| --- | --- |
| 49 |  |
| 50 | } |

As it can be seen above, it will fail the test case if it’s not yet implemented. From here on now, it’s a matter of discipline from the developer to create concrete and solid test cases.

Observing the JUnit Test cases.

* Using @Test to define a test method
* Put them all on the Test package (src/test/main/)
* Class always has a suffix of Test (AccountServiceImplTest)
* Methods always starts with “test”.

Here’s is the modified version of the JUnit Test cases. This will now be the basis of our implementation code.

*AccountServiceImplTest.java*

|  |  |
| --- | --- |
| 01 | /\*\* |
| 02 | \* |

|  |  |
| --- | --- |
| 03 | \*/ |
| 04 | package com.areyes1.jgc.svc; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | import static org.junit.Assert.\*; |

|  |  |
| --- | --- |
| 07 |  |
| 08 | import org.junit.After; |

|  |  |
| --- | --- |
| 09 | import org.junit.Before; |
| 10 | import org.junit.Test; |

|  |  |
| --- | --- |
| 11 | import static org.hamcrest.CoreMatchers.\*; |
| 12 | import com.areyes1.jgc.obj.Account; |

|  |  |
| --- | --- |
| 13 | /\*\* |
| 14 | \* @author alvinreyes |

|  |  |
| --- | --- |
| 15 | \* |
| 16 | \*/ |

|  |  |
| --- | --- |
| 17 | public class AccountServiceImplTest { |
| 18 |  |

|  |  |
| --- | --- |
| 19 | AccountServiceImpl accountService = new AccountServiceImpl(); |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | | /\*\* |
| 22 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#createNewAccount(com.areyes1.jgc.obj.Account)}. | |

|  |  |
| --- | --- |
| 23 | \*/ |
| 24 | @Test |

|  |  |
| --- | --- |
| 25 | public void testCreateNewAccount() { |
| 26 | Account newAccount = new Account(); |

|  |  |
| --- | --- |
| 27 | newAccount.setName("Alvin Reyes"); |
| 28 | newAccount.setDescription("This is the description"); |

|  |  |
| --- | --- |
| 29 |  |
| 30 | Account newAcccountInserted = accountService.createNewAccount(newAccount); |

|  |  |
| --- | --- |
| 31 |  |
| 32 | //  Check if the account has the same composition. |

|  |  |
| --- | --- |
| 33 | assertThat(newAccount, isA(Account.class)); |
| 34 | assertEquals(newAccount.getName(), newAcccountInserted.getName()); |

|  |  |
| --- | --- |
| 35 |  |
| 36 |  |

|  |  |
| --- | --- |
| 37 | } |
| 38 |  |

|  |  |  |
| --- | --- | --- |
| 39 | | /\*\* |
| 40 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#updateAccount(com.areyes1.jgc.obj.Account)}. | |

|  |  |
| --- | --- |
| 41 | \*/ |
| 42 | @Test |

|  |  |
| --- | --- |
| 43 | public void testUpdateAccount() { |
| 44 |  |

|  |  |
| --- | --- |
| 45 |  |
| 46 | //  The old account (assumed that this came from a database or mock) |

|  |  |
| --- | --- |
| 47 | Account oldAccount = new Account(); |
| 48 | oldAccount.setName("Alvin Reyes"); |

|  |  |
| --- | --- |
| 49 | oldAccount.setDescription("This is the description"); |
| 50 |  |

|  |  |
| --- | --- |
| 51 | String name = oldAccount.getName(); |
| 52 | //  Check if the account is still the same. it is expected to be different since we updated it. |

|  |  |
| --- | --- |
| 53 | Account expectedAccountObj =  new Account(); |
| 54 | expectedAccountObj = accountService.updateAccount(oldAccount); |

|  |  |
| --- | --- |
| 55 | assertThat(expectedAccountObj, isA(Account.class)); |
| 56 | assertNotEquals(name, expectedAccountObj.getName()); |

|  |  |
| --- | --- |
| 57 |  |
| 58 | } |

|  |  |
| --- | --- |
| 59 |  |
| 60 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 61 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#removeAccount(com.areyes1.jgc.obj.Account)}. | |
| 62 | | \*/ |

|  |  |
| --- | --- |
| 63 | @Test |
| 64 | public void testRemoveAccount() { |

|  |  |
| --- | --- |
| 65 |  |
| 66 | //  Set up the account to be removed. |

|  |  |
| --- | --- |
| 67 | Account toBeRemovedAccount = new Account(); |
| 68 | toBeRemovedAccount.setName("Alvin Reyes"); |

|  |  |
| --- | --- |
| 69 | toBeRemovedAccount.setDescription("This is the description"); |
| 70 |  |

|  |  |
| --- | --- |
| 71 | //  Removed the account. |
| 72 | assertTrue(accountService.removeAccount(toBeRemovedAccount)); |

|  |  |
| --- | --- |
| 73 |  |
| 74 | } |

|  |  |
| --- | --- |
| 75 |  |
| 76 | /\*\* |

|  |  |  |
| --- | --- | --- |
| 77 | \* Test method for {@link com.areyes1.jgc.svc.AccountServiceImpl#listAllTransactions(com.areyes1.jgc.obj.Account)}. | |
| 78 | | \*/ |

|  |  |
| --- | --- |
| 79 | @Test |
| 80 | public void testListAllTransactions() { |

|  |  |
| --- | --- |
| 81 |  |
| 82 | //   Dummy Transactions (can be mocked via mockito) |

|  |  |
| --- | --- |
| 83 | Account account = new Account(); |
| 84 | account.setName("Alvin Reyes"); |

|  |  |
| --- | --- |
| 85 |  |
| 86 | //  Service gets all transaction |

|  |  |
| --- | --- |
| 87 | accountService.listAllTransactions(account); |
| 88 |  |

|  |  |
| --- | --- |
| 89 | //  Check if there are transactions. |
| 90 | assertTrue(accountService.listAllTransactions(account).size() > 1); |

|  |  |
| --- | --- |
| 91 |  |
| 92 |  |

|  |  |
| --- | --- |
| 93 | } |
| 94 |  |

|  |  |
| --- | --- |
| 95 | } |

Running the test case will show the following result.

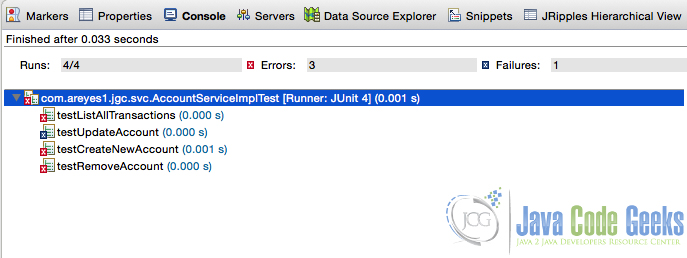
[](http://examples.javacodegeeks.com/wp-content/uploads/2015/10/failed_test_case.jpg)

Figure 4.0 Failed Test cases

It failed cause we still have to code our implementation. Now in the implementation of our logic, our goal is make sure that these test cases succeeds!

3.4 Code the implementation

Now that the test cases are setup, we can now code our implementation. The test cases we created above will be the basis of how we will create the implementation. The goal is to pass the test cases!

|  |  |
| --- | --- |
| 01 | package com.areyes1.jgc.svc; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | import java.util.ArrayList; |
| 04 | import java.util.List; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | import com.areyes1.jgc.intf.AccountService; |

|  |  |
| --- | --- |
| 07 | import com.areyes1.jgc.obj.Account; |
| 08 | import com.areyes1.jgc.obj.Transaction; |

|  |  |
| --- | --- |
| 09 |  |
| 10 | public class AccountServiceImpl implements AccountService { |

|  |  |
| --- | --- |
| 11 | public Account createNewAccount(Account account) { |
| 12 | //  Dummy Dao! Database insert here. |

|  |  |
| --- | --- |
| 13 | //  accountDao.insert(account); |
| 14 | //  Ultimately return the account with the modification. |

|  |  |
| --- | --- |
| 15 | return account; |
| 16 |  |

|  |  |
| --- | --- |
| 17 | } |
| 18 |  |

|  |  |
| --- | --- |
| 19 | public Account updateAccount(Account account) { |
| 20 | //  Dummy Dao! Database insert here. |

|  |  |
| --- | --- |
| 21 | //  accountDao.update(account); |
| 22 | //  Ultimately return the account with the modification. |

|  |  |
| --- | --- |
| 23 | account.setName("Alvin Reyes: New Name"); |
| 24 | return account; |

|  |  |
| --- | --- |
| 25 | } |
| 26 |  |

|  |  |
| --- | --- |
| 27 | public boolean removeAccount(Account account) { |
| 28 | //  Dummy Dao! Database insert here. |

|  |  |
| --- | --- |
| 29 | //  accountDao.delete(account); |
| 30 | //  Ultimately return the account with the modification. |

|  |  |
| --- | --- |
| 31 | //  if exception occurs, return false. |
| 32 | return true; |

|  |  |
| --- | --- |
| 33 | } |
| 34 |  |

|  |  |
| --- | --- |
| 35 | public List listAllTransactions(Account account) { |
| 36 | // accountDao.loadAllTransactions(account); |

|  |  |
| --- | --- |
| 37 | List listOfAllTransactions = new ArrayList(); |
| 38 | listOfAllTransactions.add(new Transaction()); |

|  |  |
| --- | --- |
| 39 | listOfAllTransactions.add(new Transaction()); |
| 40 | listOfAllTransactions.add(new Transaction()); |

|  |  |
| --- | --- |
| 41 | account.setTransactions(listOfAllTransactions); |
| 42 |  |

|  |  |
| --- | --- |
| 43 | return listOfAllTransactions; |
| 44 | } |

|  |  |
| --- | --- |
| 45 | } |

Running the test case will show the following result.

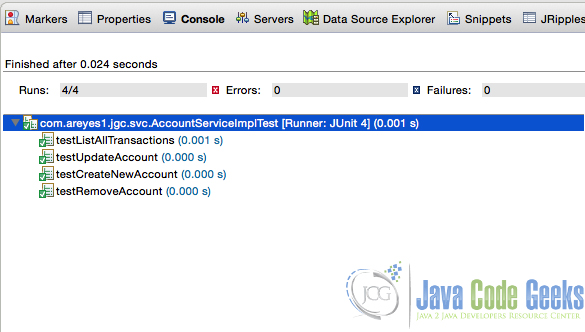
[](http://examples.javacodegeeks.com/wp-content/uploads/2015/10/passed_test_case.jpg)

Figure 5.0 Passed Test case

3.5 Run your maven

Run your maven to see results.

|  |  |
| --- | --- |
| 01 | ------------------------------------------------------- |
| 02 | T E S T S |

|  |  |
| --- | --- |
| 03 | ------------------------------------------------------- |
| 04 | Running com.areyes1.jgc.svc.AccountServiceImplTest |

|  |  |
| --- | --- |
| 05 | Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.051 sec |
| 06 |  |

|  |  |
| --- | --- |
| 07 | Results : |
| 08 |  |

|  |  |
| --- | --- |
| 09 | Tests run: 4, Failures: 0, Errors: 0, Skipped: 0 |
| 10 |  |

|  |  |
| --- | --- |
| 11 | [INFO] |
| 12 | [INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ junit-test-beginners-example --- |

|  |  |
| --- | --- |
| 13 | [INFO] Building jar: /Users/alvinreyes/EclipseProjects/Java/junit-test-beginners-example/target/junit-test-beginners-example-0.0.1-SNAPSHOT.jar |
| 14 | [INFO] |

|  |  |
| --- | --- |
| 15 | [INFO] --- maven-install-plugin:2.4:install (default-install) @ junit-test-beginners-example --- |
| 16 | [INFO] Installing /Users/alvinreyes/EclipseProjects/Java/junit-test-beginners-example/target/junit-test-beginners-example-0.0.1-SNAPSHOT.jar to /Users/alvinreyes/.m2/repository/com/jgc/areyes/junit/junit-test-beginners-example/0.0.1-SNAPSHOT/junit-test-beginners-example-0.0.1-SNAPSHOT.jar |

|  |  |
| --- | --- |
| 17 | [INFO] Installing /Users/alvinreyes/EclipseProjects/Java/junit-test-beginners-example/pom.xml to /Users/alvinreyes/.m2/repository/com/jgc/areyes/junit/junit-test-beginners-example/0.0.1-SNAPSHOT/junit-test-beginners-example-0.0.1-SNAPSHOT.pom |
| 18 | [INFO] |

|  |  |
| --- | --- |
| 19 | [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ junit-test-beginners-example --- |
| 20 | [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent! |

|  |  |
| --- | --- |
| 21 | [INFO] Copying 0 resource |
| 22 | [INFO] |

|  |  |
| --- | --- |
| 23 | [INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ junit-test-beginners-example --- |
| 24 | [INFO] Nothing to compile - all classes are up to date |

|  |  |
| --- | --- |
| 25 | [INFO] |
| 26 | [INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit-test-beginners-example --- |

|  |  |
| --- | --- |
| 27 | [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent! |
| 28 | [INFO] Copying 0 resource |

|  |  |
| --- | --- |
| 29 | [INFO] |
| 30 | [INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @ junit-test-beginners-example --- |

|  |  |
| --- | --- |
| 31 | [INFO] Nothing to compile - all classes are up to date |
| 32 | [INFO] |

|  |  |
| --- | --- |
| 33 | [INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit-test-beginners-example --- |
| 34 | [INFO] Skipping execution of surefire because it has already been run for thisconfiguration |

|  |  |
| --- | --- |
| 35 | [INFO] ------------------------------------------------------------------------ |
| 36 | [INFO] BUILD SUCCESS |

|  |  |
| --- | --- |
| 37 | [INFO] ------------------------------------------------------------------------ |
| 38 | [INFO] Total time: 9.456 s |

|  |  |
| --- | --- |
| 39 | [INFO] Finished at: 2015-10-12T16:28:01-05:00 |
| 40 | [INFO] Final Memory: 11M/28M |

|  |
| --- |
| 41 |