Test di Unità con JUnit

Table of Contents

JUnit 4 Primo Esempio

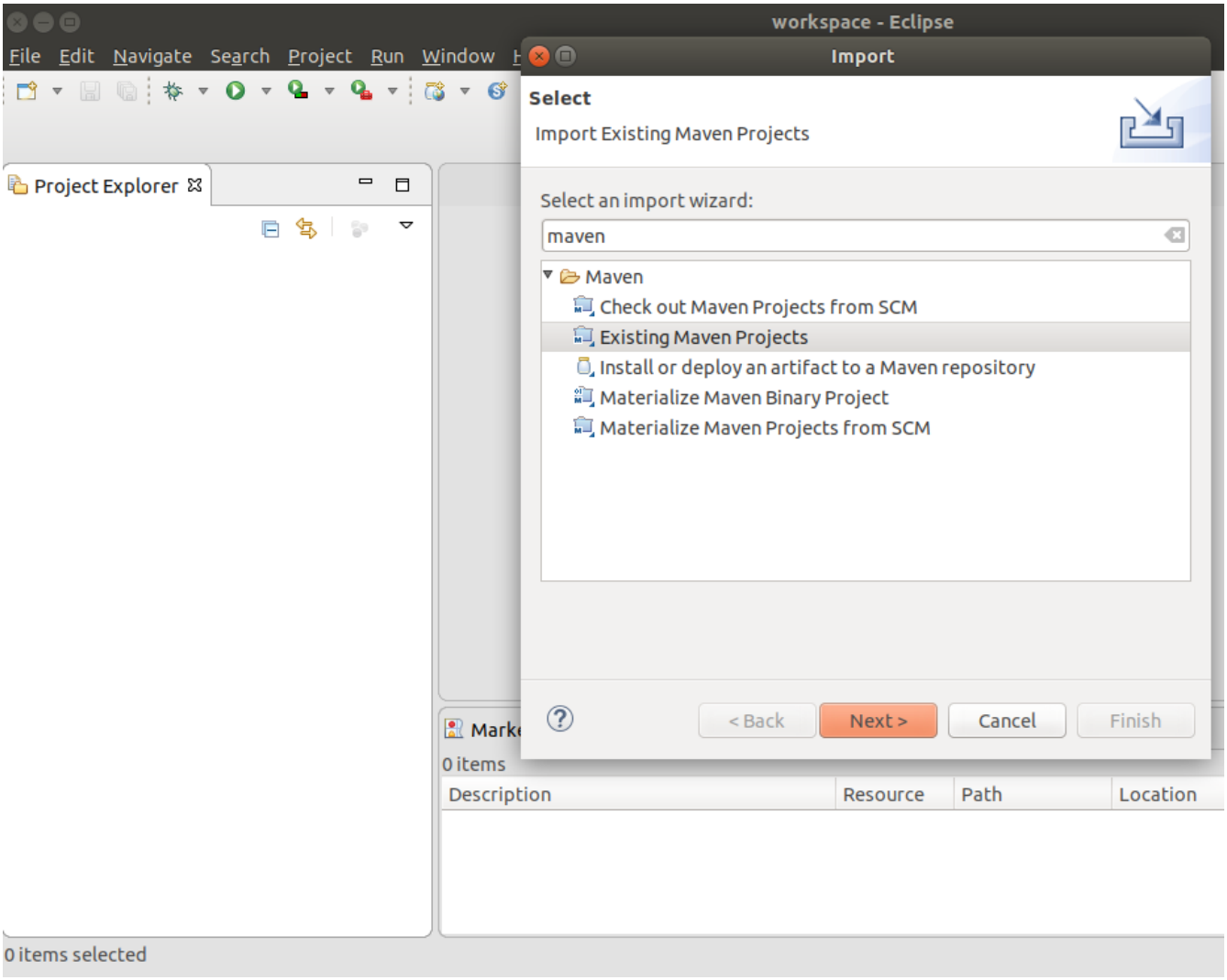
> **NOTE** l'esempio è tratto dal tutorial Getting Started di Junit 4.

Creazione del progetto

- Creare un progetto Java con maven utilizzando l'archetipo `maven-archetype-quickstart` specificando il seguente GAV:
  - GroupId = it.unipd.tos
  - ArtifactId = lab4
  - Version = 1.0-SNAPSHOT

```
mvn archetype:generate -DgroupId=it.unipd.tos -DartifactId=lab4
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

- Aprire Eclipse ed importare il progetto utilizzando la voce `import > import > Maven > Existing Maven Projects`



- Selezionare la cartella contenente il progetto appena creato.
- Il progetto verrà visualizzato in Eclipse

Modifica del pom.xml

- Aggiungere al pom.xml del progetto le seguenti proprietà

```
<project>
  [...]
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  [...]
</project>
```

- modificare la dipendenza di JUnit specificando la versione 4.12

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

- Eseguire il *Build Lifecycle* cliccando con il tasto destro del mouse nel progetto e selezionare la voce di menu `Run as > Maven Build` e specificare nel goal `clean package`.
- Verificare che nella consolle il progetto compili correttamente, esegua i test di unità e crei l'artefatto.

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building lab4 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ lab4 ---
[INFO] Deleting /home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ lab4 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ lab4 ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
[INFO] Compiling 1 source file to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ lab4 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ lab4 ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
[INFO] Compiling 1 source file to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ lab4 ---
```

```
[INFO] Surefire report directory:
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running it.unipd.tos.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ lab4 ---
[INFO] Building jar: /home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/lab4-1.0-
SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.711 s
[INFO] Finished at: 2018-11-08T21:53:54+01:00
[INFO] Final Memory: 18M/293M
[INFO] ------------------------------------------------------------------------
```

Come visualizzato nell'output, Maven ha eseguito un test di unità:

```
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ lab4 ---
[INFO] Surefire report directory:
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running it.unipd.tos.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

Il test di unità `it.unipd.tos.AppTest` è stato creato dall'artchetipo `maven-archetype-quickstart` ed è presente in:

```
/lab4/src/test/java/it/unipd/tos/AppTest.java
```

Com'è possibile notare, le classi di test (/lab4/src/test/java/) sono separate dalle classi di produzione (/lab4/src/main/java/) e non vengono inserite nel artefatto.

TIP	Aprire l'artefatto generato precedentemente (/lab4/target/lab4-1.0-SNAPSHOT.jar) con programma in grado di gestire gli archivi di tipo zip

Creazione della classe di business

- Posizionarsi nella cartella dei sorgenti `src/main/java` e selezionare la voce di menu `New > Class'
- Inserire nel parametro Name il valore `Calculator`
- Inserire nel file il seguente sorgente (tratto da Getting Started di Junit 4)

```
package it.unipd.tos;

public class Calculator {
    public int evaluate(String expression) {
        int sum = 0;
        for (String summand : expression.split("\\+"))
            sum += Integer.valueOf(summand);
        return sum;
    }
}
```

La classe `Calculator` rappresenta la nostra unità. Espone il metodo `evaluate` che, data una stringa che rappresenta un'addizione da calcolare ne restituisce il risultato.

- Rieseguire nuovamente il build lifecycle e verificare che la classe compili.

Creazione del primo test di unità

- Posizionarsi nel file `src/main/java/it/unipd/tos/Calculator.java` e selezionare la voce di menu `New > Other > JUnit > Junit Test Case'
- Modificare il parametro **Source folder** da `lab4/src/main/java` a `lab4/src/test/java`
- Cliccando il bottone *Finish*. Eclipse genera la seguente classe:

```
package it.unipd.tos;

import static org.junit.Assert.*;

import org.junit.Test;

public class CalculatorTest {

    @Test
    public void test() {
        fail("Not yet implemented");
    }

}
```

- Rieseguire nuovamente il build lifecycle e verificare che il processo di build fallisca

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building lab4 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ lab4 ---
[INFO] Deleting /home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ lab4 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ lab4 ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
[INFO] Compiling 2 source files to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ lab4 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ lab4 ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
```

```
[INFO] Compiling 2 source files to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ lab4 ---
[INFO] Surefire report directory:
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running it.unipd.tos.CalculatorTest
Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.08 sec <<< FAILURE!
test(it.unipd.tos.CalculatorTest)  Time elapsed: 0.008 sec  <<< FAILURE!
java.lang.AssertionError: Not yet implemented
    at org.junit.Assert.fail(Assert.java:88)
    at it.unipd.tos.CalculatorTest.test(CalculatorTest.java:11)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at
org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
    at
org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at
org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
    at
org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)
    at
org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)
    at
org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)
    at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
    at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)
    at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)
    at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)
    at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)
    at org.junit.runners.ParentRunner.run(ParentRunner.java:363)
    at
org.apache.maven.surefire.junit4.JUnit4Provider.execute(JUnit4Provider.java:252)
    at
org.apache.maven.surefire.junit4.JUnit4Provider.executeTestSet(JUnit4Provider.java:141
)
    at org.apache.maven.surefire.junit4.JUnit4Provider.invoke(JUnit4Provider.java:112)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
```

```
    at
org.apache.maven.surefire.util.ReflectionUtils.invokeMethodWithArray(ReflectionUtils.j
ava:189)
    at
org.apache.maven.surefire.booter.ProviderFactory$ProviderProxy.invoke(ProviderFactory.
java:165)
    at
org.apache.maven.surefire.booter.ProviderFactory.invokeProvider(ProviderFactory.java:8
5)
    at
org.apache.maven.surefire.booter.ForkedBooter.runSuitesInProcess(ForkedBooter.java:115
)
    at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:75)

Running it.unipd.tos.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 sec

Results :

Failed tests:   test(it.unipd.tos.CalculatorTest): Not yet implemented

Tests run: 2, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.330 s
[INFO] Finished at: 2018-11-08T22:09:26+01:00
[INFO] Final Memory: 15M/208M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-
plugin:2.12.4:test (default-test) on project lab4: There are test failures.
[ERROR]
[ERROR] Please refer to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/lab4/target/surefire-reports for the
individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the
following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Maven è in grado di interpretare i risultati di JUnit. Il test CalcolatorTest fallisce:

```
Failed tests:   test(it.unipd.tos.CalculatorTest): Not yet implemented
```

Lo stesso risultato si può ottenere da Eclipse selezionando la classe `src/test/java/it/unipd/tos/CalculatorTest.java` ed eseguendo la voce di menu `Run As > JUnit Test`.

NOTE	Junit utilizza un modo standard per eseguire ed esprimere se il test è stato superato, se è fallito o se sono stati prodotti degli errori. I risultati vengono interpretati correttamente sia da Maven che da Eclipse.
NOTE	E' possibile eseguire JUnit da linea di comando. per maggiori informazioni vedi https://github.com/junit-team/junit4/wiki/Getting-started#run-the-test

Creazione di un test

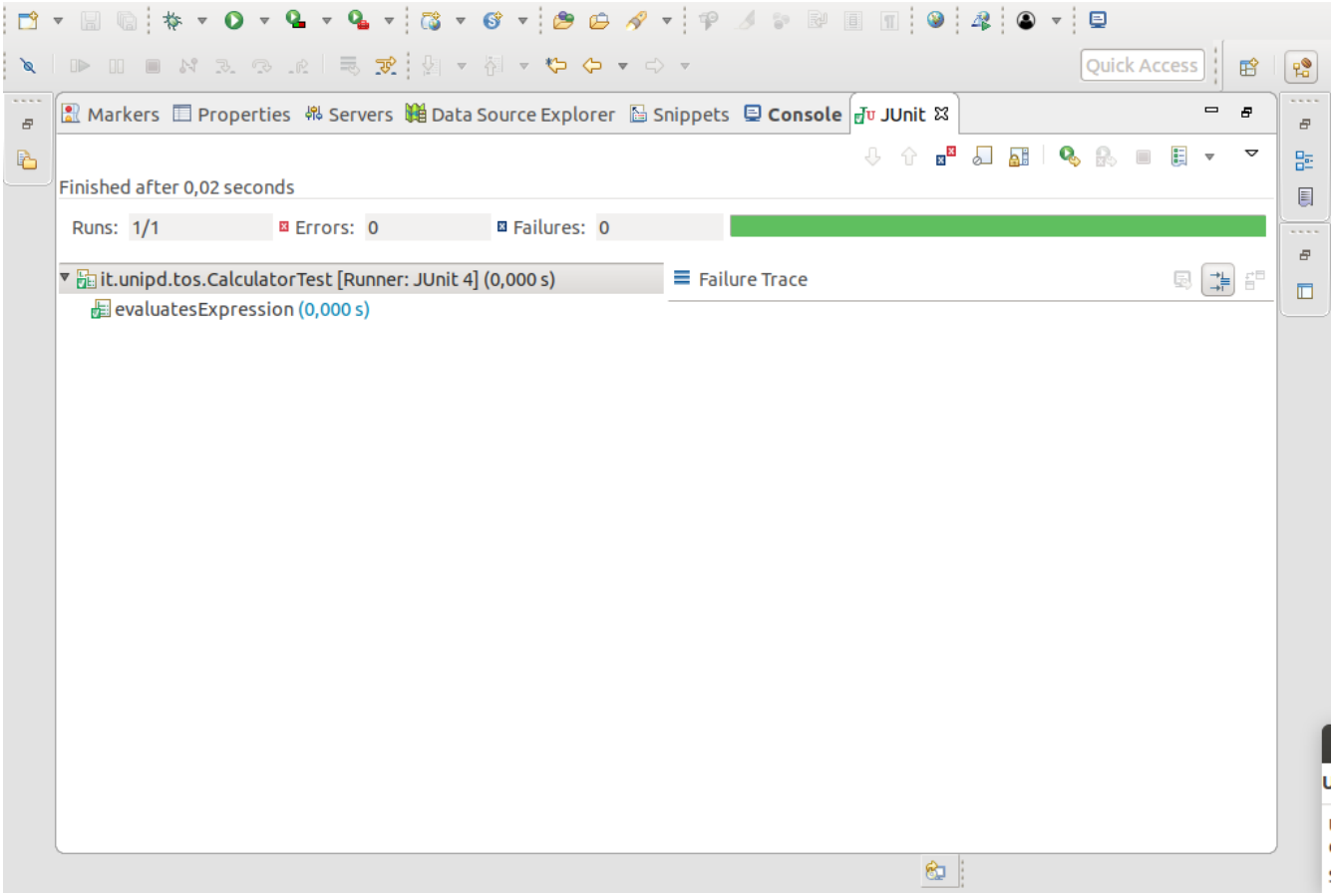
- Modificare la classe `src/test/java/it/unipd/tos/CalculatorTest.java` nel seguente modo:

```
package it.unipd.tos;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {
    @Test
    public void evaluatesExpression() {
        Calculator calculator = new Calculator();
        int sum = calculator.evaluate("1+2+3");
        assertEquals(6, sum);
    }
}
```

La classe di test è composta dai seguenti elementi:

- Riga 8: Annotazione Test: Come descritto qui: *The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.*
- Riga 12: il metodo **assertEquals** permette di verificare se il primo parametro (expected) è uguale al valore calcolato dal metodo *evaluate* (actual)

Eseguendo nuovamente il test sarà possibile vedere che il test case `evaluatesExpression` è stato eseguito con successo.



TIP	modificare la classe Calculator in modo da far fallire il test.
TIP	modificare la classe CalculatorTest inserendo i casi di test presenti nella slide 18 della lezione *8-Test-di-unita.pdf*

Secondo Esempio

NOTE	L'esempio è tratto dal tutorial In 28 Minutes - JUnit Tutorial

Importare il progetto da GitHub

- Clonare il progetto GitHub https://github.com/in28minutes/JUnitIn28Minutes.git

```
git clone https://github.com/in28minutes/JUnitIn28Minutes.git
```

- Importare il progetto Maven in Eclipse
  - Selezionare la voce di menu `File > Import  > Maven > Existing Maven Projects` e selezionare il path del progetto appena importato.
- Eseguire i goal `clean package` con maven

```
mvn clean package
```

- Il risultato ottenuto sarà simile a:

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building maven-example-1 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ junit-example ---
[INFO] Deleting /home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ junit-example
---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.2:compile (default-compile) @ junit-example ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
[INFO] Compiling 16 source files to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/target/classes

...

[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit-
example ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.2:testCompile (default-testCompile) @ junit-example
---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
platform dependent!
[INFO] Compiling 8 source files to
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/target/test-classes

...

[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit-example ---
[INFO] Surefire report directory:
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/target/surefire-reports
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.in28minutes.junit.helper.StringHelperParameterizedTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.083 sec
Running com.in28minutes.junit.helper.ArraysTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec
Running com.in28minutes.junit.helper.QuickBeforeAfterTest
Before Class
Before Test
test1 executed
After test
Before Test
test2 executed
After test
After Class
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.014 sec
Running com.in28minutes.junit.helper.StringHelperTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 sec
Running com.in28minutes.junit.helper.ArraysCompareTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec
Running com.clarity.business.ClientBOTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 sec

Results :

Tests run: 15, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ junit-example ---
[INFO] Building jar:
/home/bertazzo/Progetti/2018/UNIV/TOS/lab4/JUnitIn28Minutes/target/junit-example-1.0-
SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.503 s
[INFO] Finished at: 2018-11-08T22:59:16+01:00
[INFO] Final Memory: 19M/301M
[INFO] ------------------------------------------------------------------------
```

E' possibile notare che sono stati eseguiti 15 test case con successo.

Eseguire gli stessi test cases con Eclipse:

- selezionare il progetto ed eseguire la voce di menu 'Run As > JUnit Test'
- verranno eseguiti 23 test case con successo

NOTE	Maven per default esegue solo le classi di test con un nome che deve soddisfare un determinato pattern vedi qui

JUnit Annotation

Configurare l'ambiente di esecuzione del test

Come descritto nella documentazione di JUnit, tabella *Annotation Types Summary* sono disponibili seguenti annotazioni:

Annotation	Description
@Test public void method()	The Test annotation indicates that the public void method to which it is attached can be run as a test case.
@Before public void method()	The Before annotation indicates that this method must be executed before each test in the class, so as to execute some preconditions necessary for the test.
@BeforeClass public static void method()	The BeforeClass annotation indicates that the static method to which is attached must be executed once and before all tests in the class. That happens when the test methods share computationally expensive setup (e.g. connect to database).
@After public void method()	The After annotation indicates that this method gets executed after execution of each test (e.g. reset some variables after execution of every test, delete temporary variables etc)
@AfterClass public static void method()	The AfterClass annotation can be used when a method needs to be executed after executing all the tests in a JUnit Test Case class so as to clean-up the expensive set-up (e.g disconnect from a database). Attention: The method attached with this annotation (similar to BeforeClass) must be defined as static.
@Ignore public static void method()	The Ignore annotation can be used when you want temporarily disable the execution of a specific test. Every method that is annotated with @Ignore won't be executed.

Un esempio dell'utilizzo di queste annotazioni è presente nella classe di test `/junit-example/src/test/java/com/in28minutes/junit/helper/QuickBeforeAfterTest.java`

Eseguendo i test case presenti nella classe è possibile vedere che l'output generato è il seguente:

```
Before Class
Before Test
test1 executed
After test
Before Test
test2 executed
After test
After Class
```

Modificare la classe andando ad aggiungere l'annotazione @ignore al test 1

```
...
@Ignore
@Test
public void test1() {
  System.out.println("test1 executed");
}
...
```

In questo modo il test case *test1()* verrà ignorato.

Eseguire nuovamente la classe di test. Il risultato ottenuto sarà il seguente:

```
Before Class
Before Test
test2 executed
After test
After Class
```

Assertion

JUnit mette a disposizioni dei metodi, chiamati assertion, che permettono di verificare se il risultato atteso è uguale al risultato ritornato dal metodo sotto test.

Tutte le asserzioni messe a disposizione dal framework JUnit sono presente qui.

NOTE	In JUnit 4 sono presenti dei nuovi meccanismi per effettuare le verifiche in modo più elegante. Per maggiori informazioni

Esempio di Test di unità su una classe esterna

Supponiamo di voler verificare il comportamento del metodo `sort` della classe `java.util.Arrays` (per capirne meglio il comportamento).

Aprire la classe di test `/junit-example/src/test/java/com/in28minutes/junit/helper/ArraysCompareTest.java` e vedere che:

- La classe contiene 3 test:
  - **testArraySort_RandomArray()**: che verifica l'ordinamento di un'array di *int*
  - **testArraySort_NullArray()**: che verifica che l'ordinamento di un array nullo produce una NullPointerException.exception (vedi attibuto expected dell'annotazione *@Test*)
  - **testSort_Performance()**: che verifica che il tempo di esecuzione del metodo sort di un array di 3 *int* eseguito 1000000 è inferiore a 100 millisecondi (vedi attibuto timeout dell'annotazione *@Test*)

Esempio di Test di unità su una classe del progetto

Supponiamo di voler verificare il comportamento di una classe del nostro progetto.

- Aprire la classe `/junit-example/src/main/java/com/in28minutes/junit/helper/StringHelper.java`
- La classe contiene i metodi:
  - **truncateAInFirst2Positions**: Elimina tutte le A presenti nei primi 2 caratteri di una stringa
  - **areFirstAndLastTwoCharactersTheSame**: Verifica se i primi 2 e gli ultimi 2 caratteri di una stringa sono uguali

Andiamo ad aprire la classe di test `/junit-example/src/test/java/com/in28minutes/junit/helper/StringHelperTest.java`

NOTE	È consigliato chiamare le classi di test come le classi di produzione aggiungendo il suffisso Test ed inserire la classe nello stesso package della classe di produzione ma nella cartella dei sorgenti di test. In questo modo nei test case sarà possibile accedere sia ai metodi pubblici che prottetti, e sarà più semplice seprarere i test dal codice di produzione.

- Riga 12: la classe di test ha un'attributo di tipo StringHelper che viene inizializzato nel metodo *before()* che viene eseguito prima di ogni test
- I metodi di test sono stati creati rispettando la naming convention introditta da Roy Osherove

Esempio di Test di unità parametrici

Come visto a lezione una delle caratteristiche di buoni test di unità è che siano Esaustivi.

JUnit mette a disposizione un modo per specificare i valori di input e di output utilizzati nel caso di test in un unico metodo.

Un esempio di utilizzo di questa funzionalità è presente nella classe di test `/junit-example/src/test/java/com/in28minutes/junit/helper/StringHelperParameterizedTest.java`.

- Riga 14 @RunWith(Parameterized.class): Questa annotazione permette di eseguire i test presenti nella classe tramite la classe Parametrized: *The custom runner Parameterized implements parameterized tests. When running a parameterized test class, instances are created for the cross-product of the test methods and the test data elements.*
- Riga 24: Costruttore che prende in input 2 parametri specificati dal metodo annotato con

@Parameters

- Riga 29: Metodo che permette di specificare i valori da passare al costruttore
- Riga 38: Test case che usa gli attributi della classe come valori di input e di output del test

Eseguendo la classe di test otteniamo 2 esecuzioni del metodo *testTruncateAInFirst2Positions* con i valori specificati nel metodo annotato come @Parameters.

Il Metodo che permette di specificare i valori da passare al costruttore può essere implementato in modo da leggere i valori da una fonte esterna (p.es. un foglio di calcolo). Per maggiori informazioni vedi qui.

Esempio Test di unità su un progetto complesso

Gli esempi visti in precedenza verificano una singola classe semplice, che non dipende da altri componenti.

Andiamo ad esaminare un esempio più complesso presente nel progetto junit-example.

Codice di produzione

L'esempio permette di calcolare la somma del prezzo di più prodotti con la stessa valuta.

Il progetto è composto dai seguenti package:

- com.in28minutes.junit.business: contiene il servizio che permette di calcolare la somma del prezzo di più prodotti **
- com.in28minutes.junit.model: contiene i Business Object (POJO) che permettono di modellare:
  - Ammount: rappresenta il valore e la valuta di un prodotto
  - Currency: rappresenta la valuta
  - Product: rappresenta il prodotto
  - ProductType: rappresenta il tipo di prodotto

Codice di test

Il test realizzato per verificare il comportamento della classe di business è presente nella classe `com.clarity.business.ClientBOTest`

I test presenti nella classe permettono di verificare:

- testClientProductSum(): La somma di una lista di due prodotti con la stessa valuta
- testClientProductSum1(): la somma di una lista di due prodotti con valute diverse
- testClientProductSum2(): la somma di una lista vuota

Come si può vedere la classe di test ha i seguenti problemi: * i nomi dei metodi di test sono poco significati * viene sempre ripetuto il codice richiesto per la creazione della lista di prodotti * viene sempre ripetuto il codice per verificare la somma calcolata

Poiché il codice di test deve essere scritto in modo professionale, è stato effettuato il refactoring della classe in `com.clarity.business.ClientBOTestRefactored`

Notare che: * i nomi dei metodi di test sono stati cambiati e sono più significativi * È stato aggiunto un metodo privato per la creazione della lista di prodotti a partire da un array di Amount * È stato aggiunto un metodo privato per effettuare il confronto di due Ammount

TIP	Provare ad eseguire i test da Eclipse tramite la voce di menu: `Coverage As > JUnit Test`.

Test Suite e Test Category

A volte è necessario raggruppare più test. JUnit mette a disposizione diverse modalità per raggruppare più test case in una *test suite* e attraverso le *category*.

Test Suite

Un esempio di test suite è presente nella classe `com.in28minutes.junit.suite.DummyTestSuite` che ha le seguenti caratteristiche:

- Riga 10: Annotazione @RunWith(Suite.class) permette di indicare a JUnit che la classe è una suite di test
- Riga 11: Annotazione @SuiteClasses({ArraysTest.class,StringHelperTest.class}) permette di indicare da quali classi recuperare i test case da eseguire.

Se viene eseguita la classe da Eclispe tramite la voce di menu `Run As > JUnit Test` verranno eseguiti i seguenti test case:

- L'unico test presente nella classe com.in28minutes.junit.helper.ArraysTest
- I quattro test presenti nella classe com.in28minutes.junit.StringHelperTest

NOTE	per maggiori informazioni vedi qui.

Categories

JUnit mette a disposizione un modo per raggruppare le classi di test e i metodi di test con delle etichette.

Questo permette ad esempio di classificare i test lenti, in modo da poterli eseguire con una frequenza diversa dai test veloci.

Per maggiori informazioni vedi qui

NOTE	il plugin Maven utilizzato per eseguire i test di unità può essere configurato in modo da eseguire dei test che appartengono ad una *category*.

TIP Provare a creare una suite che esegue solo i test che verificano le eccezioni