Test-LAB-2
JUnit avanzato

Prerequisiti

COMPUTER DI LABORATORIO o PERSONALE

Sistema di sviluppo Java installato.

JUnit installato, oppure accesso a Internet

Avere accesso all’account.

LA DURATA PREVISTA DI QUESTA ATTIVITÀ È DI 1-2 ORE DI 50 MINUTI.

1. Ripetizioni

Test Fixtures

Motivazione

I test automatici devono essere eseguiti molte volte. Per garantire che il processo di test sia ripetibile, è necessario che il test sia eseguito in uno stato noto chiamato fixture.

Test Fixtures

Ad esempio,

- Creazione / distruzioni di oggetti
- Apertura file
- Messaggi informativi
- …

Funzioni di supporto

Tutti i framework di test offrono sistemi che permettono di eseguire compiti ripetitivi in modo controllato e stabile.

Comandi per la gestione delle fixtures

`@BeforeAll` codice lanciato UNA VOLTA prima di questo test. Nelle versioni precedenti di Junit si chiamava `@BeforeClass`. Deve essere un metodo statico

`@AfterAll` codice lanciato UNA VOLTA al termine di questo test. Nelle versioni precedenti di Junit si chiamava `@AfterClass`. Deve essere un metodo statico

`@BeforeEach` metodo lanciato prima di OGNI test. Nelle versioni precedenti di Junit si chiamava `@Before`.

`@AfterEach` metodo lanciato prima di OGNI test. Nelle versioni precedenti di Junit si chiamava `@After`.

Demo

Create la classe “Spezzata” scaricando il codice da qui:https://pastebin.com/snDCai4y

Create in modo automatico lo scheletro di tutti i test.

Aggiungete questo test:

```
@BeforeAll
  public static void init(){
      System.out.println("Inizio test di Spezzata");
  }
```

Demo

Useremo @BeforeEach per preparare un oggetto fixture per i test. Aggiungete questo test:

```
   @BeforeEach
    void setup() {
        //ripulisce l'arraylist, per cui ogni test non ha
memoria
        spezzata=new Spezzata();// definita come instance var
        spezzata.aggiungi(new Punto(0,0));
        spezzata.aggiungi(new Punto(1,2));     }
```

SpezzataTest

Create in modo automatico lo scheletro di tutti i test.

Implementate il test `quantiPunti()`, usando almeno due assert. La fase di Arrange in pratica è svolta dal metodo `setup()`

Lanciare i test, non dovrebbe dare errori

aggiungi

Implementate il test aggiungi, aggiungendo un punto. Dato che l'attributo `punti` è privato, l'unico modo di controllare il risultato è tramite il metodo `toString()`.

Provate, potreste incontrare delle difficoltà (ma non errori). Usate "confronta" per vedere esattamente cosa si aspetta il test.

Possibile soluzione

```
    @Test
    void aggiungi() {
        spezzata.aggiungi(new Punto(3,3));
        assertEquals("[(0.0, 0.0), (1.0, 2.0),
(3.0, 3.0)]",spezzata.toString());

    }
```

Lunghezza e toString

Implementate i due test rimanenti.

Qui c’è un errore, che va rimosso.

Checkpoint #1:

- *Sapete usare i comandi di ripetizione*
- *Sapete testare le stringhe*

2.
Eccezioni, varianti, copertura

Quando l’errore è necessario

Testare le eccezioni

In certi casi è opportuno verificare che un codice generi una particolare eccezione.

E’ possibile scrivere un test con le modalità standard, catturando le specifiche eccezioni; tuttavia può diventare lungo da scrivere e manutenere.

Junit aiuta anche in questo campo.

NOTA: questo aspetto è stato fortemente modificato da JUnit 4 a JUnit 5

Testare le eccezioni

Scriviamo un test che sia “giusto” se genera un eccezione. Per esempio, testiamo una variabile non inizializzata.

La sintassi è un po’ diversa dagli Assert normali

```
@Test
void eccezione() {
    Spezzata spezzata2 = null;  //Arrange

Assertions.assertThrows(NullPointerException.
class, () -> {  //Assert
        double d = spezzata2.lunghezza();//Act
    });
}
```

Testare le eccezioni

Utilizza le lambda expressions, introdotte in Java 8

Il corpo della lambda è il codice sottoposto a test.

L'asserzione è vera se quel codice genera un'eccezione.

```
@Test
void eccezione() {
    Spezzata spezzata2 = null;   //Arrange

Assertions.assertThrows(NullPointerException.
class, () -> {   //Assert
                double d =
                spezzata2.lunghezza();//Act
                    });
}
```

Testare le eccezioni, variante più flessibile

```
@Test
void shouldThrowException() {
    Throwable exception =
assertThrows(NullPointerException.class, () -> {
        Spezzata spezzata2 = null;
        double d = spezzata2.lunghezza();
    });
    assertEquals("Cannot invoke \"Spezzata.lunghezza()\"
because \"spezzata2\" is null", exception.getMessage(), "Not
supported");
}
```

Annotazioni da ricordare

`@DisplayName("Stringa")`

Stampa la stringa prima del test

`@Disabled`

Disabilita un test, può essere messo prima di una classe o prima di un metodo.

Coverage

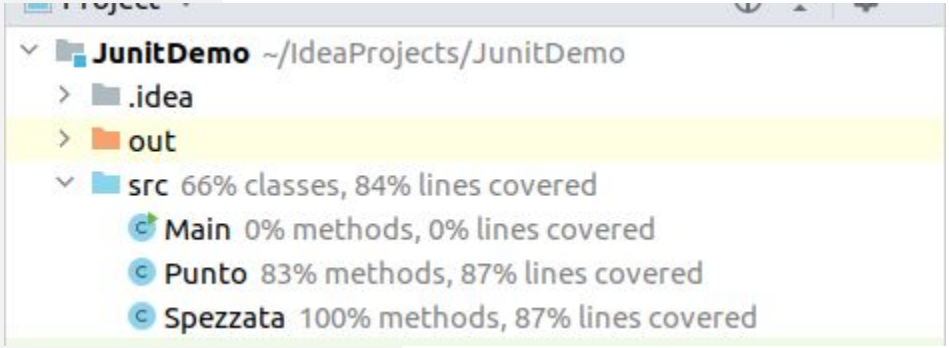
Il coverage indica la percentuale di codice che è sottoposto a test.

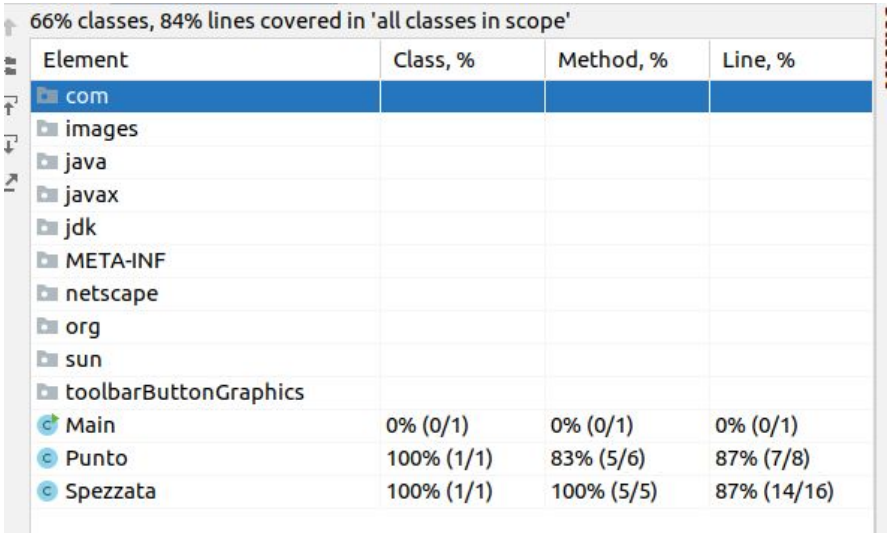
Idealmente non dovrebbe essere inferiore all’80%. L’ideale sarebbe 100%

Non dice nulla sulla QUALITA’ dei test

Molti IDE hanno il coverage integrato, per altri occorre affidarsi a tool esterni (es: Cobertura)

“Run all test with coverage”





Checkpoint #2:

- *Sapete testare le eccezioni*
- *Sapete disabilitare selettivamente i test*
- *Sapete valutare il coverage*

3.
Mocking

Crash test dummies

Unit testing

Rivediamo le caratteristiche di Unit test:

- Esaminano una unica classe/funzione
- Richiedono alta velocità di esecuzione
- Non sono influenzati da eventi esterni (servizi web o database)
- Non effettuano I/O

Se queste condizioni non sono soddisfatte parliamo di test di integrazione o end-to-end.

Unit testing?

Come facciamo ad effettuare unit test anche se queste condizioni non sono possibili?

Ad esempio, la nostra classe Spezzata usa l'oggetto Punto.

Come fare?

Arrivano i Mock objects

mock object (oggetti simulati) sono degli oggetti simulati che riproducono il comportamento degli oggetti reali in modo controllato.Un oggetto mock si usa per testare il comportamento di altri oggetti, reali, ma legati ad un oggetto inaccessibile o non implementato.

Un mock ha la stessa interfaccia dell'oggetto che simula, autorizzando così l'oggetto client a ignorare se interagisce con un oggetto reale o simulato.

“Controfigure” - definizione di Martin Fowler

- **Dummy** objects are passed around but never actually used. Usually they are just used to fill parameter lists.
- **Fake objects** actually have working implementations, but usually take some shortcut which makes them not suitable for production (an in memory database is a good example).
- **Stubs** provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test. Stubs may also record information about calls, such as an email gateway stub that remembers the messages it 'sent', or maybe only how many messages it 'sent'.
- **Mocks** are what we are talking about here: objects pre-programmed with expectations which form a specification of the calls they are expected to receive.

Esempio #1 - Spezzata

- Nel setup, utilizziamo l'oggetto Punto. Se l'implementazione di Punto cambia, anche i test possono cambiare. Questo viola il principio degli unit test.

```
void setup() {

  spezzata=new Spezzata();
  spezzata.aggiungi(
     new Punto(0,0));
  spezzata.aggiungi(
     new Punto(1,2));
     }
```

Esempio #1 - SpezzataMock

- Creiamo oggetti Mock che hanno la stessa firma di Punto - grazie all'ereditarietà, ma restituiscono valori fissi e predefiniti.

```
 class Punto00 extends Punto
{
        @Override public
double x() { return 0; }
        @Override public
double y() { return 0; }
        @Override public
String toString() { return
"(0.0, 0.0)"; }
    }
```

Esempio #1 - SpezzataMock

- Il nuovo test non ha quindi più alcuna dipendenza esterna.

```
void setup() {
    spezzata=new Spezzata();
    spezzata.aggiungi(
       new Punto00());
    spezzata.aggiungi(
       new Punto12());
    }
```

Esempio #2 - Uso della rete

In questo test, si utilizza la rete. Questo vincola il test all'essere online e rallenta terribilmente i risultati.

```
class FetchHTMLTest {


    @Test
    void fetchURL() {
        try {
            FetchHTML url = new
FetchHTML("https://www.youmath.it/domand
e-a-risposte/view/6223-linea-spezzata.ht
ml");
            url.fetch();
            assertEquals("<!doctype
html><html
xmlns=\"http://www.w3.org/1999/" ,url.toS
tring().substring(0,51));
        } catch (IOException e) {}
    }
}
```

Esempio #2 - Uso della rete

Creare un oggetto che simula il recupero di dati dalla rete. Il testo è 4K.

```
class FetchMockYoumath extends
FetchHTML{



   public FetchMockYoumath() throws
IOException {


super("https://www.youmath.it/domande-a-
risposte/view/6223-linea-spezzata.html" )
;

   }

   public void fetch(String theURL)  {

       this.HTML="<!doctype html><html
xmlns=\"http://www.w3.org/1999/xhtml \"
xml:lang=\"it-it\" lang=\"it-it\"
dir=\"ltr\" ><head><base " …"; }

 }
```

Esempio #2 - SpezzataMock

Inserita come classe interna al test fa in modo che non ci siamo più dipendenze esterne.

- FetchURL
- FetchHTMLMockTest

```
    @Test
void fetchURL() {
    try {
        FetchHTML url =
            new FetchMockYoumath();
        url.fetch();
        assertEquals("<!doctype html><html
xmlns=\"http://www.w3.org/1999/" ,url.toS
tring().substring(0,51));
    } catch (IOException e) {}
}
```

Dependency injection (teoria)

Il mocking è basato su uno dei più noti design pattern, ovvero il **Dependency injection**. Risolve i seguenti problemi;

- Rendere indipendente la creazione di un oggetto da quelli da cui dipende?
- Come configurare gli oggetti creati, magari in modi diversi?

Dependency injection separates the creation of a client's dependencies from the client's behavior, which promotes loosely coupled programs and the dependency inversion and single responsibility principles.

Mocking frameworks

Mocking Frameworks

- Il mocking può richiedere parecchio tempo e la costruzione di parecchio codice (che poi va manutenuto).
- La tradizionale pigrizia del programmatore è un fattore molto importante
- Per questo motivo esistono diversi framework che rendono la gestione del mocking più accettabile.

Mockito (Java)

- Mockito è uno dei framework più popolari per la realizzazione di oggetti mock in Java. Consente di generare un mock a partire sia da una interfaccia che da un classe semplicemente dichiarandone il comportamento, ed inoltre permette di eseguire diverse tipologie di test.
- Mockito deve essere installato separatamente ed esula dagli scopi del corso.

Mockito (Esempio da vogella.com)

```
@ExtendWith(MockitoExtension.class)
class ServiceDatabaseIdTest {
    @Mock
    Database databaseMock;
    @Test
    void ensureMockitoReturnsTheConfiguredValue() {
        // define return value for method getUniqueId()
        when(databaseMock.getUniqueId()).thenReturn(42);

        Service service = new Service(databaseMock);
        // use mock in test....
        assertEquals(service.toString(), "Using database with id: 42");
    }
}
```

Python - unittest.mock

```
from unittest.mock import MagicMock
thing = ProductionClass()
thing.method = MagicMock(return_value=3)
thing.method(3, 4, 5, key='value')
3
thing.method.assert_called_with(3, 4, 5, key='value')
---- ECCEZIONE FINTA
mock = Mock(side_effect=KeyError('foo'))
mock()
Traceback (most recent call last):
 ...
KeyError: 'foo'
```

Jest (Javascript)

- Jest è il framework di riferimento per Javascript, e contiene internamente un sistema di Mocking

```
const myMock = jest.fn();
console.log(myMock());
// > undefined

myMock.mockReturnValueOnce(10).mockReturnValueOnce('x').mockReturnValu
e(true);

console.log(myMock(), myMock(), myMock(), myMock());
// > 10, 'x', true, true
```

Checkpoint #3:

- *Sapete create test di unità davvero indipendenti*
- *Sapete cos'è un framework di mocking*

GRAZIE!

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Startupstockphotos
- Anil Gupta (www.guptaanil.com)
- Pete Nicholls (github.com/Aupajo)
- Armando Fox

Presentation design

This presentation uses the following typographies and colors:

- Titles: **Dosis**
- Body copy: **Roboto**

You can download the fonts on these pages:

https://www.fontsquirrel.com/fonts/dosis

https://material.google.com/resources/roboto-noto-fonts.html

- Orange **#ff8700**

You don't need to keep this slide in your presentation. It's only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

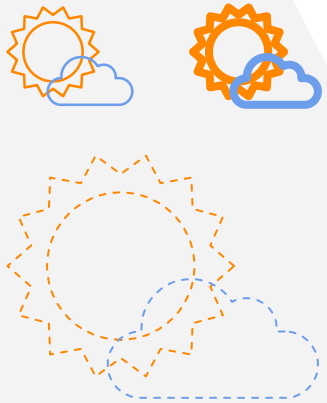
SlidesCarnival icons are editable shapes.

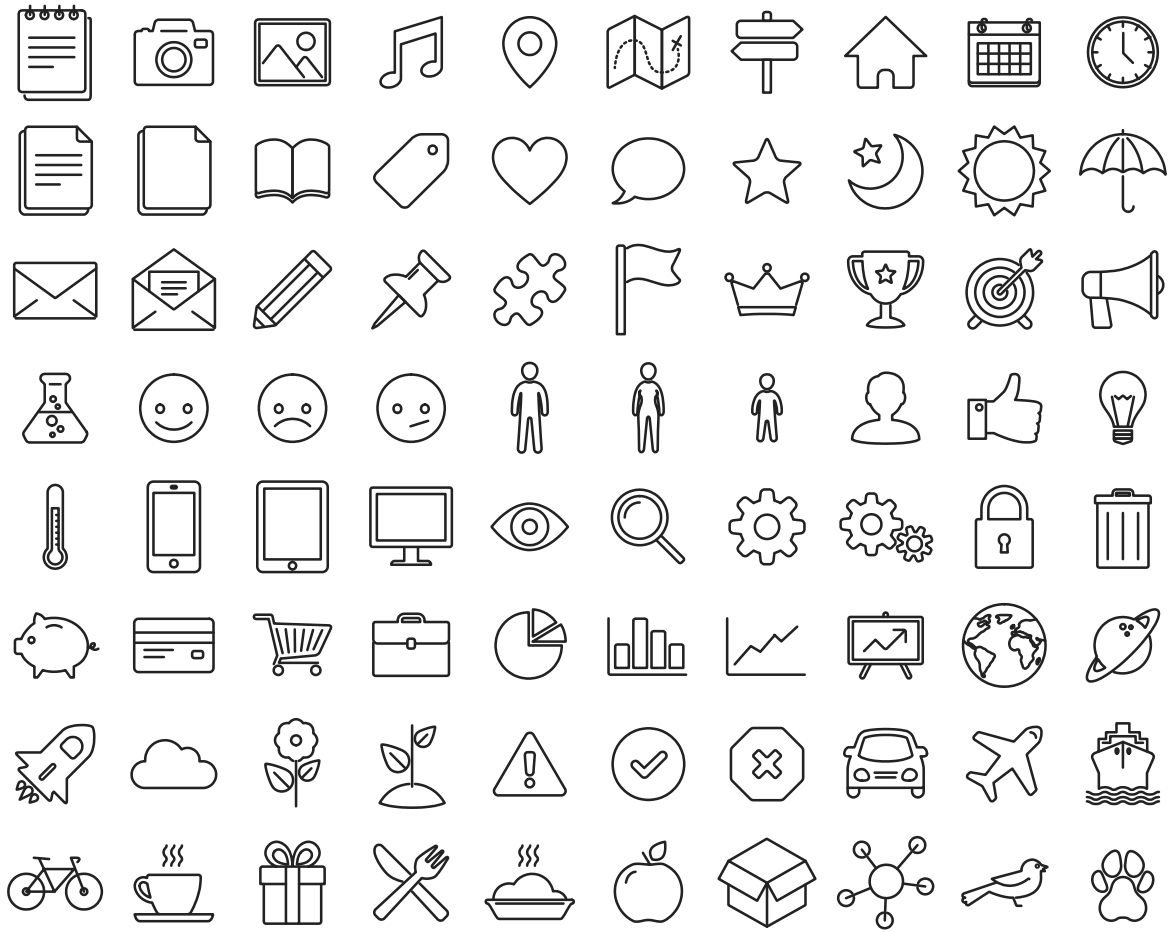
This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples:





Now you can use any emoji as an icon!
And of course it resizes without losing quality and you can change the color.

How? Follow Google instructions
https://twitter.com/googledocs/status/730087240156643328