JUnit

JUnit è un framework per l’automazione del testing di unità di programmi.
Ogni test contiene le asserzioni che controllano che il programma non contenga difetti

Junit

Usa la riflessione di Java (i programmi Java possono esaminare il loro stesso codice)

supporta i programmatori nel:

- definire ed eseguire test e test set
- formalizzare in codice i requisiti sulle unità
- scrivere e debuggare il codice
- integrare il codice e tenerlo sempre funzionante

Integrato con molti IDE (eclipse, netbeans, ...)

L'ultima versione 4 sfrutta le annotation di Java

- useremo questa versione, codice molto più semplice

Test di una classe

Per testare una classe X si crea una classe ausiliaria (in genere con nome XTest)

XTest contiene dei metodi annotati con @Test che rappresentano i casi di test

Test di un metodo

Nel singolo metodo di test testiamo ogni metodo della classe sotto test

Dobbiamo:

1. creare eventuali oggetti delle classe sottotest
2. chiamare il metodo da testare e ottenere il risultato
3. confrontare il risultato ottenuto con quello atteso
   - **per far questo usiamo dei metodi assert di Junit che permettono di eseguire dei controlli**
   - **se un assert fallisce, JUnit cattura il fallimento e lo comunica al tester**
4. ripetiamo da 1

Metodi assert

Ci sono molti metodi assert:

- **assertEquals**(*expected*, *actual*)
- **assertEquals**(String ***message***, *exp*, *act*)
  - **per controllare l'uguaglianza (con equals) di exp e act**
- **assertTrue**(expression)
  - per controllare che expression sia vera
- **assertNull**(Object ***object***)
- **fail**() e **fail**(String ***message***)
  - per terminare con un fallimento
- **assertSame**
  - per usare == invece che equals per il confronto

JUnit in Eclipse (1)

E' consigliabile usare un IDE come eclipse che assiste nella scrittura ed esecuzione dei casi di test

Per scrivere una caso di test:

1. scrivi la tua classe al solito (almeno lo scheletro)
2. seleziona la classe per cui vuoi creare i casi di test con il tasto destro -> new -> JUnit Test Case
3. appare un dialogo: selezione JUnit 4 e deseleziona tearDown, setUp ... - non sono necessari per piccoli esercizi

JUnit in Eclipse (2)

1. fai next -> seleziona i metodi per cui vuoi creare i casi di test
2. riempi i metodi di test con il codice che faccia i controlli opportuni

Per eseguire un caso di test in eclipse:

- tasto destro sulla classe di test -> run As -> Junit Test
- appare un pannello con una barra che rimane verde se tutto va bene

BeforeClass

- Se un metodo deve essere eseguito una sola volta prima dei test usa:
- @BeforeClass
  - Il metodo deve essere statico e public
- Esempio:
- @BeforeClass public static void onlyOne(){
- …
- }

eccezioni

- È possibile verificare la presenza di eccezioni, quando un metodo **deve** generare un'eccezione

```
@Test(expected=Exception.class)
public void verificaEccezione(){ …}
```

- Se un metodo non deve generare un'eccezione basterà non dire nulla (se non ci sono eccezioni controllate) o aggiungerle nel throws:

  public void maiEccezione() throws Exp { …}

- Se si verifica l'eccezione, il test fallisce

Test parametrici con Junit

- Alcune volte si vuole chiamare lo stesso test con alcuni dati
  - Il codice che effettua il test non cambia
  - I dati di ingresso e il controllo sì

- Esempio:
  - Un metodo incrementa inc(x), voglio testarlo con 0, 2 e 5 (e controllare che dia 1,3, 6)
  - Invece che scrivere 3 tests, posso utilizzare I test parametrici
    - Ho due parametri: input e inputIncrementato

Test parametrici (2)

- si dichiarano tante variabili d'istanza quanti sono i i parametri usati nel test
- si crea un costruttore del test che ha per parametri la n- upla identica alle variabili d'istanza
- si crea un metodo statico @Parameters che deve restituire una Collection (contiene le n-uple di parametri con i valori)
- si annota la classe con @RunWith(Parameterized.class)

```
@RunWith(Parameterized.class)
public class ParameterTest{
  private int input;
  private int inputIncrementato;
  // construttore
  public ParameterTest(int p1, int p2){
      input = p1; inputIncrementato = p2; }
// parametri
@Parameters public static Collection creaParametri()
 {return Arrays.asList(new Object[][] {
     {0, 1 },
     {2, 3 },
     {5, 6 }}}); }
// test
@Test
 public void testParametrico(){ ...//qui uso
 int outputAttuale = incrementa(input);
 assertEquals(inputIncrementato,outputAttuale);}
}
```