

# Programmazione ad Oggetti

## Test con JUnit

V 1.2 © Marco Torchiano 2005

## Test con JUnit

- JUnit è un ambiente di test per programmi Java
  - ♦ Sviluppato da Kent Beck
- E' un framework che offre tutte le funzionalità utili per il test
- E' integrato in Eclipse tramite un'interfaccia grafica che visualizza test passati e falliti

## Junit

---

- Test framework
  - ♦ plug-in di Eclipse
  - ♦ per fare test cases come codice Java
  - ♦ framework: set di classi e convenzioni per usarle

- Test code

```
testRaddoppia2(){  
    //...  
}
```

- Production code

```
int raddoppia(){  
    //...  
}
```

---

## Junit

---

- É possibile usare JUnit all'interno di Eclipse per eseguire i programmi invece di scrivere delle classi con il metodo `main()`.
  - É sufficiente:
    - ♦ scrivere una sotto-classe della classe `TestCase`
    - ♦ aggiungere ad essa dei metodi di test
-

## Elementi del framework

- `assert*()`
  - ◆ funzioni di confronto
- `TestCase`
  - ◆ classe contenente serie di test
  - ◆ una per ogni classe di production code
  - ◆ vari test per ogni metodo di classe production
- `TestSuite`
  - ◆ classe contenente serie di `TestCase`

## TestCase Class

```
import junit.framework.TestCase;
public class StackTester extends TestCase {
    public StackTester(String name) {
        super(name);
    }
    public void testStack() {
        Stack unoStack = new Stack();
        if(!unoStack.isEmpty()) {
            System.out.println("Lo stack dovrebbe essere vuoto!");
            unoStack.push(10);
            unoStack.push(-4);
            System.out.println("Ultimo elemento: " +
                unoStack.pop());
            System.out.println("Primo elemento: " +
                unoStack.pop());
        }
    }
}
```

**Codice generato automaticamente da Eclipse**

**Deve iniziare sempre con "test"**

## Assert\*()

---

- per condizione
    - ♦ `assertTrue("message when test fails", condition);`
  - per valori di ritorno `object`, `int`, `longs`, `byte`
    - ♦ `assertEquals(expected_value, expression);`
  - per valori di ritorno `float`, `double`:
    - ♦ `assertEquals(expected_value, expression, error);`
  - se la condizione testata
    - ♦ e' vera, esegue istruzione seguente
    - ♦ e' falsa, `break` a fine metodo
- 

## Assert (1)

---

- Sono metodi pubblici definiti nella classe `TestCase` che iniziano con "assert" e sono usati nei metod di test
    - ♦ `assertTrue("messaggio", condizione);`
    - ♦ Es. `assertTrue("stack non vuoto", unoStack.empty());`
  - Se la condizione è falsa:
    - ♦ Il test fallisce e il resto del metodo non viene eseguito
    - ♦ Viene stampato il messaggio
  - Se la condizione è vera non succede nulla
-

## Assert (2)

---

- Per oggetti, interi, long, byte:
    - ♦ `assertEquals( valore_atteso, espressione);`
    - ♦ ES. `assertEquals( 2 , unoStack.size() );`
  - Per valori floating point:
    - ♦ `assertEquals(valore_atteso, espressione, errore);`
    - ♦ ES. `assertEquals(1.0, Math.cos(3.14), 0.01);`
- 

## Assert Esempio

---

```
public void testStack() {
    Stack aStack = new Stack();
    assertTrue("Stack should be empty!",
        aStack.isEmpty());
    aStack.push(10);
    assertTrue("Stack should not be empty!",
        !aStack.isEmpty());
    aStack.push(-4);
    assertEquals(-4, aStack.pop());
    assertEquals(10, aStack.pop());
}
```

---

## Un concetto per volta..

---

```
public void testStackEmpty() {
    Stack aStack = new Stack();
    assertTrue("Stack should be empty!",
               aStack.isEmpty());
    aStack.push(10);
    assertTrue("Stack should not be empty!",
               !aStack.isEmpty());
}
public void testStackOperations() {
    Stack aStack = new Stack();
    aStack.push(10);
    aStack.push(-4);
    assertEquals(-4, aStack.pop());
    assertEquals(10, aStack.pop());
}
```

---

## Funzionamento

---

- Per un test case JUnit:
    - ♦ Esegue tutti i suoi metodi di test pubblici
      - Ovvero quelli che iniziano con “test”
    - ♦ Ignora tutto il resto
  - La classe può contenere metodi di supporto (helper methods)
    - ♦ Non sono pubblici oppure
    - ♦ Non iniziano con “test”
-

## TestSuite

---

- Combina vari test case in una test suite:

```
public class AllTests extends TestSuite {  
  
    public AllTests(String name) {  
        super(name);  
    }  
  
    public static TestSuite suite() {  
        TestSuite suite = new TestSuite();  
        suite.addTestSuite(StackTester.class);  
        suite.addTestSuite(AnotherTester.class);  
    }  
}
```

---

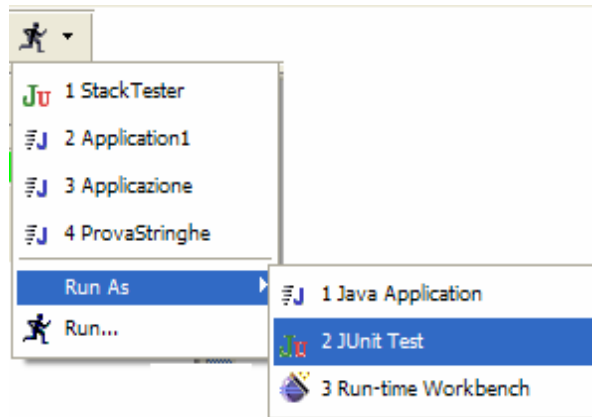
## Junit in Eclipse – Setup

---

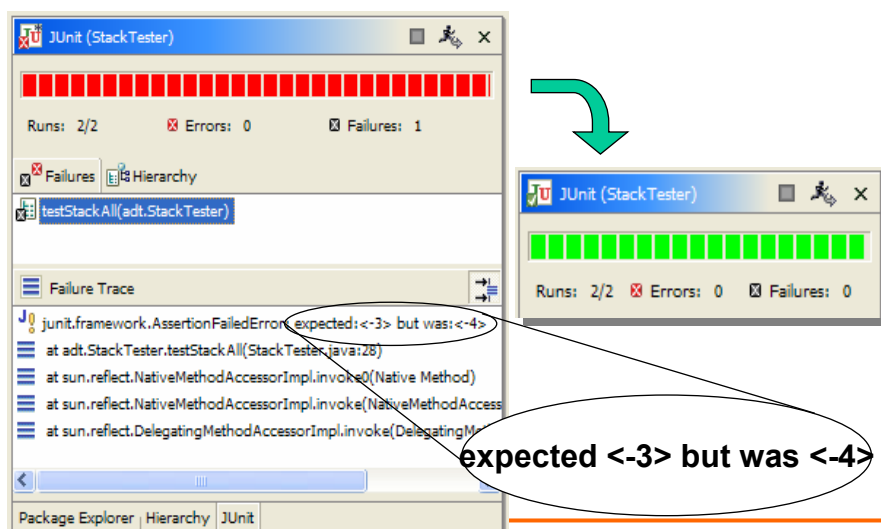
- In Eclipse
  - Aprire project's property window
  - java build path
  - libraries
  - Add external jar
    - ◆ add org.junit
-

## Junit in Eclipse – Run as JUnit Test

- Run
- Run As..
- Junit Test



## Red / Green Bar



...usate JUnit

---

**Keep the bar green to keep the code clean...**

