JUnit

Casos de Teste
Suítes de testes
Fixtures

Frederico Costa Guedes Pereira © 2006
fredguedespereira@gmail.com

Conceitos

Testes de unidade testam **unidades de lógica**. Em Java: métodos e objetos

Testes de unidade são programas que testam outros programas

Conceitos

Ah! Eu faço isso sempre! Toda vez que eu escrevo uma classe eu implemento um main() que testa os objetos dessa classe!

Conceitos

É um começo, mas isso não é uma forma "padronizada" nem prática de se fazer testes de unidade. Você já ouviu falar no JUnit?

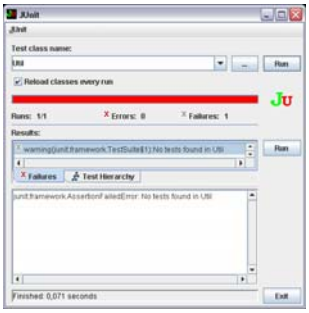
JUnit

- Framework para definição e execução de testes de unidade em Java
  - Testes de unidade: feito por programadores para programadores
- Desenvolvido por Kent Beck (XP) e Erich Gamma (GoF)
  - Site: http://www.junit.org
- JUnit não é uma metodologia de testes!
  - É uma tecnologia de testes!

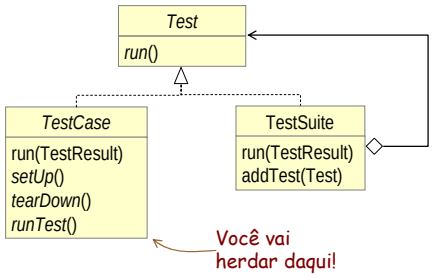
JUnit

- Como fazer para utilizar?
  - Pôr o arquivo junit.jar no *classpath*
  - Escrever classes *casos de teste*
  - Escrever *métodos de teste* nestas classes de casos de teste (devem começar com o prefixo *test*)
  - Escrever, opcionalmente, *suítes de teste* que agrupem vários casos de teste
  - Executar o caso de teste ou a suíte de testes usando o *TestRunner*, ferramentas que vem com o JUnit
  - Analisar os resultados: SUCESSO ou FALHA

O *TestRunner*



API do JUnit



Anatomia de um caso de teste

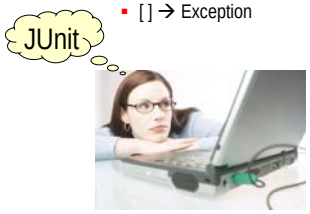
Caso de teste

```
public class TesteUtil extends TestCase {
    public TesteUtil(String name) {
        super(name);
    }
    Método de teste
    public void testSimples() {
        assertEquals(9, Util.maior(new int[] {7,9,8}));
    }
}
```

- Os métodos de teste devem começar com *test*
  - JUnit utiliza reflexão

Pensando nos casos de testes

- Caso simples:
  - [1,3,2] → 3
- Extremos:
  - [3,1,2] → 3
  - [1,2,3] → 3
- Duplicidade
  - [1,3,2,3] → 3
- Arranjo unitário
  - [1] → 1
- Valores negativos
  - [-1,-2,-3] → -1
- Arranjo vazio
  - [ ] → Exception



Para escrever os testes

- Métodos herdados de TestCase:
  - assertEquals(*a, b*)
  - assertTrue(boolean b)
  - assertFalse(boolean b)
  - assertNull(Object o)
  - assertNotNull(Object o)
  - assertSame(Object a, Object b)
  - assertNotSame(Object esp, Object real)
  - fail(String msg)

Suítes de testes

- Por reflexão, o JUnit testa todos os métodos que começam por ***test***
- Suite de testes:
  - Você define quais métodos testar
  - Você pode compor vários casos de testes num único teste
- Implemente o método estático suite:

```
public static Test suite() {...}
```

Suítes de testes

```
public class CasoUm extends TestCase {
    public void testUm() {...}
    public void testDois() {...}
    public void testTrês() {...}
}
```

suíte

```
public class CasoDois extends TestCase {
    public void testUm() {...}
    public void testDois() {...}
    public void testTrês() {...}
    public void testQuatro() {...}
    public void testCinco() {...}
}
```

suíte

Suítes de testes

```
public class CasoUm extends TestCase {
    public void testUm() {...}
    public void testDois() {...}
    public void testTrês() {...}

    public static Test suite() {
        TestSuite suite = new TesteSuite();
        suite.addTest(new CasoUm("testUm"));
        suite.addTest(new CasoUm("testDois"));
        return suite;
    }
}
```


Suítes de testes

```
public class CasoDois extends TestCase {
    public void testUm() {...}
    public void testDois() {...}
    public void testTrês() {...}
    public void testQuatro() {...}
    public void testCinco() {...}

    public static Test suite() {
        TestSuite suite = new TesteSuite();
        suite.addTest(new CasoDois("testTres"));
        suite.addTest(new CasoDois("testQuatro"));
        suite.addTest(new CasoDois("testCinco"));
        return suite;
    }
}
```


Compondo suítes de testes

```
public class TesteComposite extends TestCase {
    public TesteComposite(String metodo) {
        super(metodo);
    }

    public static Test suite() {
        TestSuite suite = new TesteSuite();
        suite.addTestSuite(CasoUm.class);
        suite.addTestSuite(CasoDois.class);
        return suite;
    }
}
```


Fixtures

- Testes devem ser independentes entre si
- Testes muitas vezes possuem contextos iniciais comuns
- Métodos de TestCase:
  - setUp() throws Exception
    - Prepara o ambiente (fixture) para cada caso de teste. Executado imediatamente antes de cada teste.
  - tearDown() throws Exception
    - Desfaz a preparação. Executado logo depois de cada teste.

Fixtures

```
public class TesteQualquer extends TestCase {

    public void testUm()   { }

    public void testDois() { }

    public void testTrês() { }

    protected void setUp() { }

    protected void tearDown() { }

}
```

```
setUp()
    testUm()
tearDown()

setUp()
    testDois()
tearDown()

setUp()
    testTrês()
tearDown()
```

Referências

- Beck, Kent *JUnit Test Infected: Programmers Love Writing Testes*.
- Fowler, Martin. *Refatoração – Aperfeiçoando o Projeto de Código Existente*. Bookman. 2004.
- Hunt, Andrew e Thomas, David. *Pragmatic Unit Testing – In Java with JUnit*. 2003
- Teles, Vinícius M. *Extreme Programming*. Novatec. 2004.
- http://www.junit.org