Universidade de São Paulo
Brasil

Aula 04 – Testes Automáticos com JUnit

MAC0321 - Laboratório de Programação Orientada a Objetos

Professor: Marcelo Finger (mfinger@ime.usp.br)

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

Como verificar a correçãodos programas?

Tópicos

1. O que é JUnit?
2. Anotações em Java
3. Assertivas
4. Programação de Testes
5. Testes com impressão

O que é JUnit?

- Framework Open Source
- Apoio à automatização de testes em Java
- Apoio à fase de **testes de unidades**
- “... é um framework, open source e de simples manipulação, que tem a funcionalidade de apoiar a escrita e execução de código de teste para programas Java ...”
- “... o framework utiliza a arquitetura xUnit, que é o padrão para frameworks automáticos de testes de unidades, estruturados e eficientes em atividades normais de desenvolvimento...”

Pra que serve o JUnit?

- Depuração é uma das maneiras mais simples de se escrever código de teste durante seu desenvolvimento (`System.out.println(meuValor);`)
  - **Limitação:** *Dependência de julgamento humano*
- Testes com JUnit não requerem interpretação humana
- Escrever código de teste de maneira simples e executar muitos testes ao mesmo tempo
- Útil para o apoio ao Teste de Regressão (novas versões)
- Reexecutar testes automaticamente quantas vezes for necessário
- Framework mais popular na indústria de software para testes em unidades de programas Java

Teste de Unidade

- Objetivo: testar a menor parte do código garantindo maior qualidade do produto no processo de desenvolvimento.
- Teste de um único componente isolado em uma maneira reutilizável e replicável.
- xUnit fornece uma API simples, mas eficaz para a execução dos testes de unidade.
- Em Java, o teste de unidade pode ser feito, por meio do JUnit, em cada método separadamente.
- Pode-se afirmar que JUnit configura um **padrão** para o teste de unidade Java.

Teste de Unidade **NÃO** é Teste Unitário

- Teste unitário é só uma má tradução da expressão em inglês, UNIT TEST.
- Várias Linguagens de programação possuem módulos para testes de unidade.
- Algumas possuem mais de uma biblioteca de testes de unidade
- Em Java, JUnit tem o status de padrão!

Instalação no Eclipse

- Garantir que Junit 5 está baixado
- Ao criar um projeto Java, clicar em Next
  - Selecionar Aba de Libraries, selecionar ~~module path~~ class path
  - Add Library, Selecionar Junit, Junit5, Finish
- Em projeto existente, inspecionar Properties
  - Selecionar Java Build Path
  - Selecionar Aba de Libraries, selecionar ~~module path~~ class path
  - Add Library, Selecionar Junit, Junit5, Finish

Utilização no Eclipse

- Fase 1. Criar uma classe de Teste para uma **classe pública**
  - **Nota**: só os métodos públicos são testados
- Fase 2. Escrever um método de teste com uma assertiva que indique o resultado esperado da AUT (Application Under Test) por meio de métodos assert*, importando-os estaticamente de:
  - JUnit 4: org.junit.Assert.*
  - Junit 5: org.junit.jupiter.api.Assertions.*;
- Fase 3. Executar os testes

Fase 1 - Classe de Teste JUnit5

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.*;

class LibraryTest {

…

}
```

Fase 2 – Escrita de Testes

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.*;
class LibraryTest {
    @Test
    public void bookInLibrary() { // test method
        Library library = new Library();
        boolean result = library.checkTitle("King Lear");
        assertEquals(true, result);
    }
}
```

Nota: `@Test` é uma *Anotação de Java*

Anotações em Java

Java Annotations

- Anotações são uma forma de metadados
- Fornecem dados sobre um programa, que não fazem parte do programa em si
- Anotações não têm efeito direto na operação do código que anotam.

Para que servem as Annotations?

Informações para o compilador — As anotações podem ser usadas pelo compilador para detectar erros ou suprimir avisos.

Processamento em tempo de compilação e implantação — As ferramentas de software podem processar informações de anotação para gerar código, arquivos XML e assim por diante.

Processamento em tempo de execução — Algumas anotações estão disponíveis para serem examinadas em tempo de execução.

Anotações podem ser **programadas,** como parte da técnica de *Reflexão de Código*

Neste curso, iremos apenas utilizar anotações pré-definidas.

Annotations JUnit

Junit 4

@After

@AfterClass

@Before

@BeforeClass

@Ignore

@Parameters

@RunWith

@SuiteClasses

@Test

Junit 5

@AfterEach

@AfterAll

@BeforeEach

@BeforeAll

@Disables

@Parameters

@ExtendWith

@SuiteClasses

@Test

Programação de testes

Assertivas do JUnit

Assertiva	Função
assertNull(Object x)	Valida o teste quando o parâmetro for null
assertNotNull(Object x)	Valida o teste se o parâmetro não for null
assertTrue(boolean x)	Valida o teste se o parâmetro é true
assertFalse(boolean x)	Valida o teste se o parâmetro é false
assertEquals(Object x,Object y)	Valida o teste por meio do método equals(Object obj1,Object obj2)
assertSame(Object x, Object y)	Valida o teste por meio do operador Java
assertNotSame(Object x, Object y)	Valida o teste por meio da negação do operador Java
fail()	Sempre falha

Anotações e Assertivas em Ação

Ver programa exemplo fornecido

Utilização do main()

- O advento dos IDE runners tornou essa etapa menos importante, ou melhor, desnecessária
- Alguns testadores acham que é uma prática auxiliar do teste
- Nós não utilizaremos isso

```
public static void main (String args[]) {
    org.junit.runner.JUnitCore.main("Nome (completo) do Test");
}
```

Fase 3: Execução

- Para rodar os testes, com um runner, a partir de uma classe com o método main():
- `java ClassTest`
- Para rodar o teste, por linha de comando, a partir do console (não é necessário que a classe ClassTest tenha um método main()):
- `java org.junit.runner.JUnitCore ClassTest`

Testando Valores Double

- Com valores double, igualdade não deve ser usada
- Usamos

assertEquals(double expected, double actual, double epsilon)

- Equivalente a usar:

assertTrue(|d1-d2| < epsilon)

Considere usar: assertTrue(|d1-d2|/|d1| < epsilon) para regularizar a dimensão dos valores

Equivalentemente: assertEquals(1.0, actual/expected, epsilon)

Falha por Timeout

- Testes são falhos quando demoram mais do que o timeout
- O tempo é contado em segundos
- Não precisamos de assertivas. O teste abaixo sempre falha

```
@Test
@Timeout(5)
public void infinity() {
    while (true);
}
```

Timeout detalhado

- `@Timeout(value = 2, unit = TimeUnit.XX)`
- `XX =` **`DAYS | HOURS | MICROSECONDS | MILLISECONDS | MINUTES | NANOSECONDS | SECONDS`**

Testes com Impressão

Teste de impressão

- Para testar a impressão, redirecionamos a saída padrão e a saída de erros
- Esse redirecionamento cria arquivos temporários
- Podemos então verificar se o conteúdo do arquivo era o esperado.
- Recursos utilizados
  - ByteArrayOutputStream
  - System.setOut()
  - System.setErr()

Setup do Teste de Impressão

- Para testar a impressão, redirecionamos a saída padrão e a saída de erros
- Esse redirecionamento cria arquivos temporários
- Podemos então verificar se o conteúdo do arquivo era o esperado.
- Recursos utilizados
  - ByteArrayOutputStream
  - System.setOut()
  - System.setErr()

Teste de Impressão

```
private final ByteArrayOutputStream outContent = new ByteArrayOutputStream();

private final ByteArrayOutputStream errContent = new ByteArrayOutputStream();


@BeforeEach
public void setUpStreams() {
    System.setOut(new PrintStream(outContent));
    System.setErr(new PrintStream(errContent));
}
@AfterEach
public void cleanUpStreams() {
    System.setOut(System.out);
    System.setErr(System.err);
}
```

Teste de Impressão

```
@Test
public void out() {
   System.out.print("hello");
   assertEquals("hello", outContent.toString());
}

@Test
public void err() {
   System.err.print("hello again");
   assertEquals("hello again", errContent.toString());
}
```

Lista de exercícios

No computador com o Eclipse

Entrega até o final do dia

MAC321
Lab POO

- Professor: Marcelo Finger

  E-mail: mfinger@ime.usp.br