

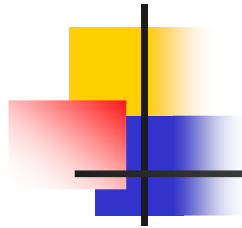
Padrões de projeto, testes automatizados e XML

03. Trabalhando com arquivos XML



Esp. Márcio Palheta

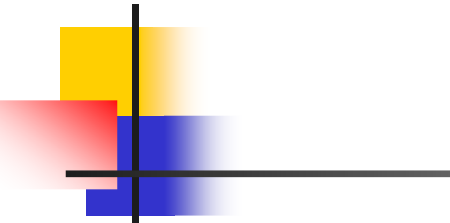
Gtalk: marcio.palheta@gmail.com



Trabalhando com XML

- Como vamos pegar os dados da bolsa de valores para popular nossas Candles?
- **XML** é o formato encontrado para resolver esse tipo de problema;
- Para isso, precisamos conhecer sua **estrutura e entender** como os dados estão agrupados;

Estrutura de arquivos XML



```
1 <list>
2   <negocio>
3     <preco>43.5</preco>
4     <quantidade>1000</quantidade>
5     <data>
6       <time>555454646</time>
7     </data>
8   </negocio>
9   <negocio>
10    <preco>44.1</preco>
11    <quantidade>700</quantidade>
12    <data>
13      <time>555454646</time>
14    </data>
15  </negocio>
16  <negocio>
17    <preco>42.3</preco>
18    <quantidade>1200</quantidade>
19    <data>
20      <time>559329406</time>
21    </data>
22  </negocio>
23 </list>
```

XML – eXtensible Markup Language

- Formalização da W3C para **linguagens** de **marcação** que atendem a um grande numero de necessidades;
- <http://www.w3c.org/XML/>
- Separação do conteúdo do formato
- Simplicidade e legibilidade
- Criação de **novas tags**
- Arquivos de validação – DTD's



XStream

- Boa alternativa para trabalhos com XML em **persistência**, transmissão de dados e configuração;
- Facilidade de uso;
- Utilizado em container de **inversão de controle**: NanoContainer; e
- Frameworks de **redes neurais**: Joone;



XStream

- Atua como **façade** de acesso aos principais recursos da API;
- O construtor recebe um **Driver** que vai gerar/consumir XML;
- O método **toXML()** retorna uma **String**;
- Sobrescrita de toXML() que recebe um Object e um **OutputStream** – grava, por exemplo, em arquivo ou socket;



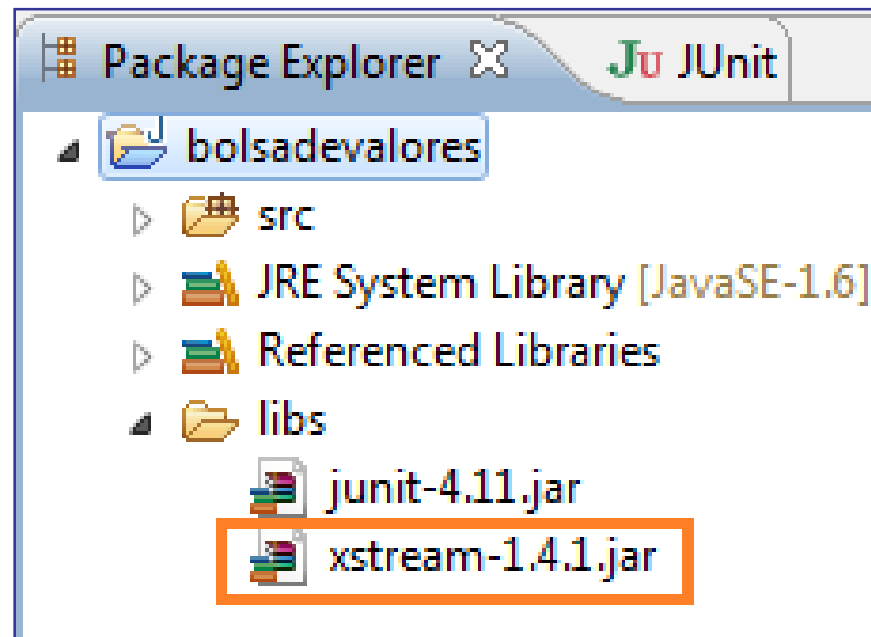
XStream

- **Serialização** de objetos através de atributos, não de **gets** e **sets**;
- A seguir, configurações iniciais do Xstream;

Exercício 01

Importação da biblioteca

- Copie o arquivo `xstream-1.4.1.jar` para a pasta **libs** criada e adicione ao path:





Adicione a **lib** ao classpath

- Clique com botão direito no arquivo e **xstream-1.4.1.jar** selecione **Build Path / Add to build path**;
- Depois disso, o eclipse passa a reconhecer as classes do **XStream**
- Agora, vamos criar um arquivo para escrita no formato XML;

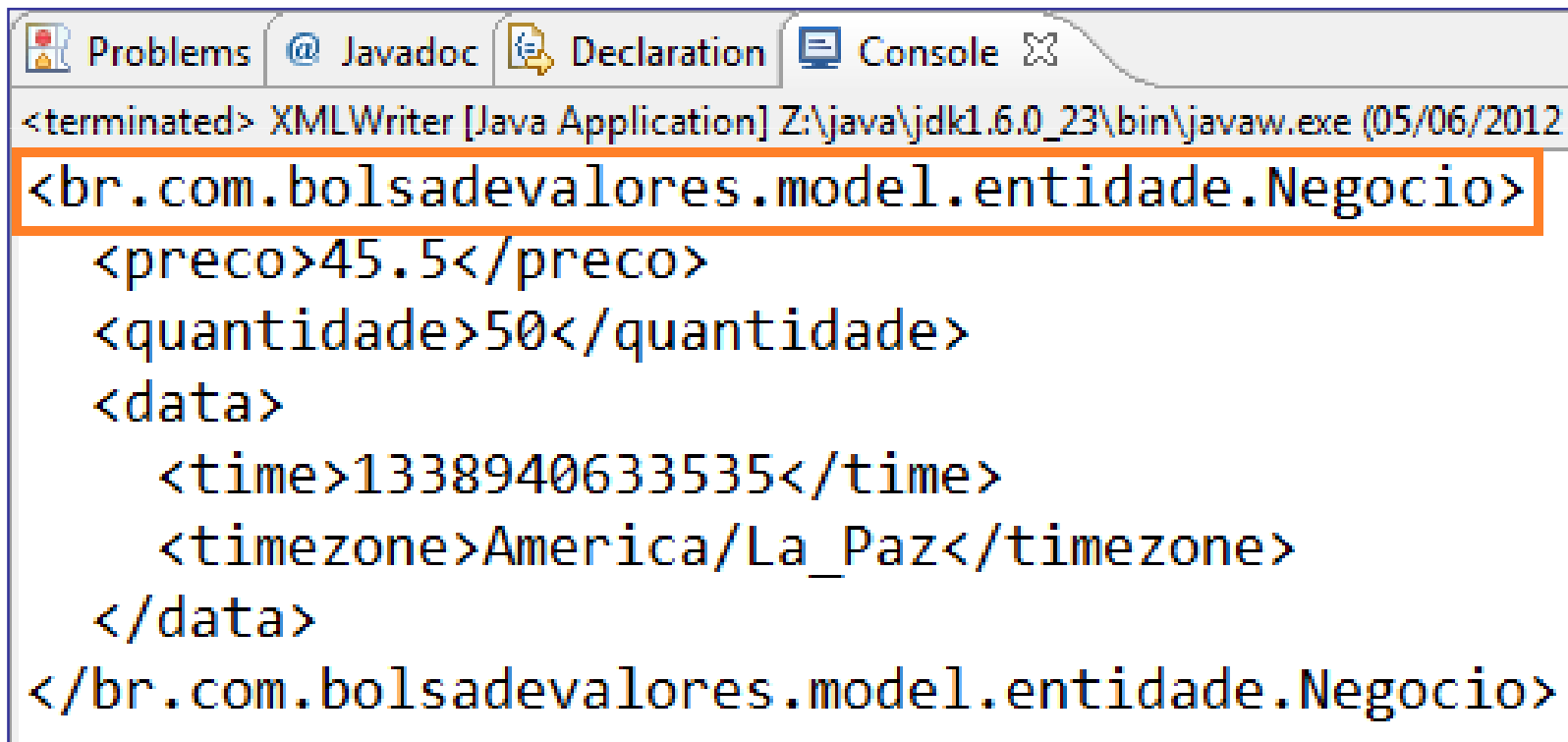
Exercício 02:

Escrevendo em formato XML

```
package br.com.bolsadevalores.xml.reader;
import java.util.Calendar;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
import br.com.bolsadevalores.model.entidade.Negocio;
public class XMLWriter {
    public static void main(String[] args) {
        // Definicao do objeto de negocio
        Negocio negocio = new Negocio(45.50, 50,
            Calendar.getInstance());
        // Definicao do objeto de escrita
        XStream stream = new XStream(new DomDriver());
        // Execucaao do metodo toXML()
        System.out.println(stream.toXML(negocio));
    }
}
```

Resultado gerado

- Definição da TAG baseada em **Negocio**:



The screenshot shows a Java IDE window with a console tab selected. The console output displays XML data generated by a Java application. The root element is `<br.com.bolsadevalores.model.entidade.Negocio>`, which is highlighted with an orange border. Inside this element, there are sub-elements for `<preco>`, `<quantidade>`, and `<data>`. The `<data>` element contains `<time>` and `<timezone>` sub-elements.

```
<terminated> XMLWriter [Java Application] Z:\java\jdk1.6.0_23\bin\javaw.exe (05/06/2012 :
<br.com.bolsadevalores.model.entidade.Negocio>
  <preco>45.5</preco>
  <quantidade>50</quantidade>
  <data>
    <time>1338940633535</time>
    <timezone>America/La_Paz</timezone>
  </data>
</br.com.bolsadevalores.model.entidade.Negocio>
```

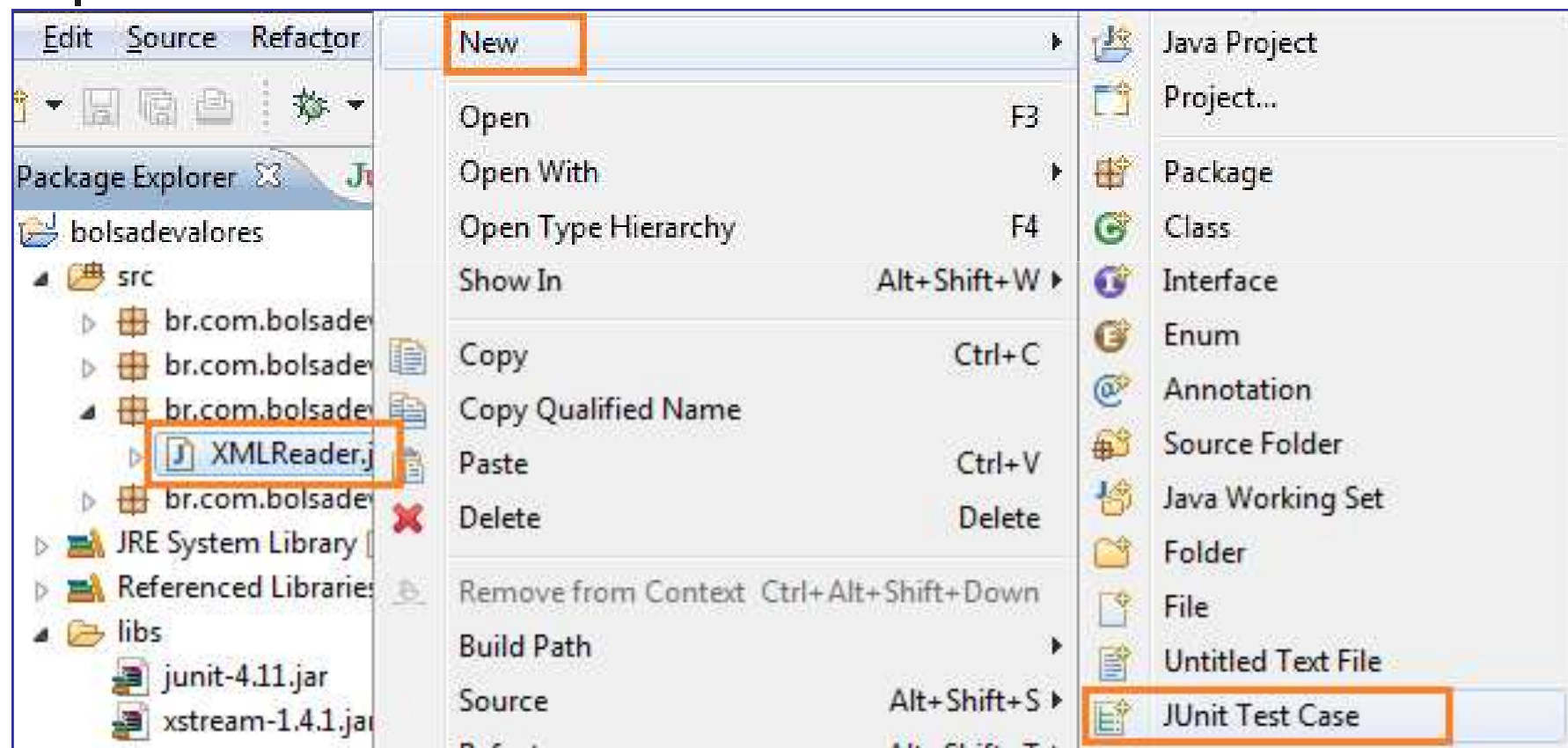
Exercício 03

Leitura no formato XML

```
package br.com.bolsadevalores.xml.reader;
import java.io.Reader;
import java.util.List;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
import br.com.bolsadevalores.model.entidade.Negocio;
@SuppressWarnings("unchecked")
public class XMLReader {
    public List<Negocio> carregar(Reader fonte){
        //Definicao do Objeto XStream
        XStream stream = new XStream(new DomDriver());
        //Atribuindo apelido aa tag Negocio
        stream.alias("negocio", Negocio.class);
        //Execucao do parse do XML
        return (List<Negocio>) stream.fromXML(fonte);
    }
}
```

Exercício 04 – Teste do leitor

Criação do teste unitário



Exercício 04 – Teste do leitor

Definição da classe

New JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:

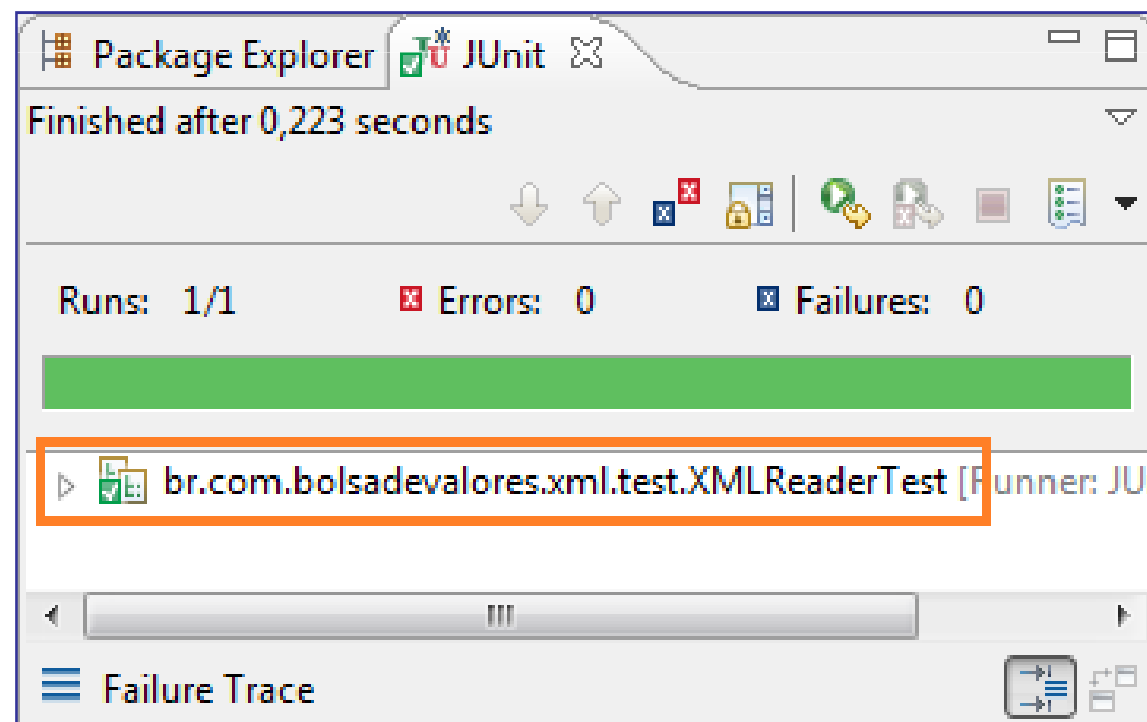
Exercício 04

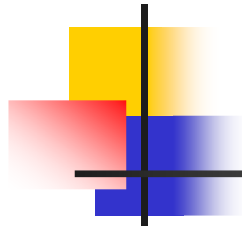
Classe de teste de leitura

```
package br.com.bolsadevalores.xml.test;
import java.io.StringReader;
public class XMLReaderTest {
    @Test
    public void testLeitorXML() {
        String xml = "<list>" +
            "<negocio>" +
            "    <preco>43.5</preco>" +
            "    <quantidade>1000</quantidade>" +
            "    <data>" +
            "        <time>555454646</time>" +
            "    </data>" +
            "</negocio>" +
            "</list>";
        XMLReader reader = new XMLReader();
        List<Negocio> lista = reader.carregar(new StringReader(xml));
        Assert.assertEquals(1, lista.size());
        Assert.assertEquals(43.50, lista.get(0).getPreco(), 0.00001);
        Assert.assertEquals(1000, lista.get(0).getQuantidade());
    }
}
```

Resultado dos testes

- Após a execução dos testes, verificamos que tudo ocorreu conforme o esperado:





Teste na mesma data

- Em nossa factory, vamos pegar uma série de **Negócios** e gerar uma **lista de Candles**;
- Separação de negócios por data;
- Precisamos verificar se todos os negócios estão na **mesma data**

Começando pelos testes

- Usando TDD, vamos primeiro testar;
- Novo método `isMesmoDia()`;

```
1 package br.com.bolsadevalores.model.teste;
2+ import java.util.Arrays;
10 public class CandleFactoryTest {
11     @Test
12     public void testComparaMesmoDiaCalendar() {
13         CandleFactory factory = new CandleFactory();
14         Calendar data1 = Calendar.getInstance();
15         Calendar data2 = Calendar.getInstance();
16         Assert.assertTrue(factory.isMesmoDia(data1, data2));
17     }
18     //outros metodos aqui...
```

Criação automática de método

- O código anterior não compila;
- Na linha do erro, pressione **Ctrl + 1**
- Selecione a opção
 - Create method **isMesmoDia(Calendar, Calendar)** in type **CandleFactory**

```
Calendar data2 = Calendar.getInstance();  
Assert.assertTrue(factory.isMesmoDia(data1, data2));  
}  
//outros metodos aqui...
```

● Create method 'isMesmoDia(Calendar, Calendar)' in type 'CandleFactory'
➤ Add cast to 'factory'
📄 Rename in file (Ctrl+2, R)



Método gerado

- O Eclipse inclui em **CandleFactory** o método **isMesmoDia(Calendar, Calendar)**
- Retorno padrão: **false**

```
package br.com.bolsadevalores.model.entidade;

import java.util.Calendar;

public class CandleFactory {

    public boolean isMesmoDia(Calendar data1, Calendar data2) {
        // TODO Auto-generated method stub
        return false;
    }

    //Outros metodos aqui
```



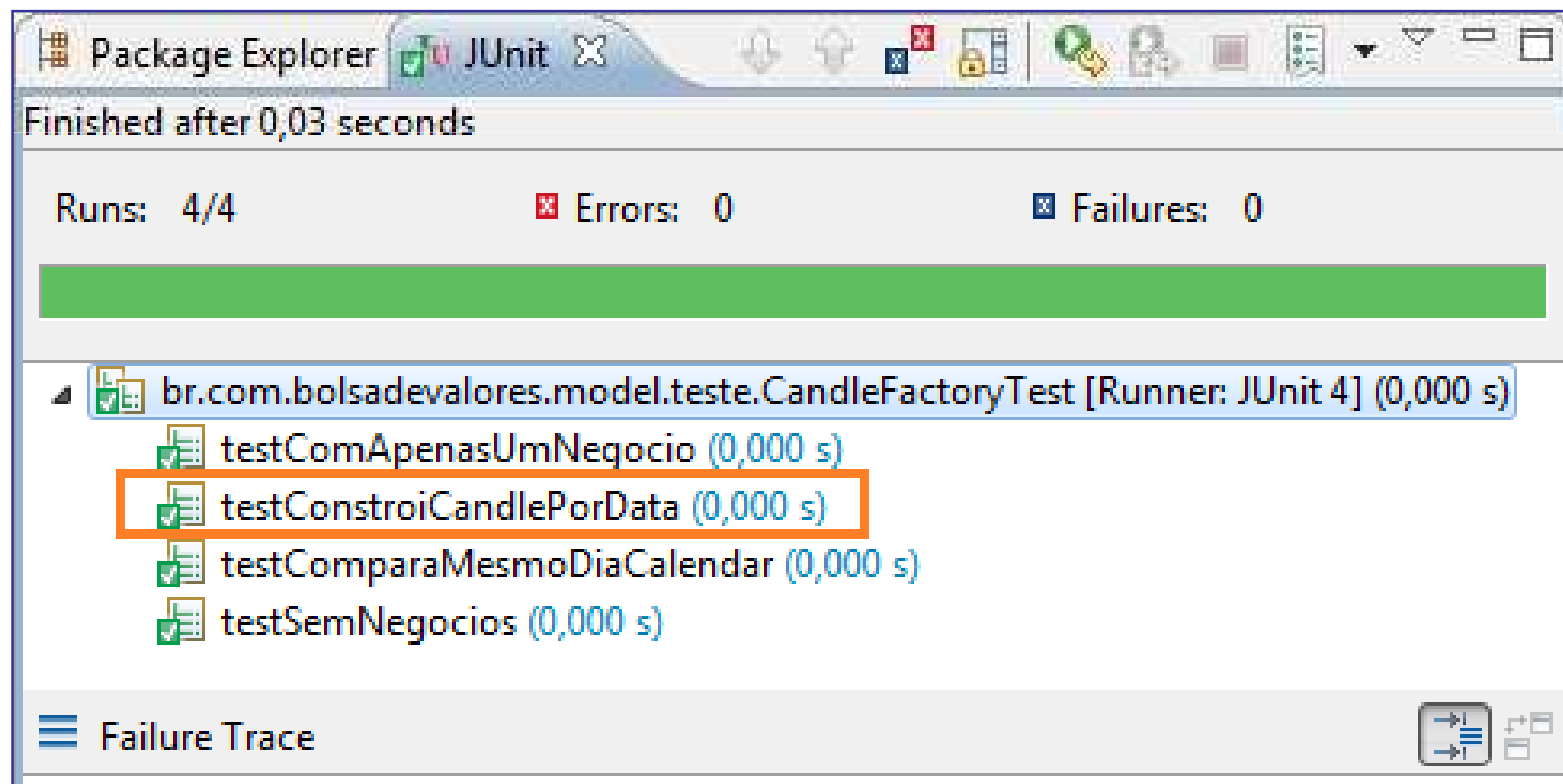
Atualização do método: CandleFactory.isMesmoDia()

- Vamos comparar as datas passadas como parâmetro para o método;
- Uso do método **Calendar.equals()**;

```
public class CandleFactory {  
  
    public boolean isMesmoDia(Calendar data1, Calendar data2) {  
  
        return data1.equals(data2);  
    }  
    //Outros metodos aqui
```

Executando o teste das datas

- Todos os testes funcionaram. 😊





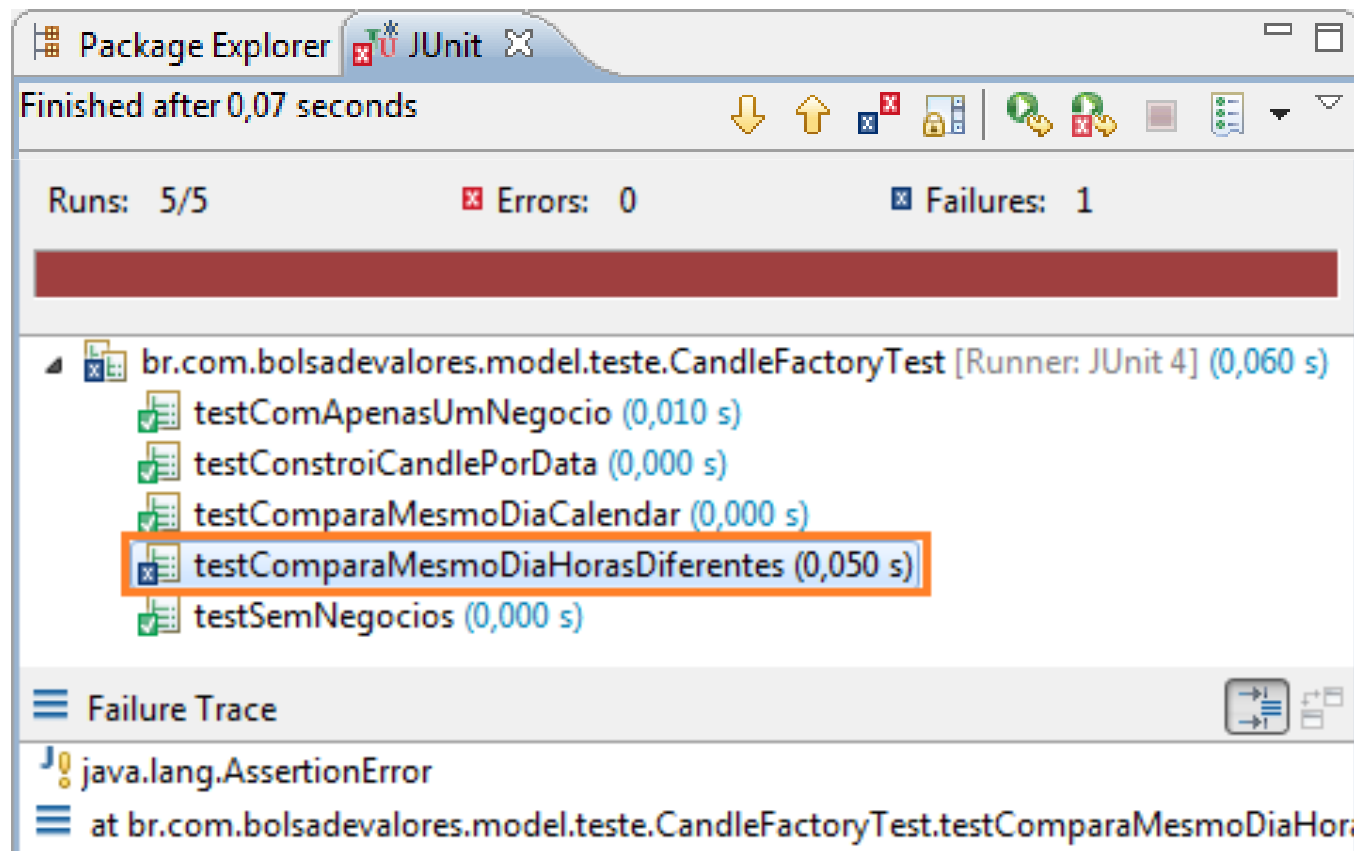
Novo teste de datas

- Vamos testar objetos com a mesma data, mas **horários diferentes**:

```
package br.com.bolsadevalores.model.teste;
import java.util.Arrays;
public class CandleFactoryTest {
    @Test
    public void testComparaMesmoDiaHorasDiferentes() {
        CandleFactory factory = new CandleFactory();
        // usando GregorianCalendar(year, month, day, hour, minute)
        Calendar data1 = new GregorianCalendar(2012, 12, 25, 8, 30);
        Calendar data2 = new GregorianCalendar(2012, 12, 25, 10, 30);
        //Verificando se as datas continuam validas
        Assert.assertTrue(factory.isMesmoDia(data1, data2));
    }
}
```

Qual foi o resultado?

- Por que o teste falhou?





Atualizando o código

- Calendar possui **timestamp**;
- O método *equals* não resolve o problema de **comparação**;

```
public class CandleFactory {  
    public boolean isMesmoDia(Calendar data1, Calendar data2) {  
        return data1.get(Calendar.DAY_OF_MONTH) == data2.get(Calendar.DAY_OF_MONTH)  
            && data1.get(Calendar.MONTH) == data2.get(Calendar.MONTH)  
            && data1.get(Calendar.YEAR) == data2.get(Calendar.YEAR);  
    }  
    //Outros metodos aqui
```

Teste: gerar lista de Candles a partir de lista de Negocios

```
public class CandleFactoryTest {
    @Test
    public void testConstruirCandlesDeMuitosNegocios() {
        //Definicao de objetos
        Calendar hoje = Calendar.getInstance();
        Negocio negocio1 = new Negocio(40.5, 100, hoje);
        Negocio negocio2 = new Negocio(45.0, 100, hoje);

        Calendar amanha = (Calendar) hoje.clone();
        amanha.add(Calendar.DAY_OF_MONTH, 1);
        Negocio negocio3 = new Negocio(48.8, 100, amanha);
        Negocio negocio4 = new Negocio(49.3, 100, amanha);

        Calendar depois = (Calendar) amanha.clone();
        depois.add(Calendar.DAY_OF_MONTH, 1);
        Negocio negocio5 = new Negocio(51.8, 100, depois);
        Negocio negocio6 = new Negocio(52.3, 100, depois);
        //Continua ...
    }
}
```

Continuação do código de teste – Não compila: Ctrl + 1

```
//Continua ...
```

```
List<Negocio> negocios = Arrays.asList(negocio1, negocio2,  
    negocio3, negocio4, negocio5, negocio6);
```

```
CandleFactory fabrica = new CandleFactory();
```

```
List<Candle> candles = fabrica.constroiCandles(negocios);
```

```
Assert.assertEquals(3, candles.size());
```

```
Assert.assertEquals(40.5, candles.get(0).getAbertura(), 0.00001);
```

```
Assert.assertEquals(42.3, candles.get(0).getFechamento(), 0.00001);
```

```
Assert.assertEquals(48.8, candles.get(1).getAbertura(), 0.00001);
```

```
Assert.assertEquals(49.3, candles.get(1).getFechamento(), 0.00001);
```

```
Assert.assertEquals(51.8, candles.get(2).getAbertura(), 0.00001);
```

```
Assert.assertEquals(52.3, candles.get(2).getFechamento(), 0.00001);
```

```
}
```

Atualização da CandleFactory

```
public class CandleFactory {  
    public List<Candle> constroiCandles(List<Negocio> negocios) {  
        List<Candle> candles = new ArrayList<Candle>();  
        // lista com negocios com mesma data  
        List<Negocio> negociosMesmoDia = new ArrayList<Negocio>();  
        Calendar dataPrimeiro = negocios.get(0).getData();  
        for (Negocio negocio : negocios) {  
            // se não for mesmo dia, fecha candle e reinicia variáveis  
            if (!isMesmoDia(dataPrimeiro, negocio.getData())) {  
                candles.add(constroiCandlePorData(dataPrimeiro,  
                    negociosMesmoDia));  
                // Nova colecao de negocios  
                negociosMesmoDia = new ArrayList<Negocio>();  
                // Nova data  
                dataPrimeiro = negocio.getData();  
            }  
            // Adiciona um negocio aa lista de negocios do mesmo dia  
            negociosMesmoDia.add(negocio);  
        }  
        // adiciona último candle  
        candles.add(constroiCandlePorData(dataPrimeiro, negociosMesmoDia));  
        return candles;  
    }  
}
```



Bibliografia

- Java - Como programar, de Harvey M. Deitel
- Use a cabeça! - Java, de Bert Bates e Kathy Sierra
- (Avançado) Effective Java Programming Language Guide, de Josh Bloch



Referências WEB

- **Site oficial:**

- SUN: www.java.sun.com

- **Fóruns e listas:**

- Javaranch: www.javaranch.com
- GUJ: www.guj.com.br

- **Apostilas:**

- Argonavis: www.argonavis.com.br
- Caelum: www.caelum.com.br

Padrões de projeto, testes automatizados e XML

03. Trabalhando com arquivos XML



Esp. Márcio Palheta

Gtalk: marcio.palheta@gmail.com