

Criando testes unitários em Java utilizando JUnit

Olá!

Eu sou a Franciele

- Tenho 22 anos
- Bacharela em Sistemas de Informação
- Analista de Sistemas no LEMAF/UFLA
- Mestranda em Engenharia de Sistemas e Automação na UFLA
- Membro organizador do GDG Lavras



/franciferreiraap



/franciele-ferreira



/francieleap



/franciferreiraap

O que aprenderão ?

- ▣ Como criar testes unitários do ZERO.
- ▣ Isolar o método de teste de dependências externas.
- ▣ Obter métricas de qualidade a partir dos testes.
- ▣ Aplicar o básico de TDD.
- ▣ Criar builders de objetos para centralizar a criação de entidades.

Roteiro

1. Informações básicas do minicurso.
2. Visão geral: testes unitários.
3. Trabalhando com JUnit.
4. Introdução à TDD.
5. Usando Data Builders.
6. Análise de Cobertura.
7. Conclusão

1.

Informações básicas do minicurso



Requisitos básicos:

- Conhecimento básico em: **Java**
- Ter instalado alguma IDE como: **Eclipse**
- O minicurso está disponível no seguinte link:
<https://github.com/francieleap/minicurso-junit>
- A documentação do JUnit está disponível no seguinte link: <https://junit.org/>

2.

**Visão geral:
testes unitários.**

O que são testes unitários?

- O **Teste Unitário** é uma modalidade de teste que é implementado com base no **menor elemento testável** (unidades) do software.
- Em **linguagens orientadas a objetos**, essa menor parte do código pode ser um **método** de uma classe.
- Etapas básicas para criação de um teste unitário:

CENÁRIO

AÇÃO

VALIDAÇÃO

Qual a importância do uso de testes unitários ?



- Evitar efeito borboleta:

“Uma coisa tão simples, quanto o bater de asas de uma borboleta, pode causar um tufão do outro lado do mundo”

- Vantagens do uso de testes unitários:
 - Permitem maior cobertura de teste.
 - Previnem regressão.
 - Incentivam o refactoring.
 - Evitam longas sessões de debug.
 - Servem como documentação.

**Vamos para o
código ...**

Aula-01: Testando sem usar framework

Importe no eclipse o projeto maven inicial do minicurso:

<https://github.com/francieleap/minicurso-junit/tree/master/Aula-01>

Adicionar este trecho:

```
29 public static void main(String[] args) {
30
31     //Cenário
32
33     LocacaoService service = new LocacaoService();
34     Usuario usuario = new Usuario("Usuário 01");
35     Filme filme = new Filme("Filme", 10, 5.0);
36
37     //Ação
38
39     Locacao locacao = service.alugarFilme(usuario, filme);
40
41     //Verificação
42
43     System.out.println(locacao.getValor() == 5);
44     System.out.println(DataUtils.isMesmaData(locacao.getDataLocacao(), new Date()));
45     System.out.println(DataUtils.isMesmaData(locacao.getDataRetorno(), DataUtils.adicionarDias(new Date(), 1)));
46 }
```

3.

**Trabalhando
com JUnit**

Alguns frameworks de testes unitários



LuaUnit

JUnit



Conhecendo o framework JUnit

- O **JUnit** é um framework de testes escrito por Erich Gamma e Kent Beck que facilita a implementação **de unidades de teste em Java**.
- JUnit é **open source** e oferece um conjunto de classes permitindo a fácil integração e execução regular de testes durante o processo de desenvolvimento
- Permite a **criação rápida de código de teste**.
- Checa os resultados dos testes e fornece uma **resposta imediata**.
- Pode ser utilizado da **linha de comando ou integrado** em IDE, e.x., Eclipse.

**Vamos para o
código ...**

Aula-02: Testando usando JUnit

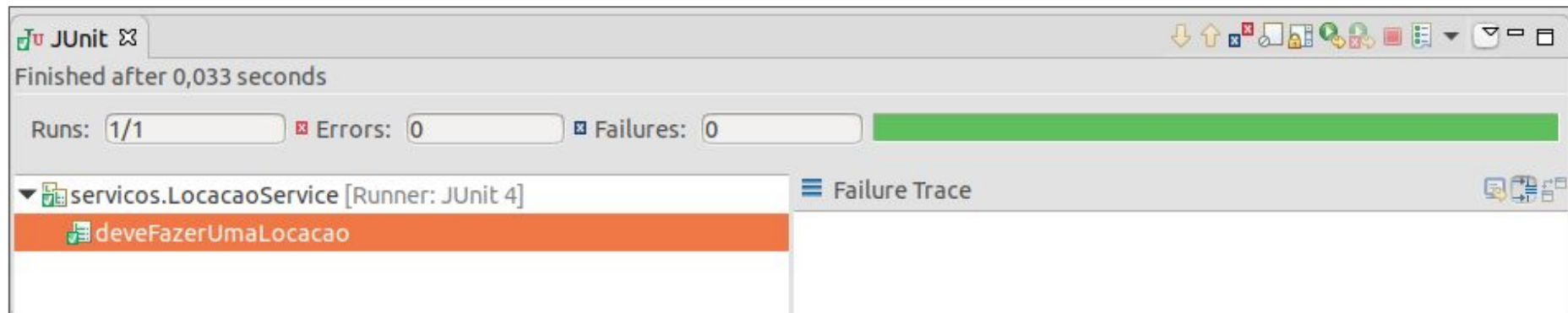
Importando JUnit 4.12:

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>br.com</groupId>
6     <artifactId>aula-02</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8
9     <dependencies>
10         <dependency>
11             <groupId>junit</groupId>
12             <artifactId>junit</artifactId>
13             <version>4.12</version>
14         </dependency>
15     </dependencies>
16 </project>
```

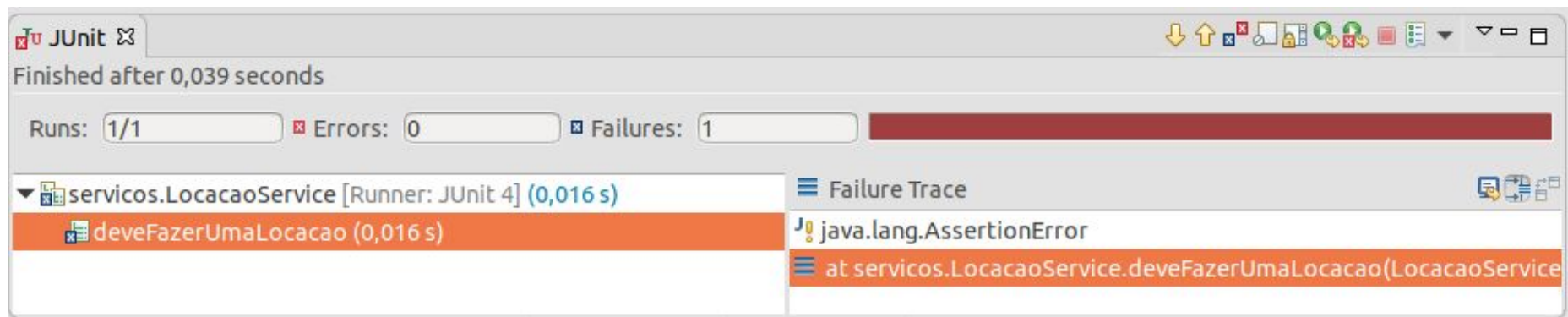
<https://github.com/francieleap/minicurso-junit/tree/master/Aula-02>

Aula-02: Testando usando JUnit

Teste com sucesso:



Teste com erro:



Aula-02: Testando usando JUnit

Adicionando primeiro teste:

```
@Test
public void deveFazerUmaLocacao() {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 10, 5.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertTrue(locacao.getValor() == 5);
    Assert.assertTrue(DataUtils.isMesmaData(locacao.getDataLocacao(), new Date()));
    Assert.assertTrue(DataUtils.isMesmaData(locacao.getDataRetorno(), DataUtils.adicionarDias(new Date(), 1)));

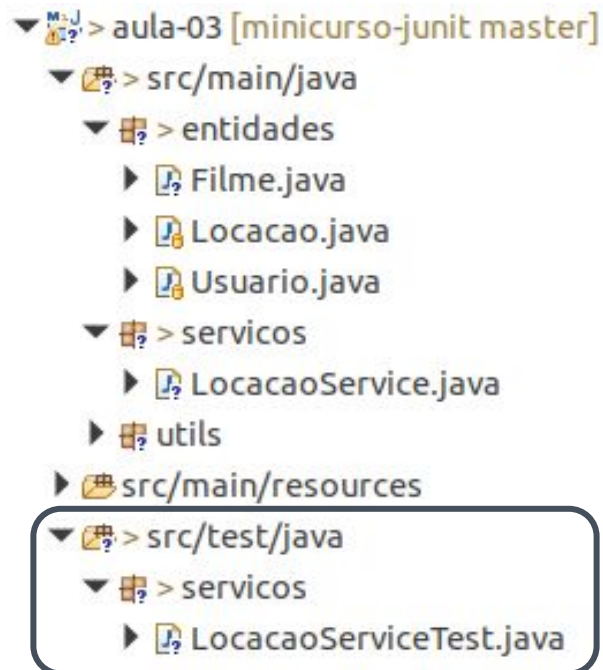
}
```

<https://github.com/francieleap/minicurso-junit/tree/master/Aula-02>

Aula-03: Organização dos arquivos de teste

Convenção:

- Uma classe de teste por classe a ser testada.
- Nome da classe de teste = NomeDaClasse + Test



```
4 import java.util.Date;
5
6 import org.junit.Assert;
7 import org.junit.Test;
8
9 import entidades.Filme;
10 import entidades.Locacao;
11 import entidades.Usuario;
12 import servicos.LocacaoService;
13 import utils.DataUtils;
14
15 public class LocacaoServiceTest {
16
17     @Test
18     public void deveFazerUmaLocacao() {
19
20         //Cenário
21     }
```

Aula-04: Trabalhando com Assertivas



Alguns exemplos de assertivas:

- `AssertTrue(condicao);`
- `AssertFalse(condicao);`
- `AssertEquals(valor esperado, valor atual);`
- `AssertNotEquals(valor esperado, valor atual);`
- `AssertArrayEquals(array esperado, array atual);`
- `AssertNull(objeto);`
- `AssertNotNull(objeto);`
- `AssertSame(objeto esperado, objeto atual);`
- `AssertNotSame(objeto esperado, objeto atual);`
- `AssertThat(atual, matcher);`
- `Fail();`

Aula-04: Trabalhando com Assertivas

Exemplos:

```
@Test
public void testeAssertivas() {

    Assert.assertTrue(1 == 1 );
    Assert.assertFalse(1 == 2);

    Assert.assertEquals(1, 1);
    Assert.assertEquals("teste", "teste");

    int numerosPares[] = new int[3];
    numerosPares[0] = 2;
    numerosPares[1] = 4;
    numerosPares[2] = 6;

    int copiaNumerosPares[] = new int[3];
    copiaNumerosPares[0] = 2;
    copiaNumerosPares[1] = 4;
    copiaNumerosPares[2] = 6;

    Assert.assertArrayEquals(numerosPares, copiaNumerosPares);

    Assert.assertEquals(Math.PI, 3.14, 0.01);
```

Aula-04: Trabalhando com Assertivas

Exemplos:

```
//Trabalhando com objetos
```

```
Usuario usuario1 = new Usuario("Usuario 1");  
Usuario usuario2 = new Usuario("Usuario 1");
```

```
Assert.assertEquals(usuario1, usuario2);
```

**Qual o resultado
desse teste ?**

SUCESSO OU ERRO

Aula-04: Trabalhando com Assertivas

Exemplos:

```
//Trabalhando com objetos
```

```
Usuario usuario1 = new Usuario("Usuario 1");  
Usuario usuario2 = new Usuario("Usuario 1");
```

```
Assert.assertEquals(usuario1, usuario2);
```

Qual o resultado
desse teste ?

SUCESSO OU ERRO

A diagram consisting of six arrows pointing towards the word "ERRO" in the text "SUCESSO OU ERRO". The arrows originate from the top, bottom, and sides, converging on the word "ERRO".

Como não foi implementado o método **equals** na classe Usuario o assertEquals compara os objetos a nível de **instância**.

Aula-04: Trabalhando com Assertivas

Exemplos:

```
21 @Override
22 public boolean equals(Object obj) {
23     if (this == obj)
24         return true;
25     if (obj == null)
26         return false;
27     if (getClass() != obj.getClass())
28         return false;
29     Usuario other = (Usuario) obj;
30     if (nome == null) {
31         if (other.nome != null)
32             return false;
33     } else if (!nome.equals(other.nome))
34         return false;
35     return true;
36 }
37
```

Solução

Aula-04: Trabalhando com Assertivas



Exemplos:

```
//Trabalhando com objetos  
  
Usuario usuario1 = new Usuario("Usuario 1");  
Usuario usuario2 = new Usuario("Usuario 1");  
  
Assert.assertEquals(usuario1, usuario2);  
  
Usuario usuario3 = usuario2;  
  
Assert.assertSame(usuario2, usuario3);  
  
Usuario usuario4 = null;  
  
Assert.assertNull(usuario4);
```

Aula-04: Trabalhando com Assertivas

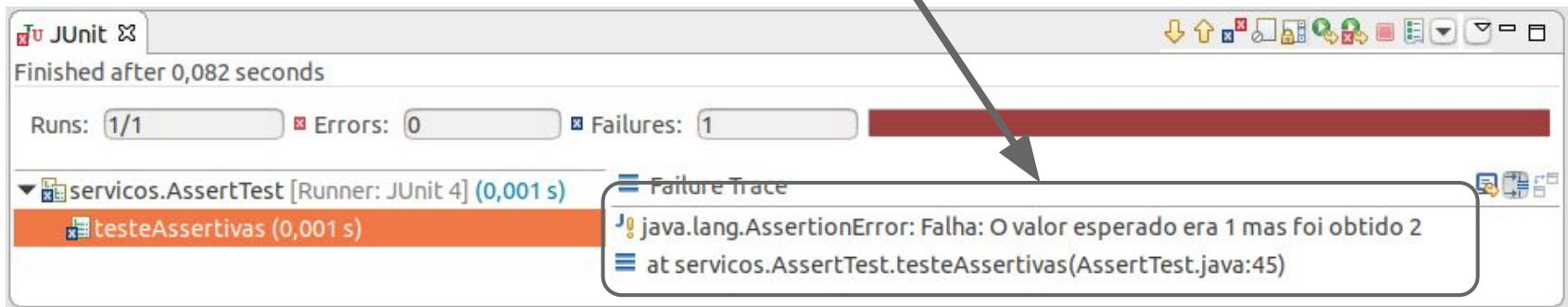
Exemplos:

```
//Customizando mensagem de exceção
```

```
int a = 1; int b=2;
```

```
String mensagem = "Falha: O valor esperado era %d mas foi obtido %d";
```

```
Assert.assertTrue(String.format(mensagem, a, b), a == b);
```



Aula-04: Trabalhando com Assertivas



Exemplos:

```
//Trabalhando com fail()

try {
    // faz um teste que deveria dar exception...
    Assert.assertTrue((2/0) == 1);
    Assert.fail();

} catch (Exception e) {
    Assert.assertTrue(true);
}
```

A ideia do **fail()** é ser usado para interromper a execução quando a linha em que ele é usado jamais deveria ter sido alcançada.

<https://github.com/francieleap/minicurso-junit/tree/master/Aula-04>

Aula-05: Trabalhando com AssertThat e Hamcrest



- O método **AssertThat**(valor atual, matcher) oferece uma maneira melhor de escrever asserções.
- O **Hamcrest** é um framework que possibilita a criação de regras de verificação(**matchers**) de forma declarativa.
- Um matcher Hamcrest é um objeto que:
 - Reporta se um dado objeto satisfaz um determinado critério;
 - Pode descrever este critério; e
 - É capaz de descrever porque um objeto não satisfaz um determinado critério.

Aula-05: Trabalhando com AssertThat e Hamcrest

Porque usar AssertThat?

Legibilidade:

```
assertThat(actual, is(equalTo(expected)));
```

Melhores mensagens de erro:

```
assertThat(actual, containsString(expected));  
java.lang.AssertionError:  
Expected: a string containing "abc" got: "def"
```

Tipo de segurança:

```
assertEquals("abc", 123);  
//compila mais falha
```

```
assertThat(123, is("abc"));  
//não compila
```

Flexibilidade:

```
assertThat("test", anyOf(is("test2"), containsString("te")));  
  
assertThat("test", anyOf(is("test2"), containsString("ca")));  
java.lang.AssertionError: Expected: (is "test2" or a string containing "ca") got:  
"test"
```

Aula-05: Trabalhando com AssertThat e Hamcrest



Alguns exemplos de matchers:

- `CoreMatchers.is();`
- `CoreMatchers.any();`
- `CoreMatchers.describeAs();`
- `CoreMatchers.allOf();`
- `CoreMatchers.anyOf();`
- `CoreMatchers.not();`
- `CoreMatchers.equalTo();`
- `CoreMatchers.instanceOf();`
- `CoreMatchers.notNullValue();`
- `CoreMatchers.nullValue();`
- `CoreMatchers.sameInstance();`

<https://github.com/francieleap/minicurso-junit/tree/master/Aula-05>

Aula-05: Trabalhando com AssertThat e Hamcrest

Exemplos:

```
@Test
public void testeAssertivaThat() {

    Assert.assertThat("123", is("123"));

    Assert.assertThat(123, any(Integer.class));

    Assert.assertThat(123, describedAs("Inteiro igual a %0", equalTo(123), 123));

    Assert.assertThat("123", allOf(isA(String.class), equalTo("123")));

    Assert.assertThat("123", anyOf(isA(String.class), equalTo("111")));

    Usuario usuario = new Usuario();

    Assert.assertThat(usuario, instanceof(Usuario.class));

    String texto = "texto";
    Assert.assertThat(texto, notNullValue(String.class));

    Assert.assertThat(usuario, sameInstance(usuario));
}
```

Aula-05: Trabalhando com AssertThat e Hamcrest

Exemplos:

```
3 import org.hamcrest.Description;
4 import org.hamcrest.TypeSafeMatcher;
5
6 public class CustomMatcher extends TypeSafeMatcher<String> {
7
8     private String letter;
9
10    private CustomMatcher(String letter) {
11        this.letter = letter;
12    }
13
14    public void describeTo(Description description) {
15        description.appendValue("Esperava uma palavra que começa com " + this.letter);
16    }
17
18    @Override
19    protected boolean matchesSafely(String item) {
20        String letra = String.valueOf(item.charAt(0));
21        return letra.equals(this.letter);
22    }
23
24    public static CustomMatcher startWithLetter(String letter) {
25        return new CustomMatcher(letter);
26    }
27 }

```



```
43 //Customizando matchers
44 Assert.assertThat("Aluno", CustomMatcher.startWithLetter("A"));
```


Aula-06: Formas de dividir um teste

Convenção:

- Uma assertiva para cada método de teste. Dessa forma o teste não vai parar caso algum dê erro.

```
@Test
public void deveChecarValorLocacao() {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 10, 5.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertTrue(locacao.getValor() == 5);
}
```

```
@Test
public void deveChecarDataLocacao() {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 10, 5.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertTrue(DataUtils.
        isMesmaData(locacao.getDataLocacao(), new Date()));
}
```

Aula-07: Tratamento de exceções

Nova regra:

- Não deve alugar filme sem estoque.

```
public Locacao alugarFilme(Usuario usuario, Filme filme) throws Exception {  
    //Validação filme sem estoque  
  
    if (filme.getEstoque() == 0) {  
        throw new Exception("Filme sem estoque.");  
    }  
  
    Locacao locacao = new Locacao();  
    locacao.setFilme(filme);  
    locacao.setUsuario(usuario);  
    locacao.setDataLocacao(new Date());  
    locacao.setValor(filme.getPrecoLocacao());  
  
    //Entrega no dia seguinte  
    Date dataEntrega = new Date();  
    dataEntrega = adicionarDias(dataEntrega, 1);  
    locacao.setDataRetorno(dataEntrega);  
  
    return locacao;  
}
```

```
@Test  
public void deveChecarValorLocacao() {  
  
    //Cenário  
    LocacaoService service = new LocacaoService();  
    Usuario usuario = new Usuario("Usuário 01");  
    Filme filme = new Filme("Filme", 10, 5.0);  
  
    //Ação  
    Locacao locacao;  
    try {  
        locacao = service.alugarFilme(usuario, filme);  
  
        //Verificação  
        Assert.assertTrue(locacao.getValor() == 5);  
    } catch (Exception e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

Aula-07: Tratamento de exceções

Exemplos:

```
@Test
public void deveChecarValorLocacao() {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 0, 5.0);

    //Ação
    Locacao locacao;
    try {
        locacao = service.alugarFilme(usuario, filme);

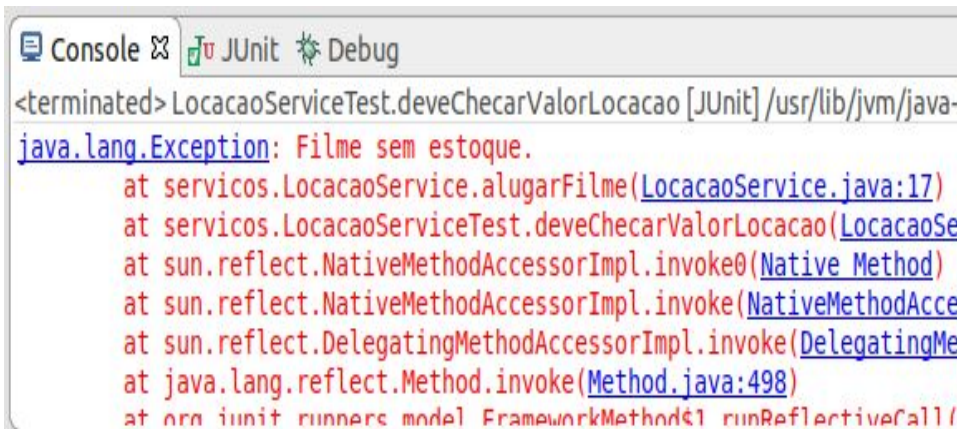
        //Verificação
        Assert.assertTrue(locacao.getValor() == 5);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Qual o resultado
desse teste ao
zerar o estoque?

SUCESSO OU ERRO

Aula-07: Tratamento de exceções

Exemplos:

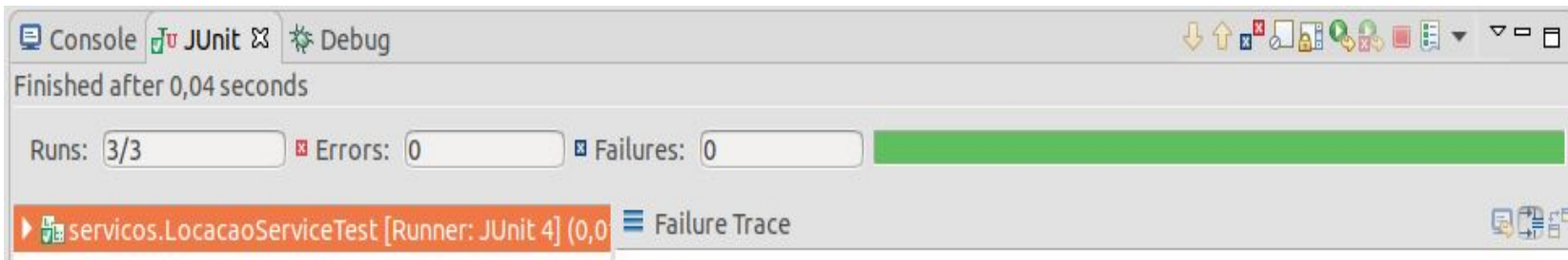


Console JUnit Debug

```
<terminated> LocacaoServiceTest.deveChecarValorLocacao [JUnit] /usr/lib/jvm/java-  
java.lang.Exception: Filme sem estoque.  
    at servicos.LocacaoService.alugarFilme(LocacaoService.java:17)  
    at servicos.LocacaoServiceTest.deveChecarValorLocacao(LocacaoSe  
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAcce  
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMe  
    at java.lang.reflect.Method.invoke(Method.java:498)  
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall()
```

Qual o resultado
desse teste ao
zerar o estoque?

SUCESSO OU ERRO



Console JUnit Debug

Finished after 0,04 seconds

Runs: 3/3 Errors: 0 Failures: 0

servicos.LocacaoServiceTest [Runner: JUnit 4] (0,0 Failure Trace

Aula-07: Tratamento de exceções

Exemplos:

```
//Tratamento exceção não esperada
@Test
public void deveChecarValorLocacao() {
    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 0, 5.0);

    //Ação
    Locacao locacao;
    try {
        locacao = service.alugarFilme(usuario, filme);

        //Verificação
        Assert.assertTrue(locacao.getValor() == 5);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        Assert.fail();
    }
}
```

Exceção não esperada!

Solução 1

Aula-07: Tratamento de exceções

Exemplos:

```
//Tratamento de exceção não esperada.
@Test
public void deveChecarValorLocacao() throws Exception {

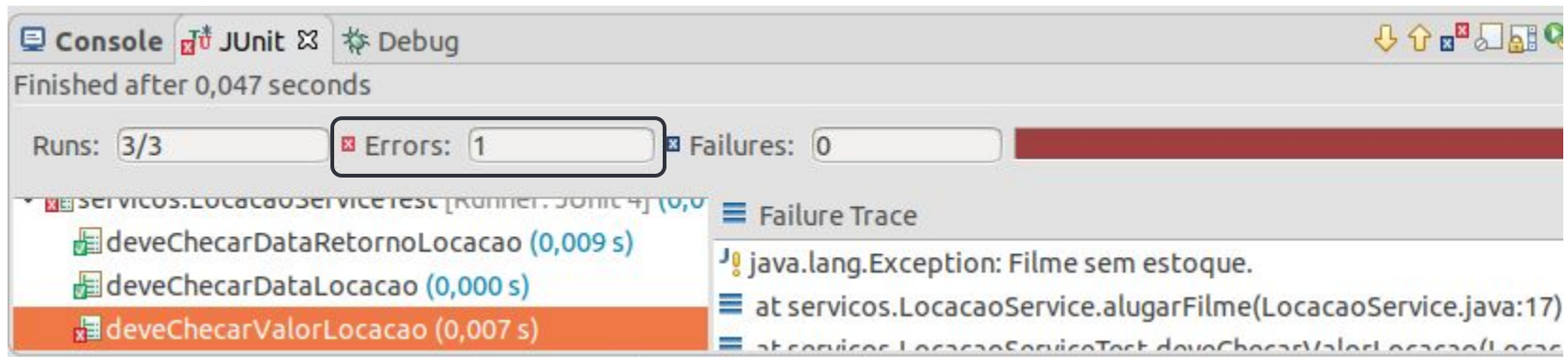
    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 0, 5.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertTrue(locacao.getValor() == 5);
}
```

Exceção não esperada!

Solução 2



Aula-07: Tratamento de exceções

Exemplos:

```
//Tratamento de exceção esperada.
@Test(expected=Exception.class)
public void deveChecarFilmeSemEstoque() throws Exception {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 0, 5.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertTrue(locacao.getValor() == 5);
}
```

Exceção esperada!

Solução 1



Aula-07: Tratamento de exceções

Exemplos:

Exceção esperada!

```
//Tratamento de exceção esperada.
@Test
public void deveChecarFilmeSemEstoque_() {

    //Cenário
    LocacaoService service = new LocacaoService();
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 0, 5.0);

    //Ação
    Locacao locacao;
    try {
        locacao = service.alugarFilme(usuario, filme);

        //Verificação
        Assert.assertTrue(locacao.getValor() == 5);

        Assert.fail("Deveria ter lançado uma exceção!");
    } catch (Exception e) {

        Assert.assertThat(e.getMessage(), CoreMatchers.is("Filme sem estoque."));
    }
}
```

Solução 2

Aula-08: Usando as anotações Before e After

Exemplos:

```
public class LocacaoServiceTest {  
  
    LocacaoService service ;  
  
    @Before  
    public void inicializa() {  
        System.out.println("@Before");  
        service = new LocacaoService();  
    }  
  
    @After  
    public void encerra() {  
        System.out.println("@After");  
    }  
  
    @BeforeClass  
    public static void inicializaClasse() {  
        System.out.println("@BeforeClass");  
    }  
  
    @AfterClass  
    public static void encerraClasse() {  
        System.out.println("@AfterClass!");  
    }  
}
```

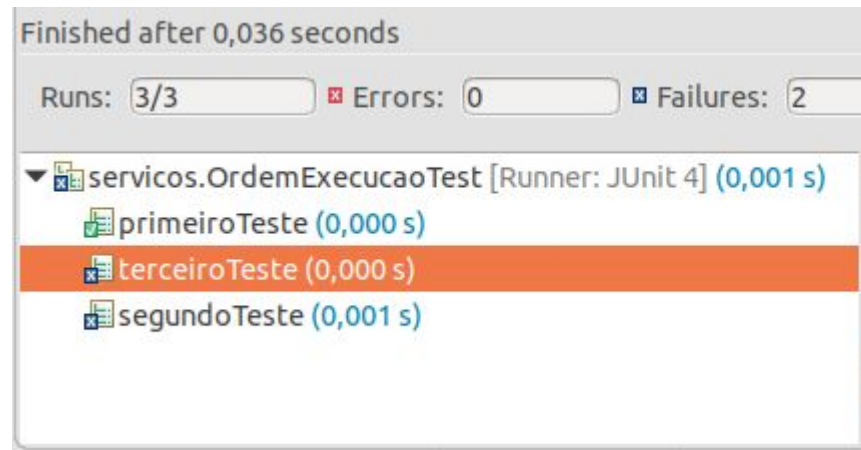


Console JUnit Debug
<terminated> LocacaoServiceTest (5)
@BeforeClass
@Before
@After
@Before
@After
@Before
@After
@Before
@After
@Before
@After
@AfterClass!

Aula-09: Ordem de execução dos testes

Exemplos:

```
7 public class OrdemExecucaoTest {
8
9     public static int contador = 0;
10
11     @Test
12     public void primeiroTeste() {
13         contador = contador + 1;
14         Assert.assertThat(contador, CoreMatchers.is(1));
15     }
16
17     @Test
18     public void segundoTeste() {
19         contador = contador + 1;
20         Assert.assertThat(contador, CoreMatchers.is(2));
21     }
22
23     @Test
24     public void terceiroTeste() {
25         contador = contador + 1;
26         Assert.assertThat(contador, CoreMatchers.is(3));
27     }
28
29 }
```



Aula-09: Ordem de execução dos testes

Exemplos:

```
9 @FixMethodOrder(MethodSorters.NAME_ASCENDING)
10 public class OrdemExecucaoTest {
11
12     public static int contador = 0;
13
14     @Test
15     public void teste1() {
16         contador = contador + 1;
17         Assert.assertThat(contador, CoreMatchers.is(1));
18     }
19
20     @Test
21     public void teste2() {
22         contador = contador + 1;
23         Assert.assertThat(contador, CoreMatchers.is(2));
24     }
25
26     @Test
27     public void teste3() {
28         contador = contador + 1;
29         Assert.assertThat(contador, CoreMatchers.is(3));
30     }
31
32 }
```

Solução

@FixMethodOrder
(MethodSorters.NAME_ASCENDING)

Finished after 0,029 seconds

Runs: 3/3 Errors: 0 Failures: 0

▼ servicicos.OrdemExecucaoTest [Runner: JUnit 4] (0,000 s) Failure Tr

- teste1 (0,000 s)
- teste2 (0,000 s)
- teste3 (0,000 s)

4.

Introdução à TDD

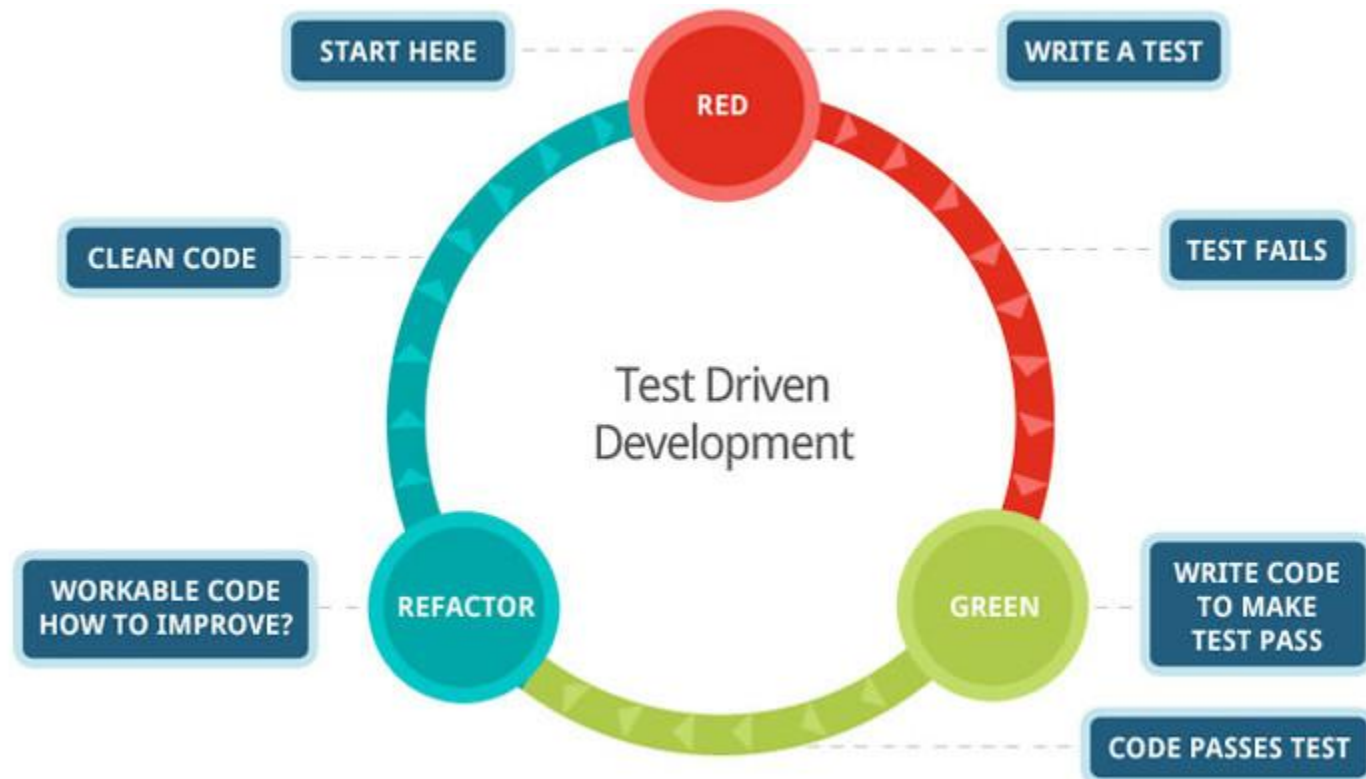
Introdução à TDD



- **Test Driven Development (TDD)** ou desenvolvimento guiado por testes é uma técnica de desenvolvimento de software que se relaciona com o conceito de **verificação** e **validação**.
- A ideia é bem simples: escreva os testes antes mesmo de escrever o código de produção.

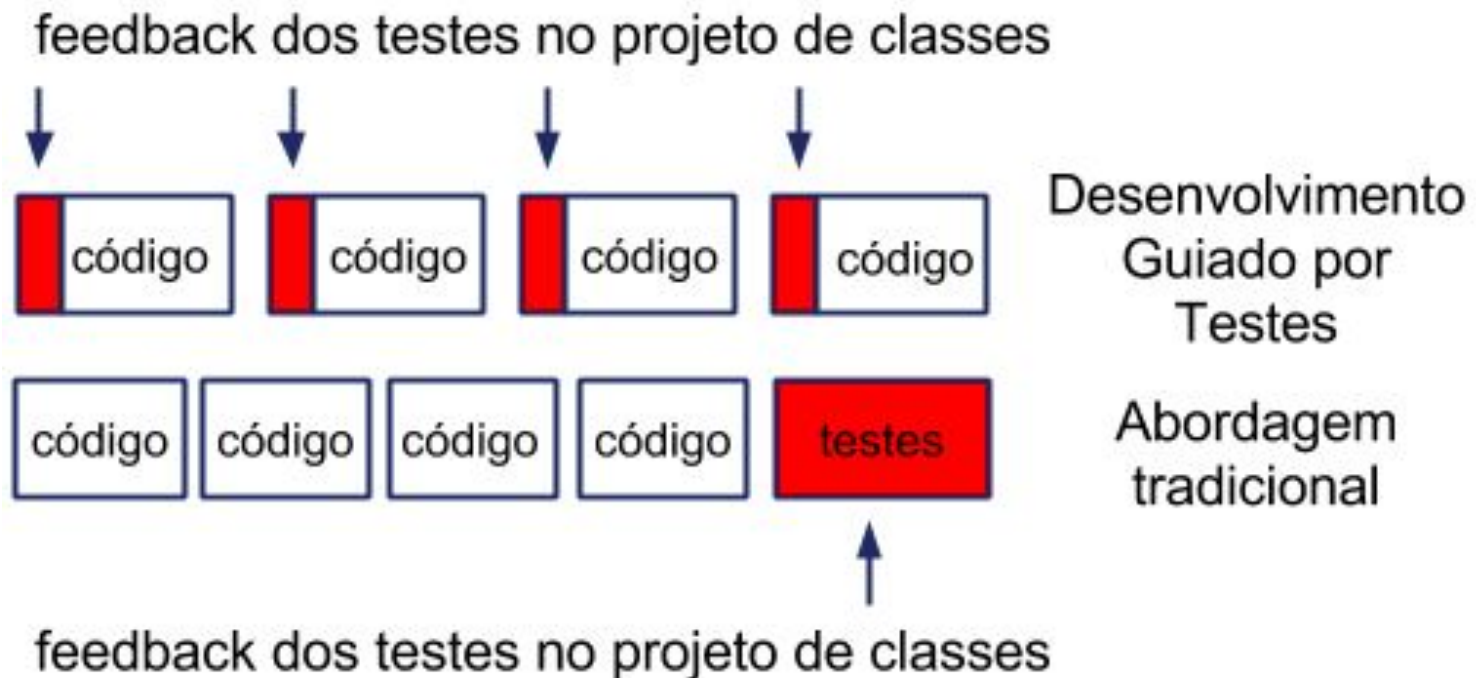
Introdução à TDD

- O ciclo do TDD é conhecido como: RED-GREEN-REFACTOR.



Introdução à TDD

- Qual a diferença entre fazer TDD e escrever o teste depois ?



Introdução à TDD



■ Alguns desafios:

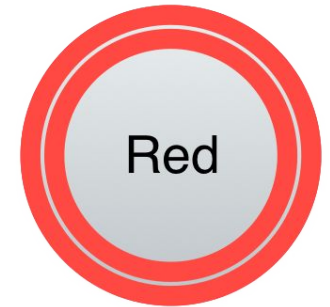
- Resistência do programador em adotar esta prática.
- A curva de aprendizagem é um pouco extensa.
- Para linguagens que não possuam frameworks o TDD pode se tornar pesado e difícil.
- O TDD é difícil de ser implementado em códigos legados.

**Vamos para o
código ...**

Aula-10: Test Driven Development - TDD



Exemplo: Vamos criar a CalculadoraTest.



```
7 public class CalculadoraTest {
8
9     @Test
10     public void deveSomarDoisNumeros() {
11         //cenário
12         int a = 5;
13         int b = 3;
14
15         Calculadora calculadora = new Calculadora();
16
17         //ação
18         int resultado = calculadora.somar(a,b);
19
20         //verificacao
21         Assert.assertThat(resultado, CoreMatchers.is(8));
22
23     }
24 }
```

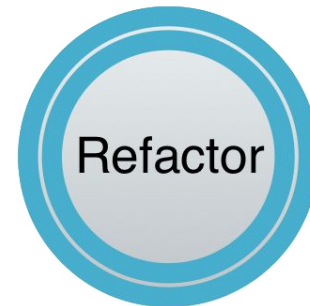
Aula-10: Test Driven Development - TDD

Exemplo: Vamos criar a Calculadora.



```
3 public class Calculadora {  
4  
5     public int somar(int a, int b) {  
6  
7         return a + b;  
8     }  
9  
10 }
```

Aula-10: Test Driven Development - TDD



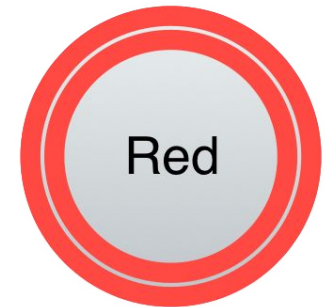
Exemplo: Vamos refatorar o código.

```
8 public class CalculadoraTest {
9
10     Calculadora calculadora;
11
12     @Before
13     public void inicializa() {
14         calculadora = new Calculadora();
15     }
16
17     @Test
18     public void deveSomarDoisNumeros() {
19         //cenário
20         int a = 5;
21         int b = 3;
22
23         //ação
24         int resultado = calculadora.somar(a,b);
25
26         //verificacao
27         Assert.assertThat(resultado, CoreMatchers.is(8));
28     }
29 }
30 }
```

Aula-10: Test Driven Development - TDD

Nova regra:

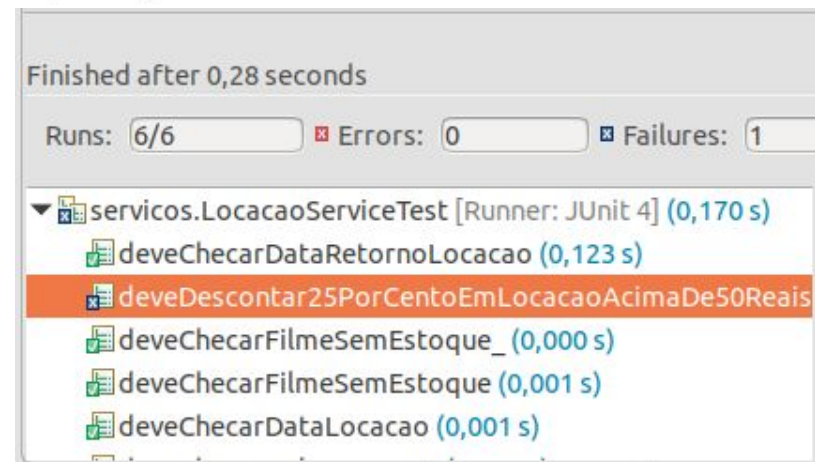
- Locação acima de 50 reais deve receber 25% de desconto.



```
@Test
public void deveDescontar25PorCentoEmLocacaoAcimaDe50Reais() throws Exception {
    //Cenário
    Usuario usuario = new Usuario("Usuário 01");
    Filme filme = new Filme("Filme", 10, 60.0);

    //Ação
    Locacao locacao = service.alugarFilme(usuario, filme);

    //Verificação
    Assert.assertThat(locacao.getValor(), CoreMatchers.is(45.0));
}
```



Aula-10: Test Driven Development - TDD



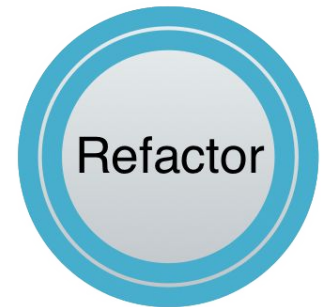
```
public Locacao alugarFilme(Usuario usuario, Filme filme) throws Exception {  
    //Validação filme sem estoque  
  
    if (filme.getEstoque() == 0) {  
        throw new Exception("Filme sem estoque.");  
    }  
  
    Locacao locacao = new Locacao();  
    locacao.setFilme(filme);  
    locacao.setUsuario(usuario);  
    locacao.setDataLocacao(new Date());  
  
    if (filme.getPrecoLocacao() > 50.0) {  
        Double desconto = filme.getPrecoLocacao() * 0.25;  
        locacao.setValor(filme.getPrecoLocacao() - desconto);  
    } else {  
        locacao.setValor(filme.getPrecoLocacao());  
    }  
  
    //Entrega no dia seguinte  
    Date dataEntrega = new Date();  
    dataEntrega = adicionarDias(dataEntrega, 1);  
    locacao.setDataRetorno(dataEntrega);  
  
    return locacao;  
}
```

Finished after 0,117 seconds

Runs: 6/6 Errors: 0 Failures: 0

▼ servicos.LocacaoServiceTest [Runner: JUnit 4] (0,001 s)
 deveChecarDataRetornoLocacao (0,000 s)
 deveDescontar25PorCentoEmLocacaoAcimaDe50Reais
 deveChecarFilmeSemEstoque_ (0,000 s)
 deveChecarFilmeSemEstoque (0,001 s)
 deveChecarDataLocacao (0,000 s)

Aula-10: Test Driven Development - TDD



```
public Locacao alugarFilme(Usuario usuario, Filme filme) throws Exception {
    //Validação filme sem estoque

    if (filme.getEstoque() == 0) {
        throw new Exception("Filme sem estoque.");
    }

    Locacao locacao = new Locacao();
    locacao.setFilme(filme);
    locacao.setUsuario(usuario);
    locacao.setDataLocacao(new Date());
    locacao.setValor(calculaValorLocacao(filme.getPrecoLocacao()));

    //Entrega no dia seguinte
    Date dataEntrega = new Date();
    dataEntrega = adicionarDias(dataEntrega, 1);
    locacao.setDataRetorno(dataEntrega);

    return locacao;
}

private Double calculaValorLocacao(Double precoFilme) {

    if (precoFilme > 50) {
        Double desconto = precoFilme * 0.25;
        return precoFilme-desconto;
    }

    return precoFilme;
}
```

Finished after 0,117 seconds

Runs: 6/6 Errors: 0 Failures: 0

▼ servicos.LocacaoServiceTest [Runner: JUnit 4] (0,001 s)

- deveChecarDataRetornoLocacao (0,000 s)
- deveDescontar25PorCentoEmLocacaoAcimaDe50Reais**
- deveChecarFilmeSemEstoque_ (0,000 s)
- deveChecarFilmeSemEstoque (0,001 s)
- deveChecarDataLocacao (0,000 s)

5.

Usando Data Builders

Usando Data Builders



- **Builder**, é um **padrão de projeto** de software criacional que permite a separação da construção de um objeto complexo da sua representação.
- Este padrão permite que o mesmo processo de construção do objeto possa criar **diferentes representações**.
- No contexto de dados, **Data Builders**, o padrão é usado para criar dados de teste de forma automatizada que facilitam a leitura dos testes de unidade.

Usando Data Builders



- O padrão pode ser implementado da seguinte maneira:
 1. Para cada classe de domínio, cria-se uma **classe Builder correspondente**.
 2. No construtor da classe Builder, **inicialize** cada propriedade com um **valor default**.
 3. Para cada atributo da classe adiciona-se um **método precedido com With ou Com**, em português, que altere a propriedade e retorne o próprio Builder.
 4. E por fim, adicione um **método build()** que retorne uma nova instância da classe de domínio com os valores passados.

Usando Data Builders

■ Exemplo:

```
User aUser = new User();  
aUser.setName("John");  
aUser.setPassword("42abc");
```



```
User aUser = UserBuiler.aUser()  
    .withName("John")  
    .withPassword("42abc")  
    .build();
```

**Vamos para o
código ...**

Aula-11: Usando Data Builders

Exemplo:

```
1 package builders;                                     //Cenário
2                                                         Usuario usuario = UsuarioBuilder.umUsuario().build();
3 import entidades.Usuario;
4
5 public class UsuarioBuilder {
6
7     private Usuario usuario;
8
9     private UsuarioBuilder() {};
10
11     public static UsuarioBuilder umUsuario() {
12
13         UsuarioBuilder builder = new UsuarioBuilder();
14         builder.usuario = new Usuario();
15         builder.usuario.setNome("Usuario 1");
16         return builder;
17     }
18
19     public Usuario build() {
20         return usuario;
21     }
22 }
```

Aula-11: Usando Data Builders

Exemplo:

```
5 public class FilmeBuilder {
6
7     private Filme filme;
8     private FilmeBuilder() {};
9
10    public static FilmeBuilder umFilme() {
11
12        FilmeBuilder builder = new FilmeBuilder();
13        builder.filme = new Filme();
14        builder.filme.setNome("A freira");
15        builder.filme.setEstoque(10);
16        builder.filme.setPrecoLocacao(5.0);
17        return builder;
18    }
19
20    public FilmeBuilder comEstoque(Integer valor) {
21        filme.setEstoque(valor);
22        return this;
23    }
24
25    public FilmeBuilder comPrecoLocacao(Double valor) {
26        filme.setPrecoLocacao(valor);
27        return this;
28    }
29
30    public Filme build() {
31        return filme;
32    }
33 }
```

//Cenário

```
Filme filme = FilmeBuilder.umFilme().comEstoque(0).build();
```

Aula-11: Usando Data Builders

Exemplo:

//Cenário

Locacao locacao = LocacaoBuilder.umaLocacao().build();

```
8 public class LocacaoBuilder {
9
10     private Locacao locacao;
11     private LocacaoBuilder() {};
12
13     public static LocacaoBuilder umaLocacao() {
14
15         LocacaoBuilder builder = new LocacaoBuilder();
16         builder.locacao.setDataLocacao(new Date());
17         builder.locacao.setDataRetorno(DataUtils.adicionarDias(new Date(), 7));
18         builder.locacao.setUsuario(UsuarioBuilder.umUsuario().build());
19         builder.locacao.setFilme(FilmeBuilder.umFilme().build());
20         builder.locacao.setValor(30.00);
21         return builder;
22     }
23
24     public LocacaoBuilder comDataRetorno(Date data) {
25         locacao.setDataRetorno(data);
26         return this;
27     }
28
29     public Locacao build() {
30         return locacao;
31     }
32
33 }
```

6.

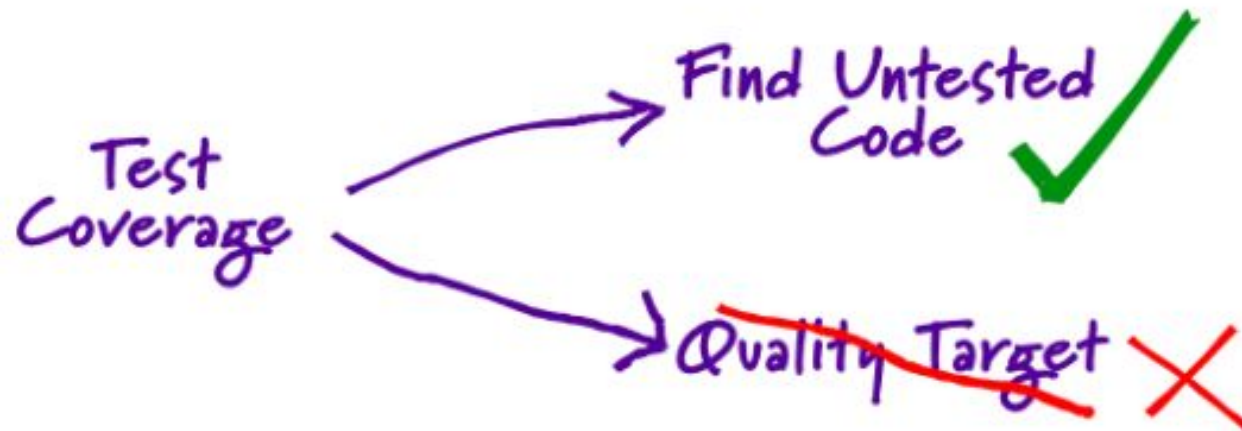
Análise de Cobertura

Análise de Cobertura

- A **Cobertura** dos testes é uma métrica que mede, em um conjunto de itens a serem testados, a porção que foi realmente testada.
- É a medida de abrangência do teste, indicando o nível de confiança atribuído ao teste.
- Esta métrica permite verificar se todas as funcionalidades do sistema estão sendo testadas, ou seja, se o **teste cobre todas as funcionalidades**.

Análise de Cobertura

- Vale ressaltar que:



Análise de Cobertura



■ Métricas:

Percentual de aceitação dos testes = n° de testes executados com sucesso / n° de testes executados.

Percentual de cobertura dos testes = n° de funcionalidades cobertas pelo teste / n° de funcionalidades no sistema

**Vamos para o
código ...**

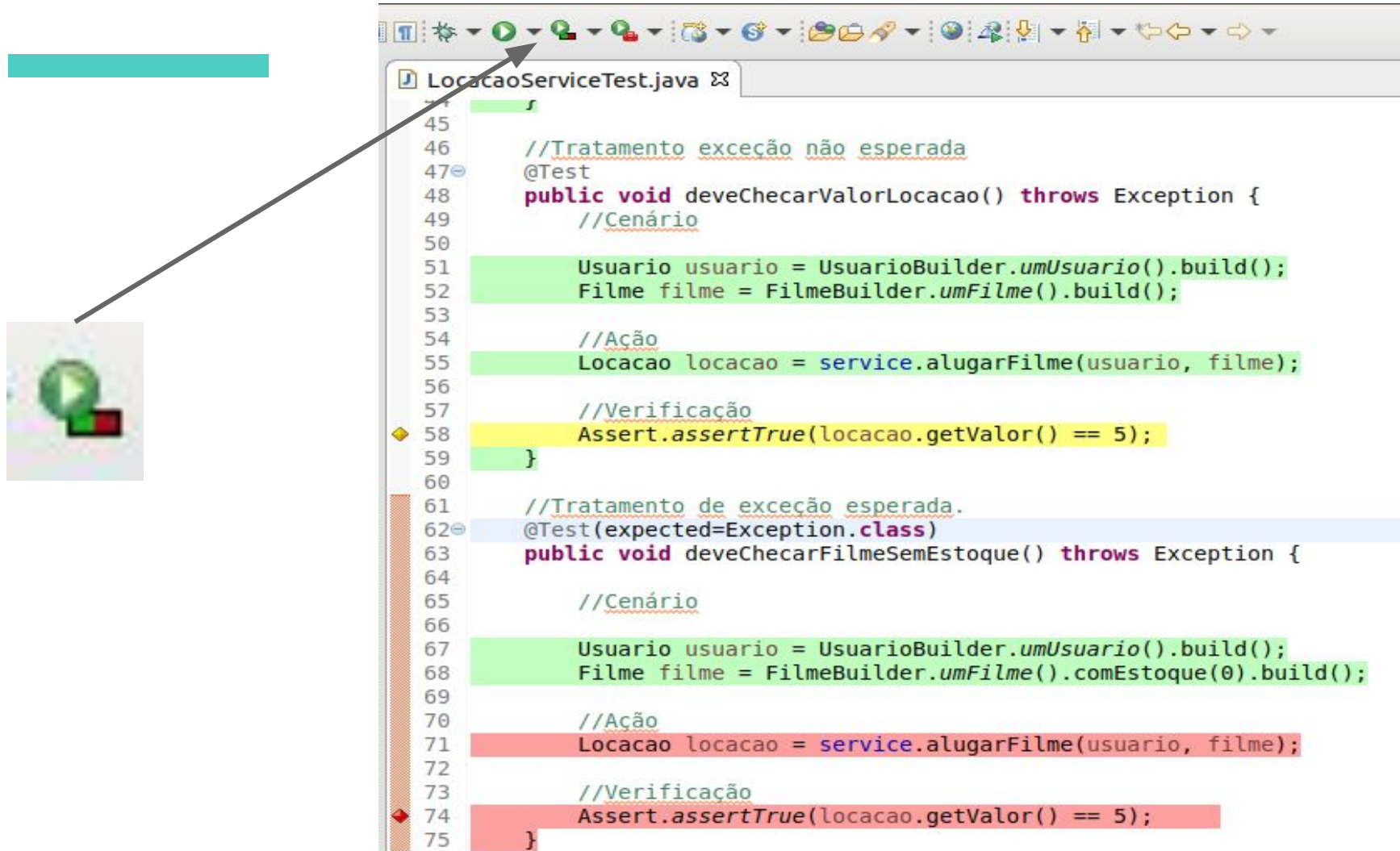
Aula-12: Análise de Cobertura

■ Instalar plugin:

- Help > Eclipse Marketplace > Search EclEmma



Aula-12: Análise de Cobertura



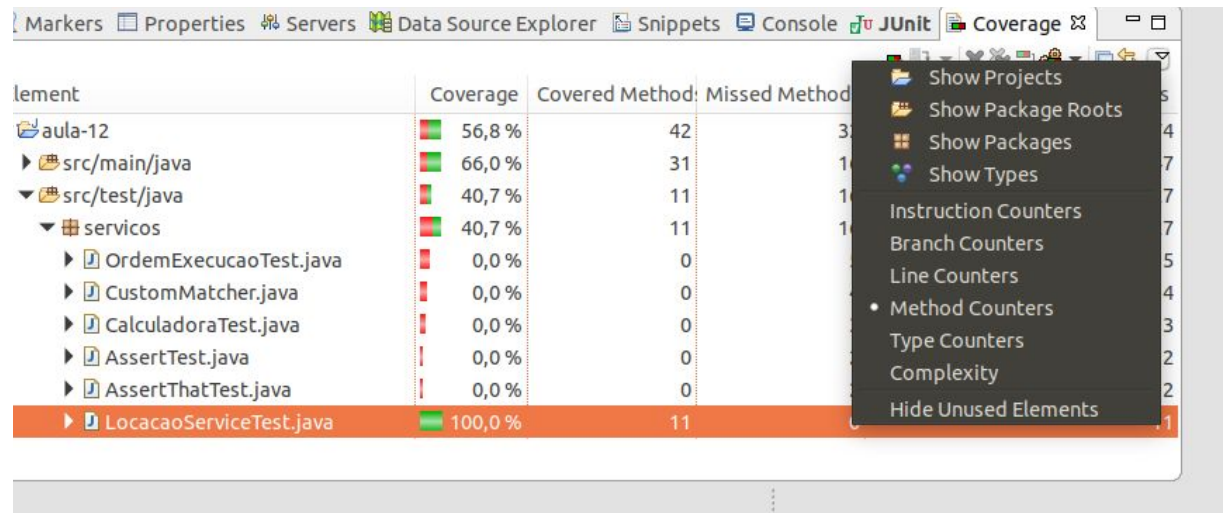
Aula-12: Análise de Cobertura



- As **linhas** de código executável recebem a cor:
 - Verde: para linhas totalmente cobertas.
 - Amarelo: para linhas parcialmente cobertas
 - Vermelho: para linhas que não foram executadas.
- Os **diamantes** recebem a cor:
 - Verde: para branches totalmente cobertos.
 - Amarelo: para branches parcialmente cobertos.
 - Vermelho: quando nenhuma branch na linha em particular foi executado.

Aula-12: Análise de Cobertura

- Contadores para calcular as métricas de cobertura.
 - Instruções
 - Branches
 - Complexidade
 - Linhas
 - Métodos
 - Classes



element	Coverage	Covered Method	Missed Method
aula-12	56,8 %	42	3
src/main/java	66,0 %	31	1
src/test/java	40,7 %	11	1
services	40,7 %	11	1
OrdemExecucaoTest.java	0,0 %	0	0
CustomMatcher.java	0,0 %	0	0
CalculadoraTest.java	0,0 %	0	0
AssertTest.java	0,0 %	0	0
AssertThatTest.java	0,0 %	0	0
LocacaoServiceTest.java	100,0 %	11	0

Aula-12: Análise de Cobertura

- Cobertura de teste **não deve ser usada para medir a qualidade do código** e sim para verificar quais partes do código ainda não foram cobertas.
- Ao atribuir letras às variáveis “a” e “b” o método de divisão irá falhar.

```
1 package servicos;
2
3 public class Calculadora {
4
5     public int somar(int a, int b) {
6
7         return a + b;
8     }
9
10    public int divide(String a, String b) {
11        return Integer.valueOf(a)/Integer.valueOf(b);
12    }
13
14 }
```

```
31 @Test
32 public void deveDividirDoisNumeros() {
33     //cenário
34     String a = "9";
35     String b = "3";
36
37     //ação
38     int resultado = calculadora.divide(a, b);
39
40     //verificacao
41     Assert.assertThat(resultado, CoreMatchers.is(3));
42
43 }
```

8.

Conclusão

Conclusão



- Neste minicurso foi apresentado o uso de testes unitários na linguagem Java utilizando Junit.
- Foi apresentado as vantagens do uso do framework bem como o uso do mesmo.
- Foi também apresentada a técnica de desenvolvimento TDD e o padrão Data Builder.
- Por fim foi apresentada a métrica de cobertura de testes.
- Com isso foi possível perceber as vantagens e abrangência dos testes unitários para melhorar a qualidade do software.

Fim ...

Obrigada!

Alguma dúvida?



/franciferreiraap



/franciele-ferreira



/francieleap



/franciferreiraap