



Automação de Testes de Software com JUnit

Prof. Evandro César Freiburger

Instituto Federal de Mato Grosso
Departamento da Área de Informática
evandro.freiberger@ifmt.edu.br

2023

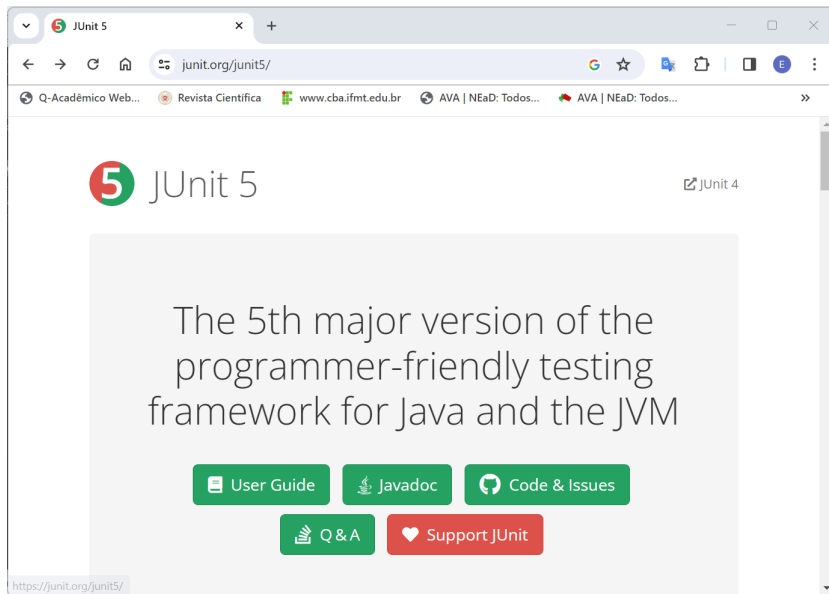
Sumário

- 1 Introdução ao Framework JUnit
- 2 Framework JUnit - Assertions
- 3 Framework JUnit - Padrão de Nomenclatura
- 4 Framework JUnit - Ciclo de Vida dos Casos de Testes
- 5 Framework JUnit - Tratando Exceções no Teste
- 6 Framework JUnit - Comparando Arrays
- 7 Framework JUnit - Teste de Performance

Framework JUnit

- O JUnit é um framework open-source, criado por Erich Gamma e Kent Beck, com suporte à criação de testes automatizados para a linguagem de programação Java.
- Projetado inicialmente para a automação de Testes de Unidade.
- Esse framework facilita a criação e manutenção do código para a automação de testes com apresentação dos resultados.
- Atualmente, com apoio de outras APIs e Framework, pode ser usado para outros níveis de testes, como teste de integração e testes de sistema.
- O JUnit permite a realização de testes de unidades, conhecidos como "caixa branca", facilitando assim a correção de métodos e objetos.

Framework JUnit



Introdução a Um Cenário de Teste

Crie um Projeto Maven (maven-archetype-quickstar), com os seguintes dados:

- **groupId** = ifmt.cba
- **artifactId** = junit01
- **version** = 1.0-SNAPSHOT
- **pasta do projeto**: JUnit01

Arquivo pom.xml do Projeto Maven (1)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>ifmt.cba</groupId>
8   <artifactId>junit01</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <name>junit01</name>
12
13   <properties>
14     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     <java.version>19</java.version>
16     <junit.version>5.10.0</junit.version>
17     <maven.compiler.source>19</maven.compiler.source>
18     <maven.compiler.target>19</maven.compiler.target>
19     <maven.compiler.plugin.version>3.11.0</maven.compiler.plugin.version>
20     <maven-surefire-plugin.version>3.2.1</maven-surefire-plugin.version>
21     <skipTests>>false</skipTests>
22   </properties>
23
24   <dependencies>
25     <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->
26     <dependency>
27       <groupId>org.junit.jupiter</groupId>
28       <artifactId>junit-jupiter</artifactId>
29       <version>${junit.version}</version>
30       <scope>test</scope>
31     </dependency>
```

Arquivo pom.xml do Projeto Maven (2)

```
32 </dependencies>
33
34 <build>
35   <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
36     <plugins>
37       <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
38       <plugin>
39         <artifactId>maven-clean-plugin</artifactId>
40         <version>3.1.0</version>
41       </plugin>
42       <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#
43         Plugin_bindings_for_jar_packaging -->
44       <plugin>
45         <artifactId>maven-resources-plugin</artifactId>
46         <version>3.0.2</version>
47       </plugin>
48       <plugin>
49         <groupId>org.apache.maven.plugins</groupId>
50         <artifactId>maven-compiler-plugin</artifactId>
51         <configuration>
52           <source>${maven.compiler.source}</source>
53           <target>${maven.compiler.target}</target>
54         </configuration>
55         <version>${maven.compiler.plugin.version}</version>
56       </plugin>
57       <plugin>
58         <groupId>org.apache.maven.plugins</groupId>
59         <artifactId>maven-surefire-plugin</artifactId>
60         <version>${maven-surefire-plugin.version}</version>
61         <configuration>
62           <skipTests>${skipTests}</skipTests>
63         </configuration>
```

Arquivo pom.xml do Projeto Maven (3)

```
63     </plugin>
64     <plugin>
65       <artifactId>maven-jar-plugin</artifactId>
66       <version>3.0.2</version>
67     </plugin>
68     <plugin>
69       <artifactId>maven-install-plugin</artifactId>
70       <version>2.5.2</version>
71     </plugin>
72     <plugin>
73       <artifactId>maven-deploy-plugin</artifactId>
74       <version>2.8.2</version>
75     </plugin>
76     <!-- site lifecycle , see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
77     <plugin>
78       <artifactId>maven-site-plugin</artifactId>
79       <version>3.7.1</version>
80     </plugin>
81     <plugin>
82       <artifactId>maven-project-info-reports-plugin</artifactId>
83       <version>3.0.0</version>
84     </plugin>
85   </plugins>
86 </pluginManagement>
87 </build>
88 </project>
```


Plugin SureFire

O plugin **maven-surefire-plugin** é responsável por executar os testes no ciclo de tarefas do maven, quando isso é feito em um console ou por um software de automação de uma esteira de integração contínua.

Por exemplo quando executamos:

- Para executar a etapa de testes: `mvn test`
- Para executar a etapa de empacotamento: `mvn package`
- Para executar uma etapa e pular os testes: `mvn package -DskipTests`

Exemplo de Classe de Teste

J PrimeiroTest.java ×

src > test > java > ifmt > cba > J PrimeiroTest.java > ...

```
1  package ifmt.cba;
2
3  import org.junit.jupiter.api.Test;
4
5  public class PrimeiroTest {
6
7      @Test
8      public void meuMetodoDeTeste1(){
9          System.out.println(x:"Primeiro exemplo de teste, que nao testa nada!!");
10     }
11
12     @Test
13     public void meuMetodoDeTeste2(){
14         System.out.println(x:"Segundo exemplo de teste, que nao testa nada!!");
15     }
16 }
```

Executando a Classe de Teste pela IDE

Informações da execução do teste

The screenshot displays an IDE interface with a Java source file and its test execution results. The source file, `PrimeiroTest.java`, is located at `src > test > java > ifmt > cba > J PrimeiroTest.java > ...`. It contains the following code:

```
1 package ifmt.cba;
2
3 import org.junit.jupiter.api.Test;
4
5 public class PrimeiroTest {
6
7     @Test
8     public void meuMetodoDeTeste1(){
9         System.out.println(x:"Primeiro exemplo de teste, que nao testa nada!!");
10    }
11
12     @Test
13     public void meuMetodoDeTeste2(){
14         System.out.println(x:"Segundo exemplo de teste, que nao testa nada!!");
15    }
16 }
```

Three green boxes highlight the test annotations and methods: `@Test` on line 7, `@Test` on line 12, and the `meuMetodoDeTeste1()` method on line 8. The IDE's bottom panel shows the **TEST RESULTS** tab, which contains the following output:

```
%TESTC 1 v2
%TSTTREE2,ifmt.cba.PrimeiroTest,true,1,false,1,PrimeiroTest,,[engine:junit-jupiter]/[
class:ifmt.cba.PrimeiroTest]
%TSTTREE3,meuMetodoDeTeste1(ifmt.cba.PrimeiroTest),false,1,false,2,meuMetodoDeTeste1(
),,[engine:junit-jupiter]/[class:ifmt.cba.PrimeiroTest]/[method:meuMetodoDeTeste1()]
%TESTS 3,meuMetodoDeTeste1(ifmt.cba.PrimeiroTest)

%TESTE 3,meuMetodoDeTeste1(ifmt.cba.PrimeiroTest)

%RUNTIME139
```

A green box highlights the `%TESTS` line. To the right, a pop-up window shows the test run details: "Test run at 10/30/2023, 8:41:46 AM" and "meuMetodoDeTeste1()", both with green checkmarks indicating success.

Executando a Classe de Teste pela IDE

Informações da execução do teste

```
src > test > java > ifmt > cba > J PrimeiroTest.java > ...
1  package ifmt.cba;
2
3  import org.junit.jupiter.api.Test;
4
5  public class PrimeiroTest {
6
7      @Test
8      public void meuMetodoDeTeste1(){
9          System.out.println(x:"Primeiro exemplo de teste, que nao testa nada!!");
10     }
11
12     @Test
13     public void meuMetodoDeTeste2(){
14         System.out.println(x:"Segundo exemplo de teste, que nao testa nada!!");
15     }
16 }
```

PROBLEMS OUTPUT TEST RESULTS TERMINAL PORTS **DEBUG CONSOLE** Filter (e.g. text, !exclude)

Primeiro exemplo de teste, que nao testa nada!!

Executando a Classe de Teste pela IDE

Informações da execução do teste

The screenshot displays an IDE window titled "PrimeiroTest.java". The code defines a class `PrimeiroTest` with two test methods, `meuMetodoDeTeste1()` and `meuMetodoDeTeste2()`, both annotated with `@Test`. The methods use `System.out.println` to output messages. The IDE interface includes tabs for "PROBLEMS", "OUTPUT", "TEST RESULTS", "TERMINAL", "PORTS", and "DEBUG CONSOLE". The "TEST RESULTS" tab is active, showing a summary of the test run. A green box highlights the class name `PrimeiroTest` in the code editor and the test results summary in the "TEST RESULTS" tab. The summary indicates that the test run was successful at 10:23:10 AM on 10/30/2023, with two test methods passing.

```
1 package ifmt.cba;
2
3 import org.junit.jupiter.api.Test;
4
5 public class PrimeiroTest {
6
7     @Test
8     public void meuMetodoDeTeste1(){
9         System.out.println(x:"Primeiro exemplo de teste, que nao testa nada!!");
10    }
11
12     @Test
13     public void meuMetodoDeTeste2(){
14         System.out.println(x:"Segundo exemplo de teste, que nao testa nada!!");
15    }
16 }
17
```

TEST RESULTS

[engine:junit-jupiter]/[class:ifmt.cba.PrimeiroTest]/[method:meuMetodoDeTeste2()]

%TESTS 3,meuMetodoDeTeste1(ifmt.cba.PrimeiroTest)

%TESTE 3,meuMetodoDeTeste1(ifmt.cba.PrimeiroTest)

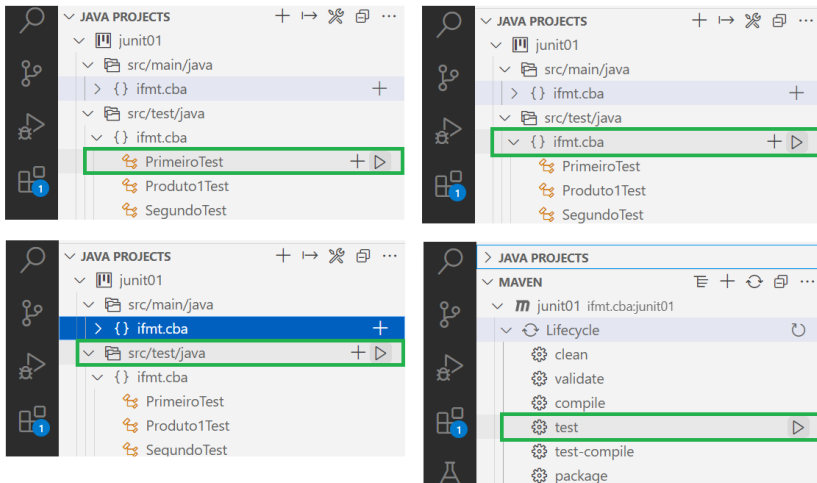
%TESTS 4,meuMetodoDeTeste2(ifmt.cba.PrimeiroTest)

%TESTE 4,meuMetodoDeTeste2(ifmt.cba.PrimeiroTest)

Test run at 10/30/2023, 10:23:10 AM

- meuMetodoDeTeste1()
- meuMetodoDeTeste2()

Executando a Classe de Teste pela IDE



Executando os testes pelo Maven no console

```
1 PS C:\Users\evand\Dropbox\Testes de Software\Implementacao\junit01 > mvn test
2 [INFO] Scanning for projects...
3 [INFO]
4 [INFO] -----< ifmt.cba:junit01 >-----
5 [INFO] Building junit01 1.0-SNAPSHOT
6 [INFO] -----[ jar ]-----
7 [INFO]
8 ----- Texto Removido para Apresentacao -----
9
10 [INFO] --- maven-surefire-plugin:3.2.1:test (default-test) @ junit01 ---
11 [INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
12 [INFO]
13 [INFO] -----
14 [INFO] T E S T S
15 [INFO] -----
16 [INFO] Running ifmt.cba.PrimeiroTest
17 Primeiro exemplo de teste, que nao testa nada!!
18 Segundo exemplo de teste, que nao testa nada!!
19 [INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.050 s -- in ifmt.cba.PrimeiroTest
20 [INFO]
21 [INFO] Results:
22 [INFO]
23 [INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
24 [INFO]
25 [INFO] -----
26 [INFO] BUILD SUCCESS
27 [INFO] -----
28 [INFO] Total time: 2.391 s
29 [INFO] Finished at: 2023-10-30T10:28:57-04:00
30 [INFO] -----
31 PS C:\Users\evand\Dropbox\Testes de Software\Implementacao\junit01 >
```

Pulando os testes pelo Maven no console

```
1 [INFO] Nothing to compile - all classes are up to date
2 PS C:\Users\evand\Dropbox\Testes de Software\Implementacao\junit01 > mvn package -DskipTests
3 [INFO] Scanning for projects...
4 [INFO]
5 [INFO] -----< ifmt.cba:junit01 >-----
6 [INFO] Building junit01 1.0-SNAPSHOT
7 [INFO] -----[ jar ]-----
8 [INFO]
9 -----Texto Excluido para Apresentacao-----
10 [INFO]
11 [INFO] --- maven-surefire-plugin:3.2.1:test (default-test) @ junit01 ---
12 [INFO] Tests are skipped.
13 [INFO]
14 [INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ junit01 ---
15 [INFO] -----
16 [INFO] BUILD SUCCESS
17 [INFO] -----
18 [INFO] Total time: 1.036 s
19 [INFO] Finished at: 2023-10-30T10:50:49-04:00
20 [INFO] -----
21 PS C:\Users\evand\Dropbox\Testes de Software\Implementacao\junit01 >
```


Lançando uma Falha no Teste

```
J SegundoTest.java ●
src > test > java > ifmt > cba > J SegundoTest.java > ...
1  package ifmt.cba;
2
3  import static org.junit.jupiter.api.Assertions.fail;
4
5  import org.junit.jupiter.api.Test;
6
7  public class SegundoTest {
8
9      @Test
10     public void meuMetodoDeTeste3(){
11         System.out.println(x:"Vai falhar!!");
12         fail("Falhou de proposito");
13     }
14
15     @Test
16     public void meuMetodoDeTeste4(){
17         System.out.println(x:"Esse nao falha!!");
18     }
19 }
```

Lançando uma Falha no Teste

```
src > test > java > ifmt > cba > J SegundoTest.java > SegundoTest > meuMetodoDeTeste3()

7 public class SegundoTest {
8
9     @Test
10    public void meuMetodoDeTeste3(){
11        System.out.println(x: "Vai falhar!!");
12        fail("Falhou de proposito"); org.opentest4j.AssertionFailedError: Falhou de proposito at ifmt.cba.SegundoTest.m
org.opentest4j.AssertionFailedError: Falhou de proposito at ifmt.cba.SegundoTest.meuMetodoDeTeste3(SegundoTest.java:12) at java.... meuMetod...
org.opentest4j.AssertionFailedError: Falhou de proposito
at ifmt.cba.SegundoTest.meuMetodoDeTeste3(SegundoTest.java:12)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

%TESTS 3,meuMetodoDeTeste3(ifmt.cba.SegundoTest)
%FAILED 3,meuMetodoDeTeste3(ifmt.cba.SegundoTest)
%TRACES
org.opentest4j.AssertionFailedError: Falhou de proposito
    at org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils.java:38)
    at org.junit.jupiter.api.Assertions.fail(Assertions.java:134)
    at ifmt.cba.SegundoTest.meuMetodoDeTeste3(SegundoTest.java:12)
    at java.base/java.lang.reflect.Method.invoke(Method.java:578)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
%TRACEE
```

Test run at 10/30/2023, 11:01:38 AM

- meuMetodoDeTeste3() org.opentest4j.AssertionFailedEr.
- meuMetodoDeTeste4()

Relatório de Falha no Console

```
1 PS C:\Users\evand\Dropbox\Testes de Software\Implementacao\junit01> mvn test
2 [INFO] Scanning for projects...
3 [INFO]
4 [INFO] -----< ifmt.cba:junit01 >-----
5 [INFO] Building junit01 1.0-SNAPSHOT
6 [INFO] -----[ jar ]-----
7 [INFO] -----
8 [INFO] T E S T S
9 [INFO] -----
10 [INFO] Running ifmt.cba.PrimeiroTest
11 Primeiro exemplo de teste , que nao testa nada!!
12 Segundo exemplo de teste , que nao testa nada!!
13 [INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.050 s -- in ifmt.cba.PrimeiroTest
14 [INFO] Running ifmt.cba.SegundoTest
15 Vai falhar!!
16 Esse nao falha!!
17 [ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.025 s <<< FAILURE! -- in ifmt.cba.SegundoTest
18 [ERROR] ifmt.cba.SegundoTest.meuMetodoDeTeste3 -- Time elapsed: 0.015 s <<< FAILURE!
19 org.opentest4j.AssertionFailedError: Falhou de proposito
20     at org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils.java:38)
21     at org.junit.jupiter.api.Assertions.fail(Assertions.java:134)
22     at ifmt.cba.SegundoTest.meuMetodoDeTeste3(SegundoTest.java:12)
23 [INFO]
24 [INFO] Results:
25 [ERROR] Failures:
26 [ERROR] SegundoTest.meuMetodoDeTeste3:12 Falhou de proposito
27 [INFO]
28 [ERROR] Tests run: 4, Failures: 1, Errors: 0, Skipped: 0
29 [INFO]
30 [INFO] -----
31 [INFO] BUILD FAILURE
32 [INFO] -----
```

Usando Assertions (1)

Classe a ser Testada

```
1 package ifmt.cba;
2
3 public class Produto1 {
4
5     private int codigo;
6     private String nome;
7     private int estoque;
8
9     public Produto1(){
10         this.estoque = 0;
11     }
12
13     public int getCodigo() {
14         return codigo;
15     }
16
17     public void setCodigo(int codigo) {
18         this.codigo = codigo;
19     }
20
21     public String getNome() {
22         return nome;
23     }
24
25     public void setNome(String nome) {
26         this.nome = nome;
27     }
28
29     public int getEstoque() {
```

Usando Assertions (2)

```
30     return estoque;
31 }
32
33 public void setEstoque(int estoque) {
34     this.estoque = estoque;
35 }
36
37 public int adicionarEstoque(int quantidade) {
38
39     int retorno = 0;
40
41     if (quantidade > 0) {
42         this.estoque += quantidade;
43         retorno = quantidade;
44     }
45
46     return retorno;
47 }
48
49 public int baixarEstoque(int quantidade) {
50
51     int retorno = 0;
52
53     if (quantidade > 0) {
54         if (quantidade <= this.estoque) {
55             this.estoque -= quantidade;
56             retorno = quantidade;
57         }
58     }
59     return retorno;
60 }
61
```

Usando Assertions (3)

```
62 public String validar(){
63     String retorno = "";
64
65     if (this.nome == null || this.nome.isEmpty()){
66         retorno += "Nome nao pode ser vazio";
67     }
68
69     if (this.estoque < 0){
70         retorno += "Estoque nao pode ser negativo";
71     }
72
73     return retorno;
74 }
75
76 @Override
77 public boolean equals(Object obj) {
78     if (this == obj) {
79         return true;
80     }
81     if (obj == null) {
82         return false;
83     }
84     if (getClass() != obj.getClass()) {
85         return false;
86     }
87     final Produto1 other = (Produto1) obj;
88     if (this.codigo != other.codigo) {
89         return false;
90     }
91     return true;
92 }
93
```

Usando Assertions (4)

```
94 |  
95 | }
```

Usando Assertions (1)

Classe de Testes

- Granularidade dos casos de testes
- Casos de testes dependem da lógica da implementação
- O ideal é que a equipe de desenvolvimento tenha um padrão para cada nível de classe

```
1 package ifmt.cba;
2
3 import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.Test;
5
6 public class Produto1Test1 {
7
8     @Test
9     public void validarComDadosCorretos() {
10         Produto1 produto = new Produto1();
11         produto.setCodigo(10);
12         produto.setNome("Produto Teste");
13         produto.setEstoque(10);
14         Assertions.assertTrue(produto.validar().isEmpty());
15     }
16
17     @Test
18     public void validarComDadosIncorretos() {
19         Produto1 produto = new Produto1();
20         Assertions.assertFalse(produto.validar().isEmpty());
21     }
22 }
```


Usando Assertions (2)

```
23 @Test
24 public void validarComEstoqueIncorreto() {
25     Produto1 produto = new Produto1();
26     produto.setCodigo(10);
27     produto.setNome("Produto Teste");
28     produto.setEstoque(-5);
29     String retorno = produto.validar();
30     Assertions.assertEquals("Estoque nao pode ser negativo", retorno);
31 }
```

```
32
33 @Test
34 public void adicionarEstoqueCorreto() {
35
36     Produto1 produto = new Produto1();
37     produto.setCodigo(10);
38     produto.setNome("Produto Teste");
39     produto.setEstoque(10);
40
41     produto.adicionarEstoque(5);
42     Assertions.assertEquals(15, produto.getEstoque());
43 }
```

```
44
45 @Test
46 public void adicionarEstoqueIncorreto1() {
47
48     Produto1 produto = new Produto1();
49     produto.setCodigo(10);
50     produto.setNome("Produto Teste");
51     produto.setEstoque(10);
52
53     produto.adicionarEstoque(-5);
54     Assertions.assertEquals(10, produto.getEstoque());
```

Usando Assertions (3)

```
55     }  
56  
57     @Test  
58     public void adicionarEstoqueIncorreto2() {  
59  
60         Produto1 produto = new Produto1();  
61         produto.setCodigo(10);  
62         produto.setNome("Produto Teste");  
63         produto.setEstoque(10);  
64  
65         int retorno = produto.adicionarEstoque(-5);  
66         Assertions.assertEquals(0, retorno);  
67     }  
68 }
```

Importação Estática e Assertions Messages (1)

Usando o import estático

Usando o parâmetro Assertions Messages das Assertions para customizar o texto na saída do relatório de teste, quando o teste falha.

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import org.junit.jupiter.api.Test;
5
6 public class Produto1Test2 {
7
8     @Test
9     public void validarComDadosCorretos() {
10         Produto1 produto = new Produto1();
11         produto.setCodigo(10);
12         produto.setNome("Produto Teste");
13         produto.setEstoque(10);
14         assertTrue(produto.validar().isEmpty(), "Dados invalidos");
15     }
16
17     @Test
18     public void validarComDadosIncorretos() {
19         Produto1 produto = new Produto1();
20         assertFalse(produto.validar().isEmpty(), "Dados invalidos");
21     }
22
23     @Test
24     public void validarComEstoqueIncorreto() {
```

Importação Estática e Assertions Messages (2)

```
25     Produto1 produto = new Produto1();
26     produto.setCodigo(10);
27     produto.setNome("Produto Teste");
28     produto.setEstoque(-5);
29     String retorno = produto.validar();
30     assertEquals("Estoque nao pode ser negativo", retorno, "Estoque incorreto");
31 }
32
33 @Test
34 public void adicionarEstoqueCorreto() {
35
36     Produto1 produto = new Produto1();
37     produto.setCodigo(10);
38     produto.setNome("Produto Teste");
39     produto.setEstoque(10);
40
41     produto.adicionarEstoque(5);
42     assertEquals(15, produto.getEstoque(), "Erro ao adicionar estoque");
43 }
44
45 @Test
46 public void adicionarEstoqueIncorreto1() {
47
48     Produto1 produto = new Produto1();
49     produto.setCodigo(10);
50     produto.setNome("Produto Teste");
51     produto.setEstoque(10);
52
53     produto.adicionarEstoque(-5);
54     assertEquals(10, produto.getEstoque(), "Erro ao adicionar estoque");
55 }
56
```

Importação Estática e Assertions Messages (3)

```
57 @Test
58 public void adicionarEstoqueIncorreto2 () {
59
60     Produto1 produto = new Produto1 ();
61     produto.setCodigo(10);
62     produto.setNome("Produto Teste");
63     produto.setEstoque(10);
64
65     int retorno = produto.adicionarEstoque(-5);
66     assertEquals(0, retorno, "Erro ao adicionar estoque");
67 }
68 }
```

Padrão de Nomenclatura dos Testes (1)

Nome de Classe de Teste: <Nome da Classe> + <sufixo Test>

Nome dos casos de Testes/métodos: <prefixo test> + <o que testar> + <condição/estado> + <resultado esperado>

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertFalse;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6
7 import org.junit.jupiter.api.Test;
8
9 public class Produto1Test3 {
10
11     @Test
12     public void testDados_QuandoCorretos_TextoValidacaoVazio() {
13         Produto1 produto = new Produto1();
14         produto.setCodigo(10);
15         produto.setNome("Produto Teste");
16         produto.setEstoque(10);
17         assertTrue(produto.validar().isEmpty(), "Dados invalidos");
18     }
19
20     @Test
21     public void testDados_QuandoInCorretos_TextoValidacaoNaoVazio() {
22         Produto1 produto = new Produto1();
23         assertFalse(produto.validar().isEmpty(), "Dados invalidos");
24     }
25 }
```

Padrão de Nomenclatura dos Testes (2)

```
25
26 @Test
27 public void testEstoque_QuandoNegativo_RetornoMensagemErro() {
28     Produto1 produto = new Produto1();
29     produto.setCodigo(10);
30     produto.setNome("Produto Teste");
31     produto.setEstoque(-5);
32     String retorno = produto.validar();
33     assertEquals("Estoque nao pode ser negativo", retorno, "Estoque incorreto");
34 }
35
36 @Test
37 public void testAdicionarEstoque_QuandoCorreto_AdicionaQuantidadeEstoque() {
38
39     Produto1 produto = new Produto1();
40     produto.setCodigo(10);
41     produto.setNome("Produto Teste");
42     produto.setEstoque(10);
43
44     produto.adicionarEstoque(5);
45     assertEquals(15, produto.getEstoque(), "Erro ao adicionar estoque");
46 }
47
48 @Test
49 public void testAdicionarEstoque_QuandoQuantidadeNegativa_NaoAdicionaQuantidadeEstoque() {
50
51     Produto1 produto = new Produto1();
52     produto.setCodigo(10);
53     produto.setNome("Produto Teste");
54     produto.setEstoque(10);
55
56     produto.adicionarEstoque(-5);
```

Padrão de Nomenclatura dos Testes (3)

```
57     assertEquals(10, produto.getEstoque(), "Erro ao adicionar estoque");
58 }
59
60 @Test
61 public void testAdicionarEstoque_QuandoQuantidadeNegativa_RetornaZeroAdicionado() {
62
63     Produto1 produto = new Produto1();
64     produto.setCodigo(10);
65     produto.setNome("Produto Teste");
66     produto.setEstoque(10);
67
68     int retorno = produto.adicionarEstoque(-5);
69     assertEquals(0, retorno, "Erro ao adicionar estoque");
70 }
71 }
```


Padrão de Nomenclatura dos Testes

The screenshot shows an IDE window with a Java file named `Produto1Test3.java`. The code defines a test class `Produto1Test3` with a single test method `testDados_QuandoCorretos_TextoValidacaoVazio()`. The test method creates a `Produto1` object, sets its code to `10`, and sets its name to `"Produto Teste"`. Below the code editor, the `TEST RESULTS` tab is active, showing a list of test methods that passed. The first test run was at 10/31/2023, 9:35:14 AM, and the second at 10/31/2023, 8:58:43 AM.

```
J Produto1Test3.java ×
src > test > java > ifmt > cba > J Produto1Test3.java > Produto1Test3 > testAdicionarEstoque_QuandoQuantidadeNegativa_Reto
5  import static org.junit.jupiter.api.Assertions.assertTrue;
6
7  import org.junit.jupiter.api.Test;
8
9  ✓ public class Produto1Test3 {
10
11      @Test
12      ✓ public void testDados_QuandoCorretos_TextoValidacaoVazio() {
13          Produto1 produto = new Produto1();
14          produto.setCodigo(codigo:10);
15          produto.setNome(nome:"Produto Teste");
16      }
17  }
```

PROBLEMS OUTPUT TEST RESULTS TERMINAL PORTS DEBUG CONSOLE

ionado()]]
%TSTTREE5,testAdicionarE
stoque_QuandoCorreto_Adi
cionaQuantidadeEstoque(i
fmt.cba.Produto1Test3),f
alse,1,false,2,testAdici
onarEstoque_QuandoCorret
o_AdicionaQuantidadeEsto
que(),,[engine:junit-jup
iter]/[class:ifmt.cba.Pr
oduto1Test3]/[method:tes
tAdicionarEstoque_Quando
Correto_AdicionaQuantida
deEstoque()]]

✓ Test run at 10/31/2023, 9:35:14 AM

- ✓ testDados_QuandoCorretos_TextoValidacaoVazio()
- ✓ testDados_QuandoInCorretos_TextoValidacaoNaoVazio()
- ✓ testEstoque_QuandoNegativo_RetornoMensagemErro()
- ✓ testAdicionarEstoque_QuandoCorreto_AdicionaQuantidadeEstoque()
- ✓ testAdicionarEstoque_QuandoQuantidadeNegativa_NaoAdicionaQuantidadeEstoque()
- ✓ testAdicionarEstoque_QuandoQuantidadeNegativa_RetornaZeroAdicionado()

> Test run at 10/31/2023, 8:58:43 AM

Usando a Anotação DisplayName (1)

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertFalse;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6
7 import org.junit.jupiter.api.DisplayName;
8 import org.junit.jupiter.api.Test;
9
10
11 @DisplayName("Testando a validacao de dados da Classe Produto1")
12 public class Produto1Test4 {
13
14     @Test
15     @DisplayName("Testando a validacao quando os dados estao corretos, deve retornar texto de validacao vazio")
16     public void validarComDadosCorretos() {
17         Produto1 produto = new Produto1();
18         produto.setCodigo(10);
19         produto.setNome("Produto Teste");
20         produto.setEstoque(10);
21         assertTrue(produto.validar().isEmpty(), "Dados invalidos");
22     }
23
24     @Test
25     @DisplayName("Testando a validacao quando os dados estao incorretos, deve retornar texto de validacao nao vazio")
26     public void validarComDadosIncorretos() {
27         Produto1 produto = new Produto1();
28         assertFalse(produto.validar().isEmpty(), "Dados invalidos");
29     }
30
31     @Test
```

Usando a Anotação DisplayName (2)

```
32 @DisplayName("Testando a validacao do estoque negativo, deve retornar o texto <Estoque nao pode ser negativo>")
33 public void validarComEstoqueIncorreto() {
34     Produto1 produto = new Produto1();
35     produto.setCodigo(10);
36     produto.setNome("Produto Teste");
37     produto.setEstoque(-5);
38     String retorno = produto.validar();
39     assertEquals("Estoque nao pode ser negativo", retorno, "Estoque incorreto");
40 }
41
42 @Test
43 @DisplayName("Testando a adicao de quantidde positiva ao estoque, deve adicionar a quantidade passada")
44 public void adicionarEstoqueCorreto() {
45
46     Produto1 produto = new Produto1();
47     produto.setCodigo(10);
48     produto.setNome("Produto Teste");
49     produto.setEstoque(10);
50
51     produto.adicionarEstoque(5);
52     assertEquals(15, produto.getEstoque(), "Erro ao adiconar estoque");
53 }
54
55 @Test
56 @DisplayName("Testando a adicao de quantidde negativa ao estoque, nao deve adicionar a quantidade passada")
57 public void adicionarEstoqueIncorreto1() {
58
59     Produto1 produto = new Produto1();
60     produto.setCodigo(10);
61     produto.setNome("Produto Teste");
62     produto.setEstoque(10);
63 }
```

Usando a Anotação DisplayName (3)

```
64     produto.adicionarEstoque(-5);
65     assertEquals(10, produto.getEstoque(), "Erro ao adicionar estoque");
66 }
67
68 @Test
69 @DisplayName("Testando o retorno da adicao de quantidde negativa ao estoque, deve retornar zero")
70 public void adicionarEstoqueIncorreto2() {
71
72     Produto1 produto = new Produto1();
73     produto.setCodigo(10);
74     produto.setNome("Produto Teste");
75     produto.setEstoque(10);
76
77     int retorno = produto.adicionarEstoque(-5);
78     assertEquals(0, retorno, "Erro ao adicionar estoque");
79 }
80 }
```

Usando a Anotação DisplayName

Produto1Test3.java

Produto1Test4.java X

Produto1Test2.java

src > test > java > ifmt > cba > Produto1Test4.java > Produto1Test4

9

10

11 @DisplayName("Testando a validacao de dados da Classe Produto1")

12 public class Produto1Test4 {

13

14 @Test

15 @DisplayName("Testando a validacao quando os dados estao corretos, deve retornar texto de validacao vazio")

16 public void validarComDadosCorretos() {

17

PROBLEMS OUTPUT TEST RESULTS TERMINAL PORTS DEBUG CONSOLE

%TESTC 6 v2

%TSTTREE2,ifmt.cba.Produto1Test4,true,6,false,1,Testando a validacao de dados da Classe Produto1,,[engine:junit-jupiter]/[class:ifmt.cba.Produto1Test4]

%TSTTREE3,adicionarEstoqueIncorreto1(ifmt.cba.Produto1Test4),false,1,false,2,Testando a adicao de qu

%TSTTREE4,adicionarEstoqueIncorreto2(ifmt.cba.Produto1Test4),false,1,false,2,Testando o retorno da a

%TSTTREE5,validarComDadosIncorretos(ifmt.cba.Produto1Test4),false,1,false,2,Testando a validacao qua

%TSTTREE6,validarComDadosCorretos(ifmt.cba.Produto1Test4),false,1,false,2,Testando a validacao quand

%TSTTREE7,adicionarEstoqueCorreto(ifmt.cba.Produto1Test4),false,1,false,2,Testando a adicao de quant

%TSTTREE8,validarComEstoqueIncorreto(ifmt.cba.Produto1Test4),false,1,false,2,Testando a validacao do

%TESTS 3,adicionarEstoqueIncorreto1(ifmt.cba.Produto1Test4)

Test run at 10/31/2023, 1...

adicionarEstoqueCorreto()

adicionarEstoqueIncorreto1()

adicionarEstoqueIncorreto2()

validarComDadosCorretos()

validarComDadosIncorretos()

validarComEstoqueIncorreto()

Desativando um Caso de Teste

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.Disabled;
6 import org.junit.jupiter.api.DisplayName;
7 import org.junit.jupiter.api.Test;
8
9 @DisplayName("Testando a validacao de dados da Classe Produto1")
10 public class Produto1Test5 {
11
12     @Test
13     @DisplayName("Testando a validacao quando os dados estao corretos, deve retornar texto de validacao vazio")
14     public void validarComDadosCorretos() {
15         Produto1 produto = new Produto1();
16         produto.setCodigo(10);
17         produto.setNome("Produto Teste");
18         produto.setEstoque(10);
19         assertTrue(produto.validar().isEmpty(), "Dados invalidos");
20     }
21
22     @Test
23     @Disabled("Teste desativado temporariamente")
24     @DisplayName("Testando a validacao quando os dados estao incorretos, deve retornar texto de validacao nao vazio")
25     public void validarComDadosIncorretos() {
26         Produto1 produto = new Produto1();
27         assertFalse(produto.validar().isEmpty(), "Dados invalidos");
28     }
29 }
```

Ciclo de Vida dos Casos de Testes

São quatro Anotações para controlar o Ciclo de Vida:

@BeforeAll - o método com essa anotação será executado uma vez **antes de todos** os casos de testes da classe

@BeforeEach - o método com essa anotação será executado **antes de cada** casos de testes da classe

@AfterEach - o método com essa anotação será executado **depois de cada** casos de testes da classe

@AfterAll - o método com essa anotação será executado **depois de todos** casos de testes da classe

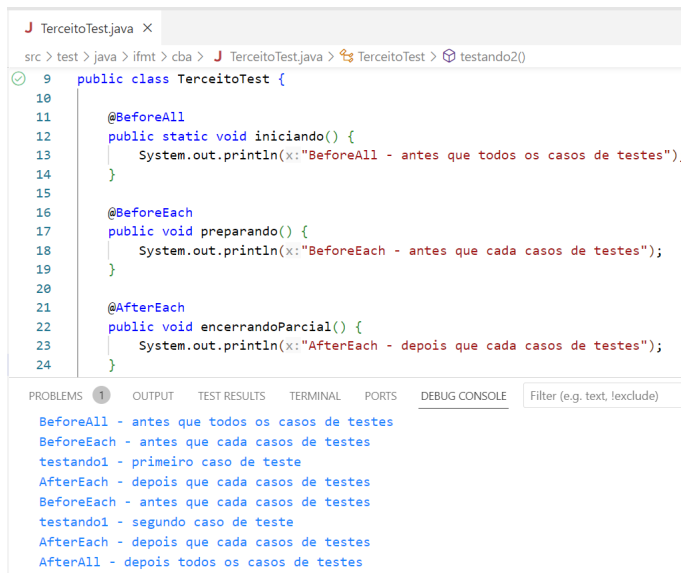
Ciclo de Vida dos Casos de Testes (1)

```
1 package ifmt.cba;
2
3 import org.junit.jupiter.api.AfterAll;
4 import org.junit.jupiter.api.AfterEach;
5 import org.junit.jupiter.api.BeforeAll;
6 import org.junit.jupiter.api.BeforeEach;
7 import org.junit.jupiter.api.Test;
8
9 public class TerceiroTest {
10
11     @BeforeAll
12     public static void iniciando() {
13         System.out.println("BeforeAll – antes que todos os casos de testes");
14     }
15
16     @BeforeEach
17     public void preparando() {
18         System.out.println("BeforeEach – antes que cada casos de testes");
19     }
20
21     @AfterEach
22     public void encerrandoParcial() {
23         System.out.println("AfterEach – depois que cada casos de testes");
24     }
25
26     @AfterAll
27     public static void encerrandoTotal() {
28         System.out.println("AfterAll – depois todos os casos de testes");
29     }
30
31     @Test
```


Ciclo de Vida dos Casos de Testes (2)

```
32 public void testando1() {  
33     System.out.println("testando1 - primeiro caso de teste");  
34 }  
35  
36 @Test  
37 public void testando2() {  
38     System.out.println("testando1 - segundo caso de teste");  
39 }  
40 }
```

Ciclo de Vida dos Casos de Testes



The screenshot shows an IDE window with a file named `TerceitoTest.java`. The code defines a `public class TerceitoTest` with three static methods: `@BeforeAll` (`iniciando()`), `@BeforeEach` (`preparando()`), and `@AfterEach` (`encerrandoParcial()`). Each method prints a message to the console. The IDE's output window at the bottom shows the execution results, including the messages from the test lifecycle methods and the test cases themselves.

```
9 public class TerceitoTest {
10
11     @BeforeAll
12     public static void iniciando() {
13         System.out.println(x:"BeforeAll - antes que todos os casos de testes");
14     }
15
16     @BeforeEach
17     public void preparando() {
18         System.out.println(x:"BeforeEach - antes que cada casos de testes");
19     }
20
21     @AfterEach
22     public void encerrandoParcial() {
23         System.out.println(x:"AfterEach - depois que cada casos de testes");
24     }
25 }
```

PROBLEMS 1 OUTPUT TEST RESULTS TERMINAL PORTS DEBUG CONSOLE Filter (e.g. text, !exclude)

```
BeforeAll - antes que todos os casos de testes
BeforeEach - antes que cada casos de testes
testando1 - primeiro caso de teste
AfterEach - depois que cada casos de testes
BeforeEach - antes que cada casos de testes
testando1 - segundo caso de teste
AfterEach - depois que cada casos de testes
AfterAll - depois todos os casos de testes
```

Ciclo de Vida dos Casos de Testese (1)

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertFalse;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6
7 import org.junit.jupiter.api.AfterEach;
8 import org.junit.jupiter.api.BeforeEach;
9 import org.junit.jupiter.api.Test;
10
11 public class Produto1Test6 {
12
13     private Produto1 produto;
14
15     @BeforeEach
16     public void iniciandoProduto() {
17         this.produto = new Produto1();
18     }
19
20     @AfterEach
21     public void encerrandoProduto() {
22         this.produto = null;
23     }
24
25     @Test
26     public void testDados_QuandoCorretos_TextoValidacaoVazio() {
27         produto.setCodigo(10);
28         produto.setNome("Produto Teste");
29         produto.setEstoque(10);
30         assertTrue(produto.validar().isEmpty(), "Dados invalidos");
31     }
32 }
```

Ciclo de Vida dos Casos de Testese (2)

```
32
33 @Test
34 public void testDados_QuandoInCorretos_TextoValidacaoNaoVazio() {
35     assertFalse(prodoto.validar().isEmpty(), "Dados invalidos");
36 }
37
38 @Test
39 public void testEstoque_QuandoNegativo_RetornoMensagemErro() {
40     produto.setCodigo(10);
41     produto.setNome("Produto Teste");
42     produto.setEstoque(-5);
43     String retorno = produto.validar();
44     assertEquals("Estoque nao pode ser negativo", retorno, "Estoque incorreto");
45 }
46
47 @Test
48 public void testAdicionarEstoque_QuandoCorreto_AdicionaQuantidadeEstoque() {
49     produto.setCodigo(10);
50     produto.setNome("Produto Teste");
51     produto.setEstoque(10);
52
53     produto.adicionarEstoque(5);
54     assertEquals(15, produto.getEstoque(), "Erro ao adiconar estoque");
55 }
56
57 @Test
58 public void testAdicionarEstoque_QuandoQuantidadeNegativa_NaoAdicionaQuantidadeEstoque() {
59     produto.setCodigo(10);
60     produto.setNome("Produto Teste");
61     produto.setEstoque(10);
62
63     produto.adicionarEstoque(-5);
```

Ciclo de Vida dos Casos de Testese (3)

```
64     assertEquals(10, produto.getEstoque(), "Erro ao adicionar estoque");
65 }
66
67 @Test
68 public void testAdicionarEstoque_QuandoQuantidadeNegativa_RetornaZeroAdicionado() {
69     produto.setCodigo(10);
70     produto.setNome("Produto Teste");
71     produto.setEstoque(10);
72
73     int retorno = produto.adicionarEstoque(-5);
74     assertEquals(0, retorno, "Erro ao adicionar estoque");
75 }
76 }
```

Produto2 - Gera Exceção (1)

```
1 package ifmt.cba;
2
3 public class Produto2 {
4
5     private int codigo;
6     private String nome;
7     private int estoque;
8
9     public Produto2(){
10         this.estoque = 0;
11     }
12
13     public int getCodigo() {
14         return codigo;
15     }
16
17     public void setCodigo(int codigo) {
18         this.codigo = codigo;
19     }
20
21     public String getNome() {
22         return nome;
23     }
24
25     public void setNome(String nome) {
26         this.nome = nome;
27     }
28
29     public int getEstoque() {
30         return estoque;
31     }
32 }
```

Produto2 - Gera Exceção (2)

```
32 public void setEstoque(int estoque) {
33     this.estoque = estoque;
34 }
35
36 public void adicionarEstoque(int quantidade) throws Exception {
37     if (quantidade > 0) {
38         this.estoque += quantidade;
39     } else {
40         throw new Exception("Quantidade deve ser maior que zero");
41     }
42 }
43
44 public void baixarEstoque(int quantidade) throws Exception {
45     if (quantidade > 0) {
46         if (quantidade <= this.estoque) {
47             this.estoque -= quantidade;
48         } else {
49             throw new Exception("Estoque insuficiente");
50         }
51     } else {
52         throw new Exception("Quantidade deve ser maior que zero");
53     }
54 }
55
56 public String validar(){
57     String retorno = "";
58
59     if (this.nome == null || this.nome.isEmpty()){
60         retorno += "Nome nao pode ser vazio";
61     }
62 }
63
```

Produto2 - Gera Exceção (3)

```
64     if (this.estoque < 0){
65         retorno += "Estoque nao pode ser negativo";
66     }
67
68     return retorno;
69 }
70
71 @Override
72 public boolean equals(Object obj) {
73     if (this == obj) {
74         return true;
75     }
76     if (obj == null) {
77         return false;
78     }
79     if (getClass() != obj.getClass()) {
80         return false;
81     }
82     final Produto2 other = (Produto2) obj;
83     if (this.codigo != other.codigo) {
84         return false;
85     }
86     return true;
87 }
88
89
90 }
```


assertDoesNotThrow e assertThrows (1)

Para testar métodos que potencialmente podem lançar Exceções.

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.AfterEach;
6 import org.junit.jupiter.api.BeforeEach;
7 import org.junit.jupiter.api.Test;
8
9 public class Produto2Test1 {
10
11     private Produto2 produto;
12
13     @BeforeEach
14     public void iniciandoProduto() {
15         this.produto = new Produto2();
16         produto.setCodigo(10);
17         produto.setNome("Produto Teste");
18         produto.setEstoque(10);
19     }
20
21     @Test
22     public void testAdicionarEstoque_QuandoQuantidadePositiva_AdicionaAoEstoque() {
23
24         assertDoesNotThrow(() -> produto.adicionarEstoque(5), "Nao era esperada uma excecao");
25         assertEquals(15, produto.getEstoque());
26     }
27
28     @Test
29     public void testAdicionarEstoque_QuandoQuantidadeNegativa_GeraExcecao() {
```

assertDoesNotThrow e assertThrows (2)

```
30  
31     assertThrows(Exception.class, () -> produto.adicionarEstoque(-5),  
32         "Esperava-se a producao de uma exceciao");  
33     }  
34 }
```

assertDoesNotThrow e assertThrows (1)

Capturando e comparando a mensagem da Exceções.

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertDoesNotThrow;
4 import static org.junit.jupiter.api.Assertions.assertEquals;
5 import static org.junit.jupiter.api.Assertions.assertThrows;
6 import static org.junit.jupiter.api.Assertions.assertTrue;
7
8 import org.junit.jupiter.api.BeforeEach;
9 import org.junit.jupiter.api.Test;
10
11 public class Produto2Test2 {
12     private Produto2 produto;
13
14     @BeforeEach
15     public void iniciandoProduto() {
16         this.produto = new Produto2();
17         produto.setCodigo(10);
18         produto.setNome("Produto Teste");
19         produto.setEstoque(10);
20     }
21
22     @Test
23     public void testAdicionarEstoque_QuandoQuantidadePositiva_AdicionaAoEstoque() {
24
25         assertDoesNotThrow(() -> produto.adicionarEstoque(5), "Nao era esperada uma excecacao");
26         assertEquals(15, produto.getEstoque());
27     }
28
29     @Test
```

assertDoesNotThrow e assertThrows (2)

```
30 public void testAdicionarEstoque_QuandoQuantidadeNegativa_GeraExcecao() {
31
32     Exception thrown = assertThrows(Exception.class, () -> produto.adicionarEstoque(-5),
33     "Esperava-se a producao de uma excecao");
34     assertTrue(thrown.getMessage().contains("Quantidade deve ser maior que zero"));
35 }
36 }
```

Comparando de Arrays com Asserts

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
5 import org.junit.jupiter.api.Test;
6
7 public class ArrayTest1 {
8
9     @Test
10     public void testComparandoArrays1 () {
11
12         int[] lista1 = {10, 1, 3, 5, 4, 9, 8, 7, 6};
13         int[] lista2 = {10, 1, 3, 5, 4, 9, 8, 7, 6};
14
15         assertEquals(lista1, lista2);
16     }
17 }
```

Comparando de Arrays com Asserts

Compara as instâncias dos objetos (ID dos objetos) e não os elementos do Array

```
J ArrayTest1.java x
src > test > java > ifmt > cba > J ArrayTest1.java > ArrayTest1 > testComparandoArrays1()
10 public void testComparandoArrays1(){
11
12     int[] lista1 = {10, 1, 3, 5, 4, 9, 8, 7, 6};
13     int[] lista2 = {10, 1, 3, 5, 4, 9, 8, 7, 6};
14
15     assertEquals(lista1, lista2); Expected [[I@4d826d77] but was: [[I@61009542]

Expected [[I@4d826d77] but was [[I@61009542] testComparandoArrays1()
Expected
-[I@4d826d77
Actual
+[I@61009542

PROBLEMS 1 OUTPUT TEST RESULTS TERMINAL PORTS DEBUG CONSOLE
%EXPECTS
[I@4d826d77
%EXPECTE
%ACTUALS
[I@61009542
%ACTUALE
%TRACES
org.opentest4j.AssertionFailedError: expected:
[I@4d826d77<[10, 1, 3, 5, 4, 9, 8, 7, 6]> but
was: [I@61009542<[10, 1, 3, 5, 4, 9, 8, 7, 6]
>
at org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:
151)
at org.junit.jupiter.api.AssertionFailureBuilder.buildAndThrow(AssertionFailureBuild
Test run at 11/3/2023, 11:55:05 AM
testComparandoArrays1()
Expected [[I@4d826d77] but was [[I@61009542]
org.opentest4j.AssertionFailedError: expected: [[I@4d826d77][10, 1, 3,...
```

Comparando de Arrays com Asserts

```
1 package ifmt.cba;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
5 import org.junit.jupiter.api.Test;
6
7 public class ArrayTest2 {
8
9     @Test
10     public void testComparandoArrays1 () {
11
12         int[] lista1 = {10, 1, 3, 5, 4, 9, 8, 7, 6, 2};
13         int[] lista2 = {10, 1, 3, 5, 4, 9, 8, 7, 6, 2};
14
15         assertEquals(lista1, lista2);
16     }
17 }
```

Classe que Simula o Processamento

```
1 package ifmt.cba;
2
3 public class AcaoDemorada {
4
5     public void metodoParaEsperar(long espera){
6         try {
7             Thread.sleep(espera);
8         } catch (InterruptedException e) {
9             // Nao faz nada
10        }
11    }
12 }
```


Anotação @Timeout

```
1 package ifmt.cba;
2
3 import java.util.concurrent.TimeUnit;
4
5 import org.junit.jupiter.api.Test;
6 import org.junit.jupiter.api.Timeout;
7
8 public class TimeOutTest1 {
9
10     @Test
11     @Timeout(value = 1000, unit = TimeUnit.MILLISECONDS)
12     public void testTimeOut() {
13         AcaoDemorada acaoDemorada = new AcaoDemorada();
14         acaoDemorada.metodoParaEsperar(150);
15         acaoDemorada.metodoParaEsperar(250);
16         acaoDemorada.metodoParaEsperar(150);
17     }
18 }
```

Todo o processamento gasto com a preparação ou encerramento do teste será computado para verificar o Timeout.

Asserts assertTimeout

```
1 package ifmt.cba;
2
3 import java.time.Duration;
4
5 import org.junit.jupiter.api.Assertions;
6 import org.junit.jupiter.api.Test;
7
8 public class TimeOutTest2 {
9
10     @Test
11     public void testTimeOut() {
12
13         AcaoDemorada acaoDemorada = new AcaoDemorada();
14         Assertions.assertTimeout(Duration.ofMillis(200), ()-> acaoDemorada.metodoParaEsperar(150));
15         Assertions.assertTimeout(Duration.ofMillis(300), ()-> acaoDemorada.metodoParaEsperar(250));
16         Assertions.assertTimeout(Duration.ofMillis(200), ()-> acaoDemorada.metodoParaEsperar(150));
17     }
18 }
```

Controla o Timeout para cada operação a ser monitorada.

Asserts assertTimeoutPreemptively

Não espera o final do processamento para encerrar o Teste, ultrapassando o limite estabelecido pela Duração, o teste falha.

```
1 package ifmt.cba;
2
3 import java.time.Duration;
4 import org.junit.jupiter.api.Assertions;
5 import org.junit.jupiter.api.Test;
6
7 public class TimeOutTest3 {
8
9     @Test
10     public void testTimeOut() {
11
12         AcaoDemorada acaoDemorada = new AcaoDemorada();
13
14         Assertions.assertTimeoutPreemptively(Duration.ofMillis(200), () -> acaoDemorada.metodoParaEsperar(1500));
15         Assertions.assertTimeoutPreemptively(Duration.ofMillis(300), () -> acaoDemorada.metodoParaEsperar(2500));
16         Assertions.assertTimeoutPreemptively(Duration.ofMillis(200), () -> acaoDemorada.metodoParaEsperar(1500));
17     }
18 }
```

assertTimeoutPreemptively