Teste Automatizado

VV&T

Prof. Marcio Delamaro

Teste

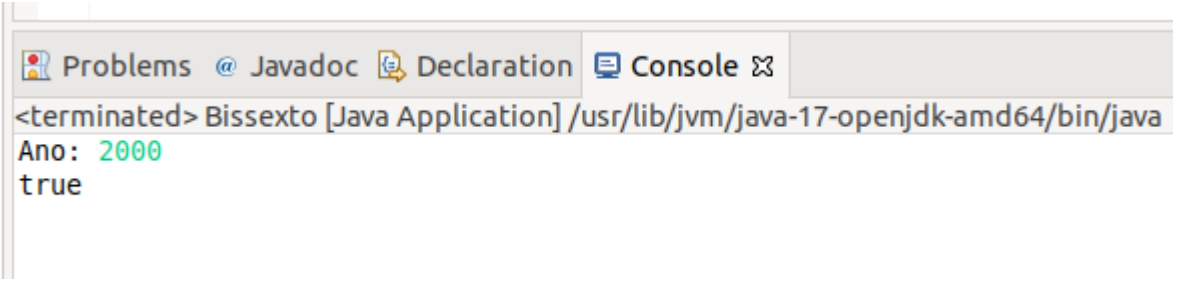
- Ato de executar um programa e verificar se os resultados produzidos estão corretos
- Manual: realizado passando-se os parâmetros e “olhando” se o resultado está certo
- Automatizado: escrevendo um programa que executa seu programa e verifica os resultados
- A vantagem do teste automatizado é que ele pode ser executado muitas vezes, com um mínimo de esforço
- Teste de regressão

Classe Bissexto

- Voltando ao Bissexto. Vamos testar a classe manualmente
- Executamos método
- Em cada vez verificamos se o resultado apresentado está correto
- Por exemplo:
  - 0, 1, 1752, 2000, 9999

Classe Bissexto

- Voltando ao Bissexto. Vamos testar a classe manualmente
- Executamos método
- Em cada vez verificamos se o resultado apresentado está correto
- Por exemplo:
  - 0, 1, 1752, 2000, 9999



Automatizar

- Antes de usar uma classe, vamos tentar garantir que ela funciona
- Ao alterarmos a implementação de uma classe também

Automatizar

- Antes de usar uma classe, vamos tentar garantir que ela funciona
- Ao alterarmos a implementação de uma classe também

```
/**
 * Não tem função real dentro da classe. Foi usada
 * apenas para testar os métodos implementados
 * @param args -- Sem uso.
 */
static public void main(String[] args)
```

Iniciando a automatização

```
static public void main(String[] args)  {
    Bissexto2 b2 = new Bissexto2();
    // ano 1
    System.out.printf("ano: %d. Valor: %b\n", 1, b2.ehBissexto(1) );

    // ano 1752
    System.out.printf("ano: %d. Valor: %b\n", 1752, b2.ehBissexto(1752) );

    // ano 2000
    System.out.printf("ano: %d. Valor: %b\n", 2000, b2.ehBissexto(2000) );

    // ano 9999
    System.out.printf("ano: %d. Valor: %b\n", 9999, b2.ehBissexto(9999) );

    //ano 0
    System.out.printf("ano: %d. Valor: %b\n", 0, b2.ehBissexto(0) );
}
```

Resultado


Problems
Javadoc
Declaration
Console
<terminated> Bissexto2 [Java Application] /usr/lib/jvm/java-17-openjdk-amd64/bin/java (27 de mar. de 2023 19:28:34 – 19:28:34
ano: 1. Valor: false
ano: 1752. Valor: true
ano: 2000. Valor: true
ano: 9999. Valor: false
Exception in thread "main" java.lang.IllegalArgumentException: Ano inválido
at Bissexto2.ehBissexto(Bissexto2.java:9)
at Bissexto2.main(Bissexto2.java:77)

Resultado

Como podemos melhorar essa automatização?

Continuamos testando manualmente. Está correto o resultado apresentado?

Melhorando

```
static public void main(String[] args)  {
    Bissexto2 b2 = new Bissexto2();
    // ano 1
    if ( b2.ehBissexto(1) )
        System.out.printf("Caso falhou: ano = %d. Valor esperado: %b\n", 1, false);
    else
        System.out.printf("Caso passou: ano = %d. Valor esperado: %b\n", 1, false);

    // ano 1752
    if ( ! b2.ehBissexto(1752) )
        System.out.printf("Caso falhou: ano = %d. Valor esperado: %b\n", 1752, true);
    else
        System.out.printf("Caso passou: ano = %d. Valor esperado: %b\n", 1752, true);

    // ano 2000
    if ( ! b2.ehBissexto(2000) )
        System.out.printf("Caso falhou: ano = %d. Valor esperado: %b\n", 2000, true);
    else
        System.out.printf("Caso passou: ano = %d. Valor esperado: %b\n", 2000, true);
```

Melhorando

```
    // ano 9999
    if ( b2.ehBissexto(9999) )
        System.out.printf("Caso falhou: ano = %d. Valor esperado: %b\n", 9999, false);
    else
        System.out.printf("Caso passou: ano = %d. Valor esperado: %b\n", 9999, false);

    //ano 0
    try {
        b2.ehBissexto(0);

        System.out.printf("Caso falhou: ano = %d. Valor esperado: %s\n", 0, "IllegalArgumentException");
    }
    catch (IllegalArgumentException e) {
        System.out.printf("Caso passou: ano = %d. Valor esperado: %s\n", 0, "IllegalArgumentException");
    }

}
```

Problema

- O problema é que a quantidade de código que é necessária para implementar os casos de teste é enorme
- Existem meios mais adequados para se fazer esse tipo de automatização

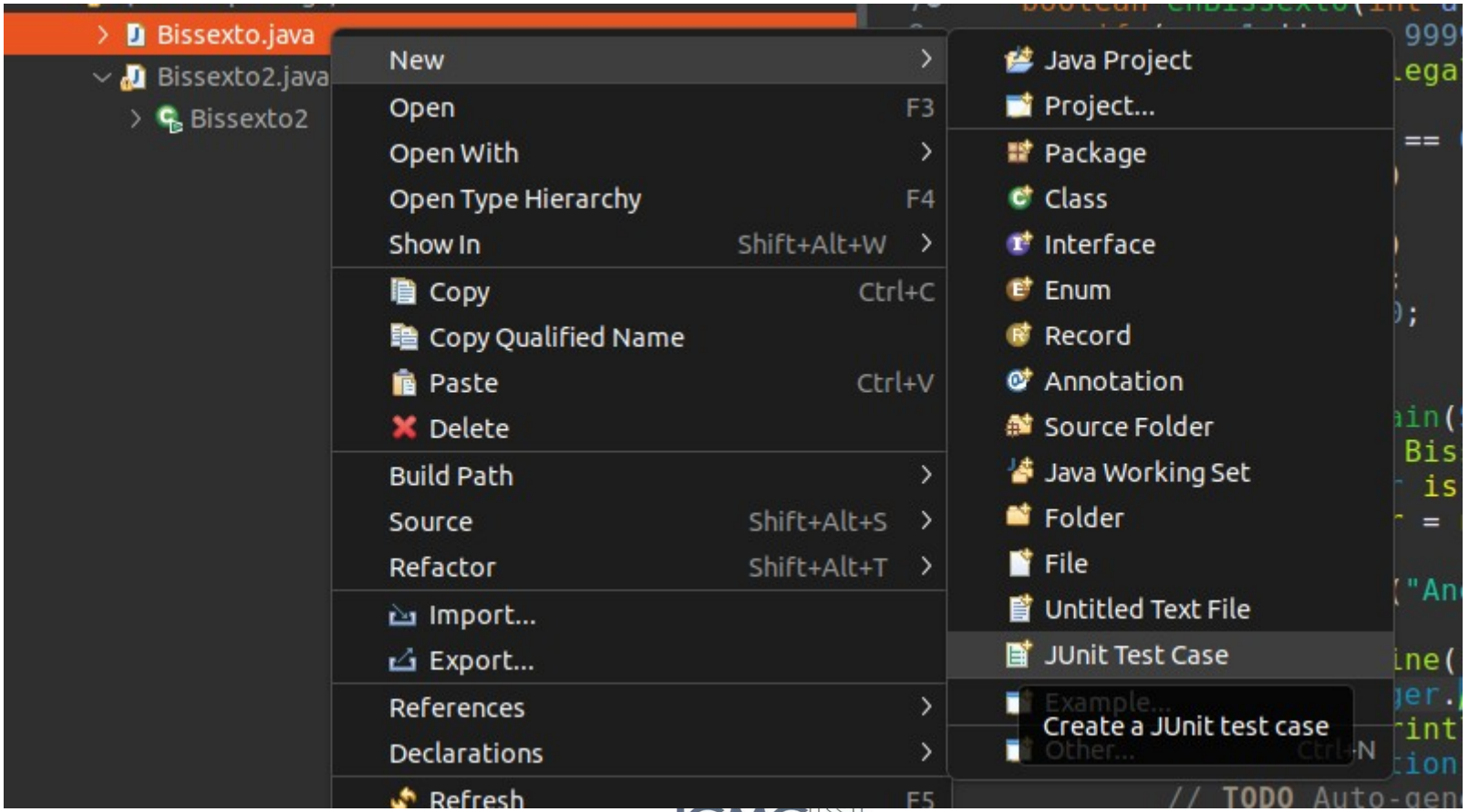
Junit

- Framework para automatização
- Teste de unidade
- Permite definir casos de teste
  - Entradas para um método
  - Saídas esperadas para aquelas entradas
- São definidos usando a própria linguagem Java

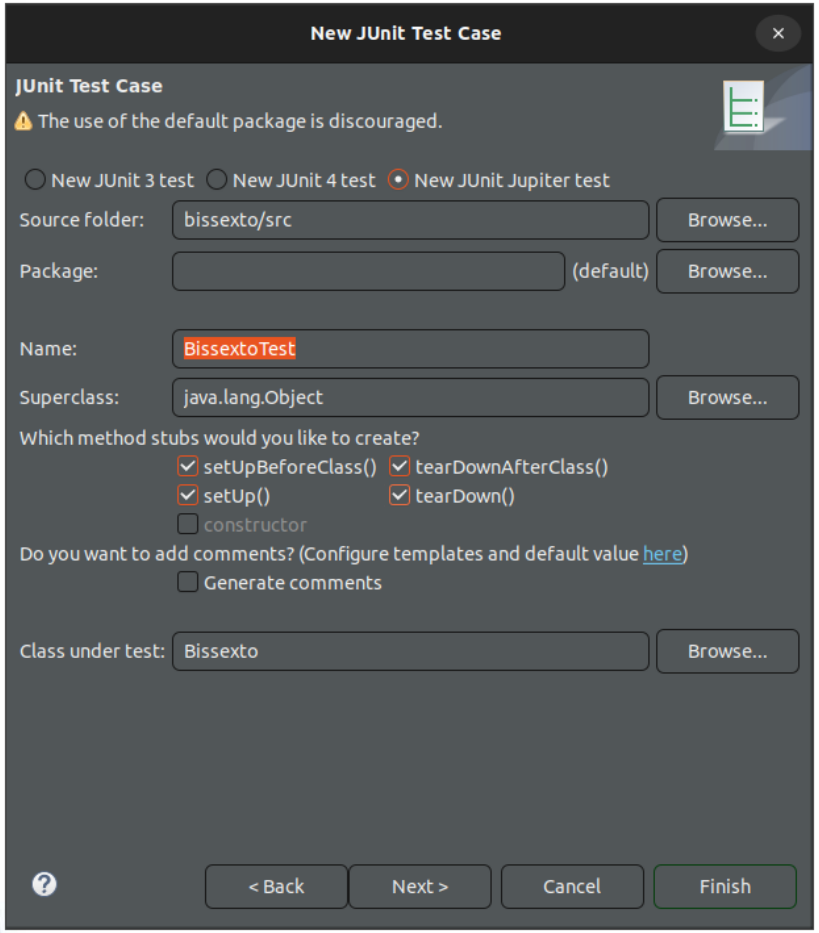
Criar classe de teste

- IDEs como Eclipse possuem suporte ao JUnit
- Criar a classe de teste: Selecionar classe a testar >> File New >> JUnit

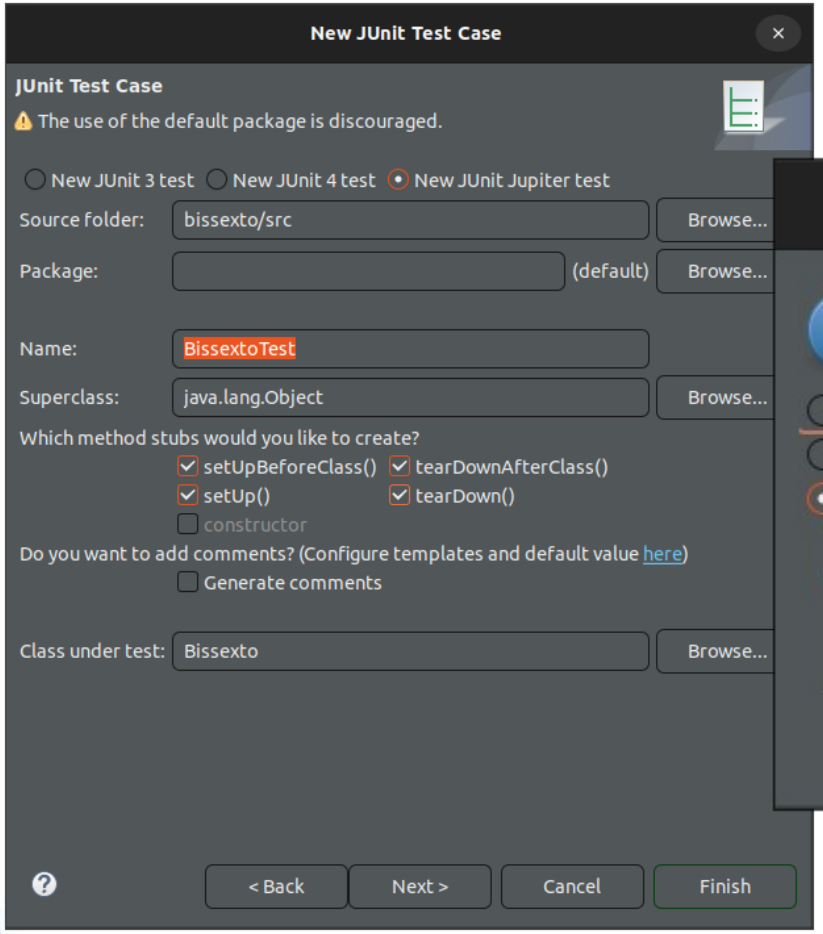
Definir classe de teste

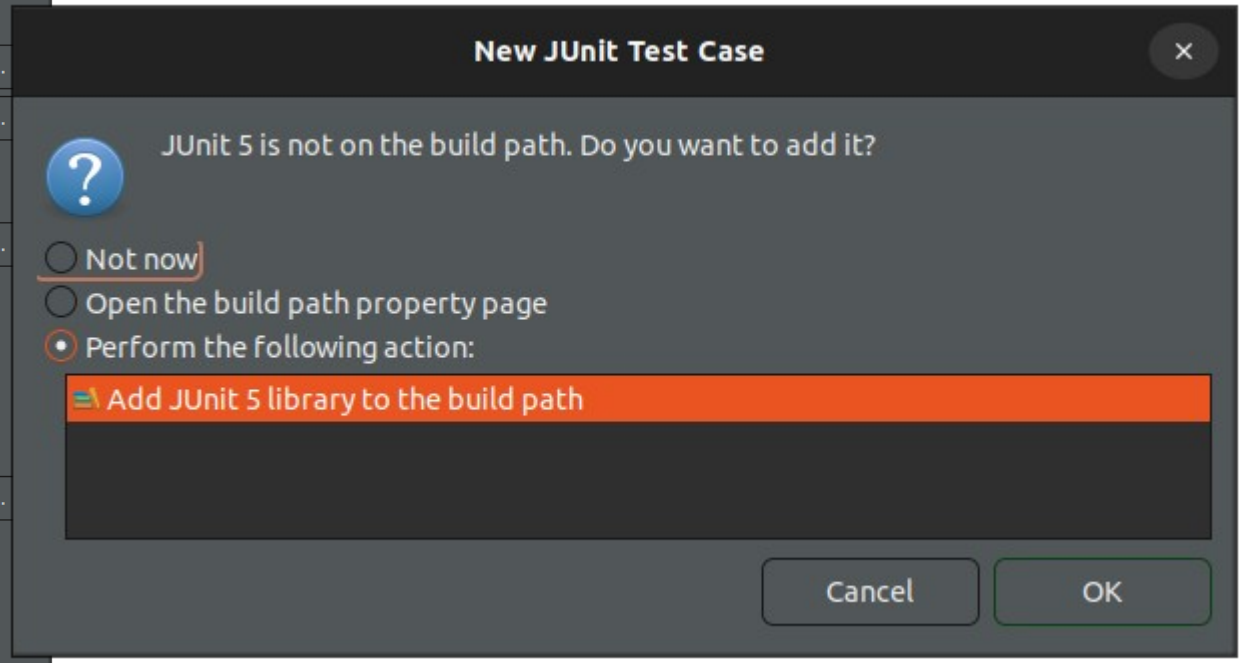


Definir classe de teste



Definir classe de teste





Resultado

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

class BissextoTest {

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
    }

    @BeforeEach
    void setUp() throws Exception {
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

Resultado

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

class BissextoTest {

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
    }

    @BeforeEach
    void setUp() throws Exception {
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

Indica que cada um desses métodos é um caso de teste

Pode-se usar qualquer nome mas aconselha-se usar um nome que indique quem está sendo testado

Resultado

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

class BissextoTest {

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
    }

    @BeforeEach
    void setUp() throws Exception {
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void testehBissextoAno1() {
        fail("Not yet implemented");
    }

}
```

Indica que cada um desses métodos é um caso de teste

Pode-se usar qualquer nome mas aconselha-se usar um nome que indique quem está sendo testado

Executando

- Ao executar esse arquivo de teste o JUnit identifica @Test e executa cada um
- Em cada caso de teste devemos chamar um método passando parâmetros que desejarmos
- Devemos também verificar se o resultado da chamada é aquele esperado

Bissexto.ehBissexto()

```java
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}
```

Bissexto.ehBissexto()

```
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}
```

Instanciação e chamada normal de método

Bissexto.ehBissexto()

```java
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}
```

Método que compara saída produzida com a esperada

Assertions

- assertEquals(expected, actual)
  - Dois valores (objetos) são iguais
- assertArrayEquals([] expecteds, [] actuals)
  - Dois arrays são iguais
- assertFalse/assertTrue(boolean)
- assertNull/assertNotNull(Object)
- assertSame/assertNotSame(Object, Object)
- fail(String)/fail()

Assertions

```java
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}

@Test
void testehBissextoAno1972() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1972);
    assertTrue(eb);
}
```

Asserts

- https://junit.org/junit5/docs/5.8.1/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html

```
assertEquals(Short expected, Short actual, Supplier<String> messageSupplier)
assertFalse(boolean condition)
assertFalse(boolean condition, String message)
assertFalse(boolean condition, Supplier<String> messageSupplier)
assertFalse(BooleanSupplier booleanSupplier)
assertFalse(BooleanSupplier booleanSupplier, String message)
assertFalse(BooleanSupplier booleanSupplier, Supplier<String> messageSupplier)
assertInstanceOf(Class<T> expectedType, Object actualValue)
assertInstanceOf(Class<T> expectedType, Object actualValue, String message)
assertInstanceOf(Class<T> expectedType, Object actualValue, Supplier<String
> messageSupplier)
assertIterableEquals(Iterable<?> expected, Iterable<?> actual)
assertIterableEquals(Iterable<?> expected, Iterable<?> actual, String message)
assertIterableEquals(Iterable<?> expected, Iterable<?> actual, Supplier<String
> messageSupplier)
assertLinesMatch(List<String> expectedLines, List<String> actualLines)
```

Ao executar

- JUnit mostra quais casos de teste falharam e a diferença existente.

Ao executar

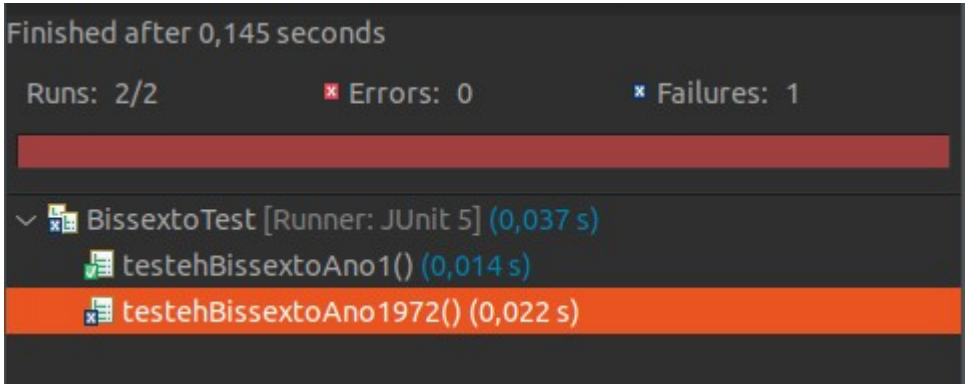
- JUnit mostra quais casos de teste falharam e a diferença existente.

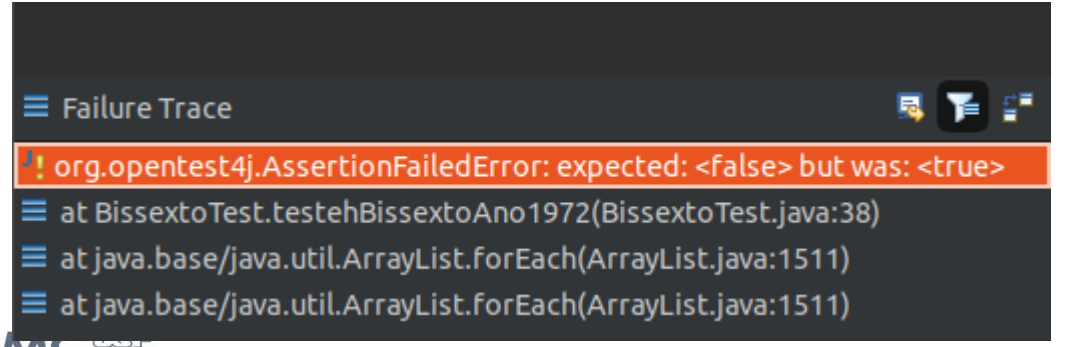
```
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}

@Test
void testehBissextoAno1972() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1972);
    assertEquals(false, eb);
}
```

Ao executar

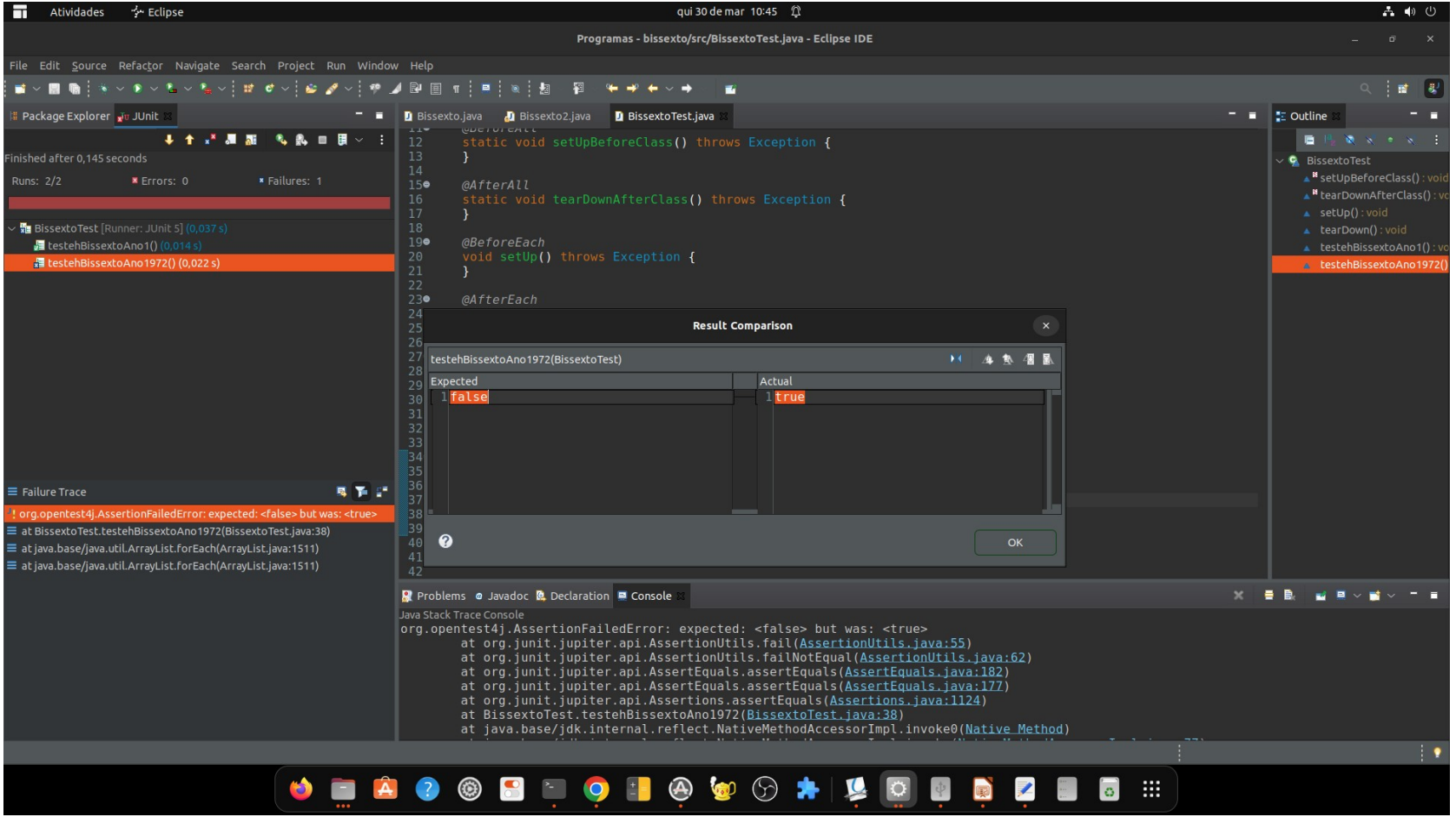
- JUnit mostra quais casos de teste falharam e a diferença existente.

```java
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}

@Test
void testehBissextoAno1972() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1972);
    assertEquals(false, eb);
}
```





Ao executar


Atividades
Eclipse
qui 30 de mar 10:45
Programas - bissexto/src/BissextoTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 0,145 seconds
Runs: 2/2
Errors: 0
Failures: 1
BissextoTest [Runner: JUnit 5] (0,037 s)
testehBissextoAno1() (0,014 s)
testehBissextoAno1972() (0,022 s)
Failure Trace
org.opentest4j.AssertionFailedError: expected: <false> but was: <true>
at BissextoTest.testehBissextoAno1972(BissextoTest.java:38)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
Bissexto.java
Bissexto2.java
BissextoTest.java
static void setUpBeforeClass() throws Exception {
}
@AfterAll
static void tearDownAfterClass() throws Exception {
}
@BeforeEach
void setUp() throws Exception {
}
@AfterEach
Result Comparison
testehBissextoAno1972(BissextoTest)
Expected
1 false
Actual
1 true
OK
Outline
BissextoTest
setUpBeforeClass() : void
tearDownAfterClass() : vo
setUp() : void
tearDown() : void
testehBissextoAno1() : vo
testehBissextoAno1972()
Problems
Javadoc
Declaration
Console
Java Stack Trace Console
org.opentest4j.AssertionFailedError: expected: <false> but was: <true>
at org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils.java:55)
at org.junit.jupiter.api.AssertionUtils.failNotEqual(AssertionUtils.java:62)
at org.junit.jupiter.api.AssertEquals.assertEquals(AssertEquals.java:182)
at org.junit.jupiter.api.AssertEquals.assertEquals(AssertEquals.java:177)
at org.junit.jupiter.api.Assertions.assertEquals(Assertions.java:1124)
at BissextoTest.testehBissextoAno1972(BissextoTest.java:38)
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

Assert para exceções

- Alguns casos de teste devem gerar exceções
- Por exemplo, quando o ano passado como parâmetro é negativo
- Existe uma forma específica para fazer a verificação do resultado

Assert para exceções

```
@Test
void testehBissextoAnoNegativo() {
    Bissexto b1 = new Bissexto();
    Exception ex = assertThrows(IllegalArgumentException.class, () -> {
        b1.ehBissexto(-1);

    });
    String expectedMessage = "Ano Inválido";

    assertEquals(expectedMessage, ex.getMessage());
}
```

Assert para exceções

```
@Test
void testehBissextoAnoNegativo() {
    Bissexto b1 = new Bissexto();
    Exception ex = assertThrows(IllegalArgumentException.class, () -> {
        b1.ehBissexto(-1);

    });
    String expectedMessage = "Ano Inválido";

    assertEquals(expectedMessage, ex.getMessage());
}
```

Outras anotaçõs

- @BeforeEach
- Um método anotado é executado imediatamente antes de **cada** caso de teste
- Útil para preparar o ambiente para o caso de teste
- Criar um objeto, por exemplo
- @AfterEach

@BeforeEach

```java
Bissexto b1;
@BeforeEach
void setUp() throws Exception {
    b1 = new Bissexto();
}

@AfterEach
void tearDown() throws Exception {
    b1 = null;
}

@Test
void testehBissextoAno1() {
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}
```

@BeforeEach

```java
Bissexto b1;
@BeforeEach
void setUp() throws Exception {
    b1 = new Bissexto();
}

@AfterEach
void tearDown() throws Exception {
    b1 = null;
}

@Test
void testehBissextoAno1() {
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}
```

```java
@Test
void testehBissextoAno1() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1);
    assertEquals(false, eb);
}

@Test
void testehBissextoAno1972() {
    Bissexto b1 = new Bissexto();
    boolean eb = b1.ehBissexto(1972);
    assertTrue(eb);
}
```

@BeforeAll

- Usado para esexutar alguma inicialização antes de qualquer teste
- Antes de @BeforeEach
- Método é estático

@BeforeClass

```
static Bissexto b2;
@BeforeAll
static void setUpBeforeClass() {
    b2 = new Bissexto();
}

@AfterAll
static void tearDownAfterClass() {
    b2 = null;
}
```

@BeforeClass

```
static Bissexto b2;
@BeforeAll
static void setUpBeforeClass() {
    b2 = new Bissexto();
}

@AfterAll
static void tearDownAfterClass() {
    b2 = null;
}
```

```
@Test
void testehBissextoAno1() {
    boolean eb = b2.ehBissexto(1);
    assertEquals(false, eb);
}
```

Outras anotações

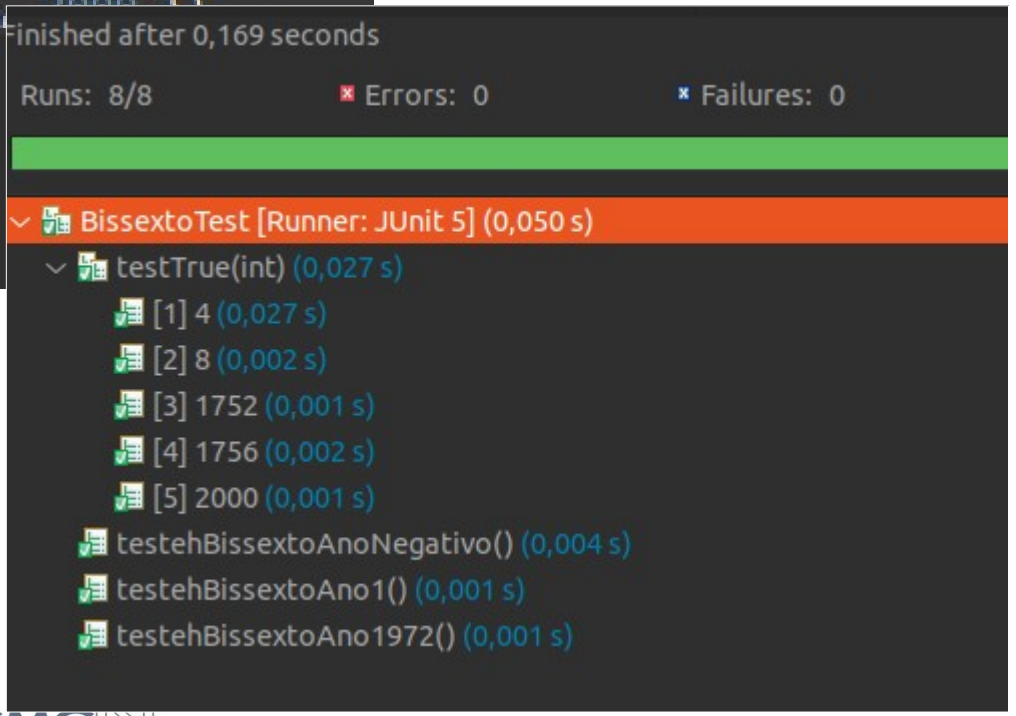
- Existem várias outras anotações
- Inclusive outros tipos de métodos de teste
- Vale a pena estudar o framework e aprender para que servem

Teste parametrizado

```
@ParameterizedTest
@ValueSource(ints = {4, 8, 1752, 1756, 2000 })
void testTrue(int ano) {
    assertTrue(b1.ehBissexto(ano));

}
```

Teste parametrizado

```java
@ParameterizedTest
@ValueSource(ints = {4, 8, 1752, 1756, 2000 })
void testTrue(int ano) {
    assertTrue(b1.ehBissexto(ano));

}
```



Repetições

```java
@RepeatedTest(5)
void repeatedTest1797to1802(RepetitionInfo repetitionInfo) {
    assertFalse(b1.ehBissexto(1797 + repetitionInfo.getCurrentRepetition()));
}
```

Repetições

```java
@RepeatedTest(5)
void repeatedTest1797to1802(RepetitionInfo repetitionInfo) {
    assertFalse(b1.ehBissexto(1797 + repetitionInfo.getCurrentRepetition()));
}
```

Before quais?

- Antes de cada repetição, executa-se o @BeforeEach

```
Bissexto b1;
static int cont = 0;
@BeforeEach
void setUp() throws Exception {
    b1 = new Bissexto();
    cont += 1;
    System.out.println("Executando @BeforEach " + cont);
}
```

Before quais?

- Antes de cada repetição, executa-se o @BeforeEach

```java
Bissexto b1;
static int cont = 0;
@BeforeEach
void setUp() throws Exception {
    b1 = new Bissexto();
    cont += 1;
    System.out.println("Executando @BeforEach " + cont
}
```

```
70⊖    @RepeatedTest(5)
Problems  @ Javadoc  Declaration  Console
<terminated> BissextoTest [JUnit] /usr/lib/jvm/java-17-openjdk-amd64/
Executando @BeforEach 1
Executando @BeforEach 2
Executando @BeforEach 3
Executando @BeforEach 4
Executando @BeforEach 5
Executando @BeforEach 6
Executando @BeforEach 7
Executando @BeforEach 8
Executando @BeforEach 9
Executando @BeforEach 10
Executando @BeforEach 11
Executando @BeforEach 12
Executando @BeforEach 13
```