JUNIT

testing framework for Java and the JVM

GRUPO

Vinicius Andreata Dumont

Luiz Henrique Oliveira de Assis

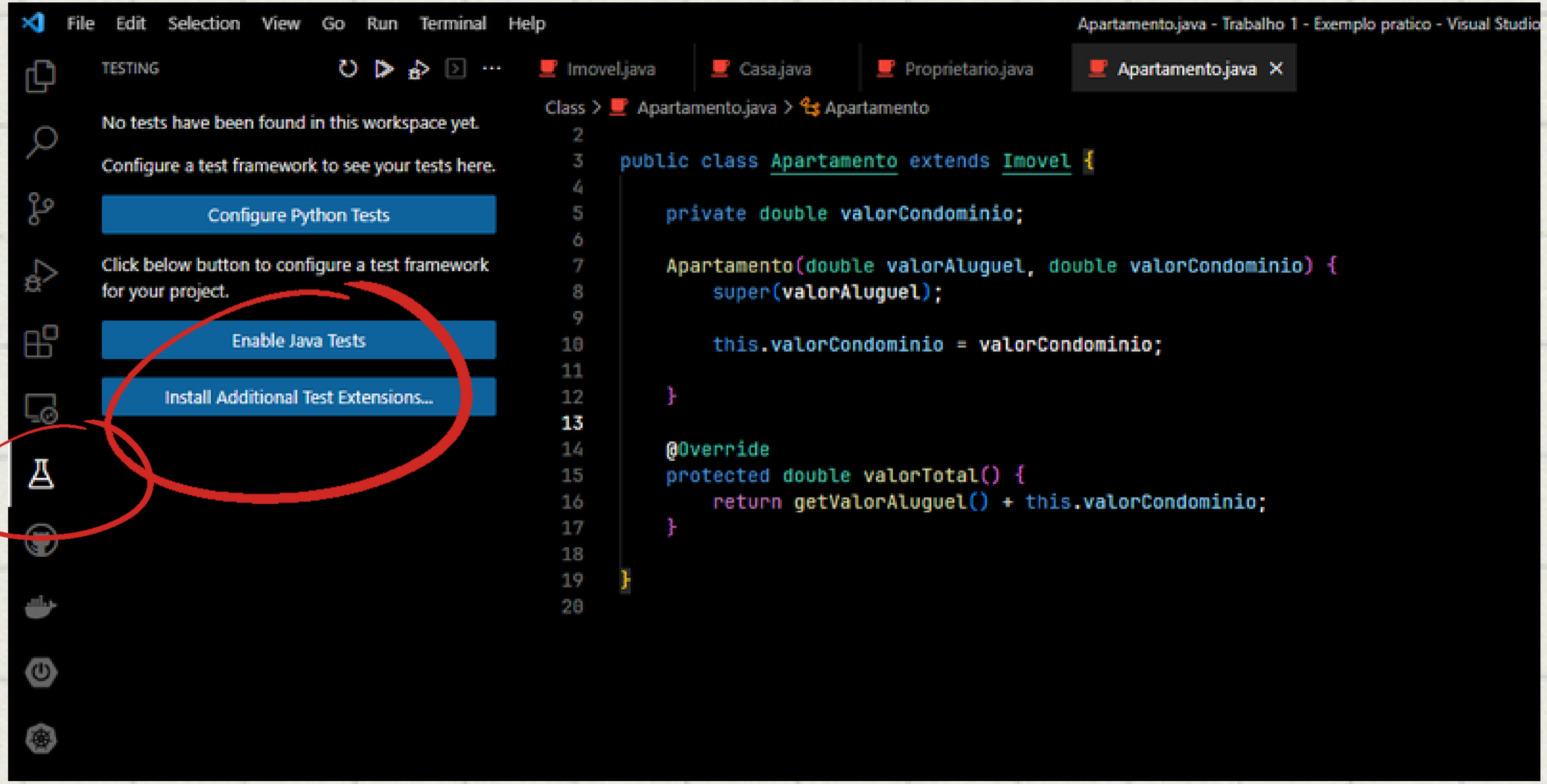
Pedro Miguel Moraes Durço

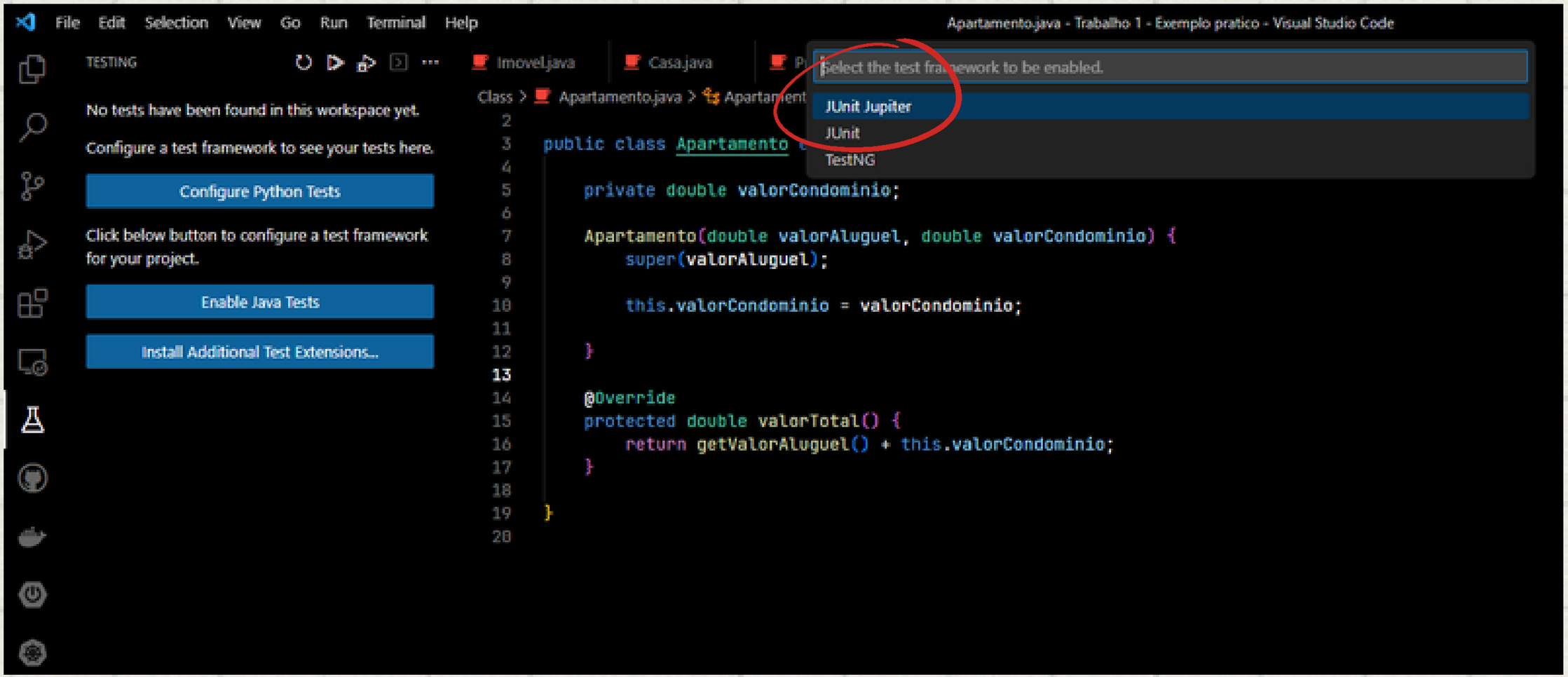
Victor Goddard

Eberson Soares Sena

COMO UTILIZAR ?

Via interface VScode

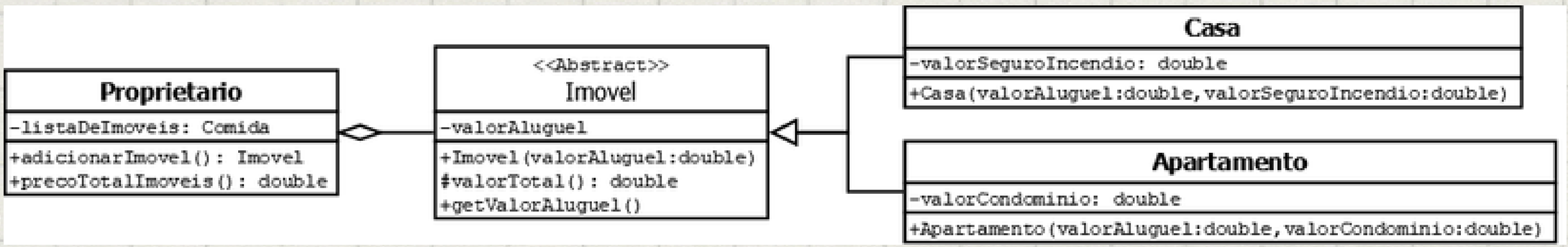

File Edit Selection View Go Run Terminal Help
Apartamento.java - Trabalho 1 - Exemplo pratico - Visual Studio
TESTING
No tests have been found in this workspace yet.
Configure a test framework to see your tests here.
Configure Python Tests
Click below button to configure a test framework for your project.
Enable Java Tests
Install Additional Test Extensions...
Imovel.java
Casa.java
Proprietario.java
Apartamento.java
Class > Apartamento.java > Apartamento
public class Apartamento extends Imovel {
private double valorCondominio;
Apartamento(double valorAluguel, double valorCondominio) {
super(valorAluguel);
this.valorCondominio = valorCondominio;
}
@Override
protected double valorTotal() {
return getValorAluguel() + this.valorCondominio;
}
}


File Edit Selection View Go Run Terminal Help
Apartamento.java - Trabalho 1 - Exemplo pratico - Visual Studio Code
TESTING
No tests have been found in this workspace yet.
Configure a test framework to see your tests here.
Configure Python Tests
Click below button to configure a test framework for your project.
Enable Java Tests
Install Additional Test Extensions...
Imovel.java
Casa.java
Class > Apartamento.java
Select the test framework to be enabled.
JUnit Jupiter
JUnit
TestNG
public class Apartamento
private double valorCondominio;
Apartamento(double valorAluguel, double valorCondominio) {
super(valorAluguel);
this.valorCondominio = valorCondominio;
}
@Override
protected double valorTotal() {
return getValorAluguel() + this.valorCondominio;
}
}

ESTRUTURA DO TESTE

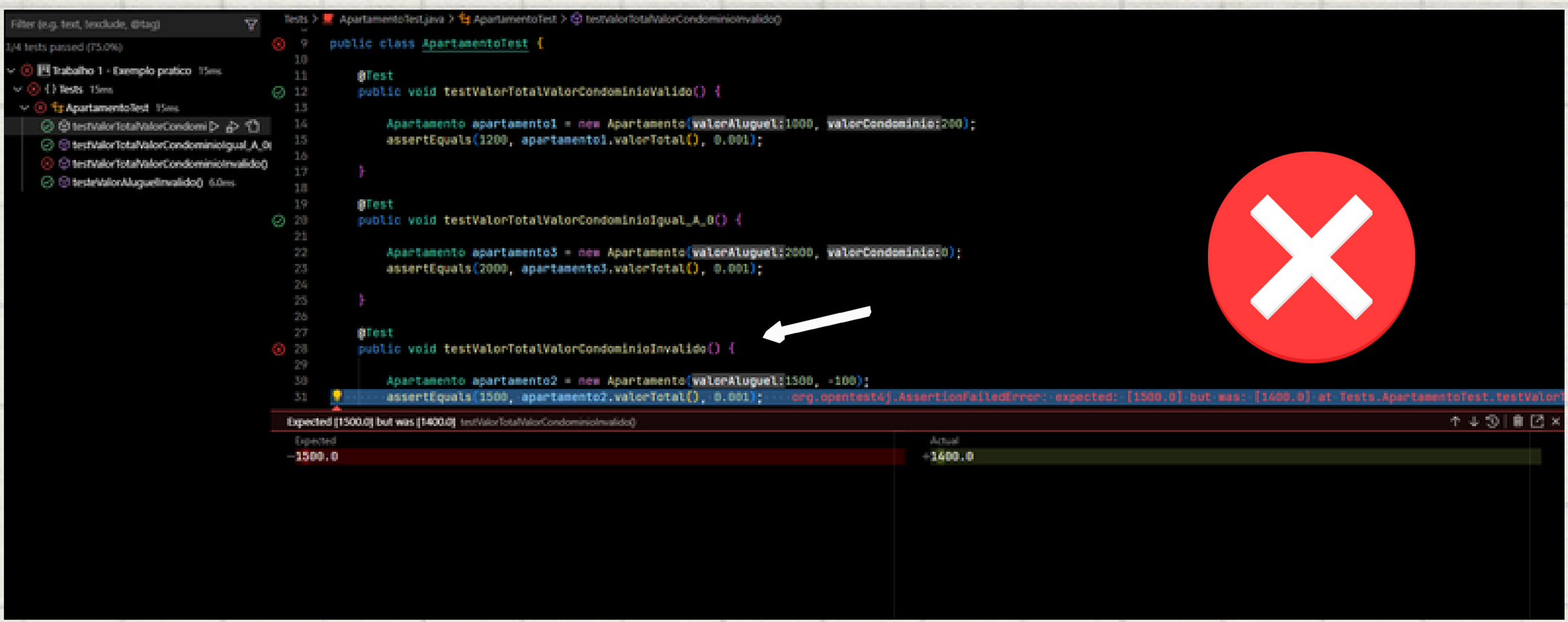
```java
@Test // Annotation padrão
public void teste(){ // publico e sem retorno
    assertEquals("Expectativa", "Metodo"); // parametros (pode variar)
}
```

CLASSES DO SISTEMA



TESTANDO CLASSE APARTAMENTO

```java
public class ApartamentoTest {

    @Test
    public void testValorTotalValorCondominioValido() {

        Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
        assertEquals(1200, apartamento1.valorTotal(), 0.001);

    }

    @Test
    public void testValorTotalValorCondominioIgual_A_0() {

        Apartamento apartamento3 = new Apartamento(valorAluguel:2000, valorCondominio:0);
        assertEquals(2000, apartamento3.valorTotal(), 0.001);

    }

    @Test
    public void testValorTotalValorCondominioInvalido() {

        Apartamento apartamento2 = new Apartamento(valorAluguel:1500, -100);
        assertEquals(1500, apartamento2.valorTotal(), 0.001);

    }

    @Test
    public void testeValorAluguelInvalido() {

        Apartamento apartamento2 = new Apartamento(-32, valorCondominio:100);
        assertEquals(100, apartamento2.valorTotal(), 0.001);

    }
}
```


Filter (e.g. text, !exclude, @tag)
3/4 tests passed (75.0%)
Trabalho 1 - Exemplo pratico 15ms
Tests 15ms
ApartamentoTest 15ms
testValorTotalValorCondomi
testValorTotalValorCondominioIgual_A_0()
testValorTotalValorCondominioInvalido()
testeValorAluguelInvalido() 6.0ms
Tests > ApartamentoTest.java > ApartamentoTest > testValorTotalValorCondominioInvalido()
public class ApartamentoTest {
@Test
public void testValorTotalValorCondominioValido() {
Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
assertEquals(1200, apartamento1.valorTotal(), 0.001);
}
@Test
public void testValorTotalValorCondominioIgual_A_0() {
Apartamento apartamento3 = new Apartamento(valorAluguel:2000, valorCondominio:0);
assertEquals(2000, apartamento3.valorTotal(), 0.001);
}
@Test
public void testValorTotalValorCondominioInvalido() {
Apartamento apartamento2 = new Apartamento(valorAluguel:1500, -100);
assertEquals(1500, apartamento2.valorTotal(), 0.001);
org.opentest4j.AssertionFailedError: expected: [1500.0] but was: [1400.0] at Tests.ApartamentoTest.testValor
Expected [1500.0] but was [1400.0] testValorTotalValorCondominioInvalido()
Expected
-1500.0
Actual
+1400.0

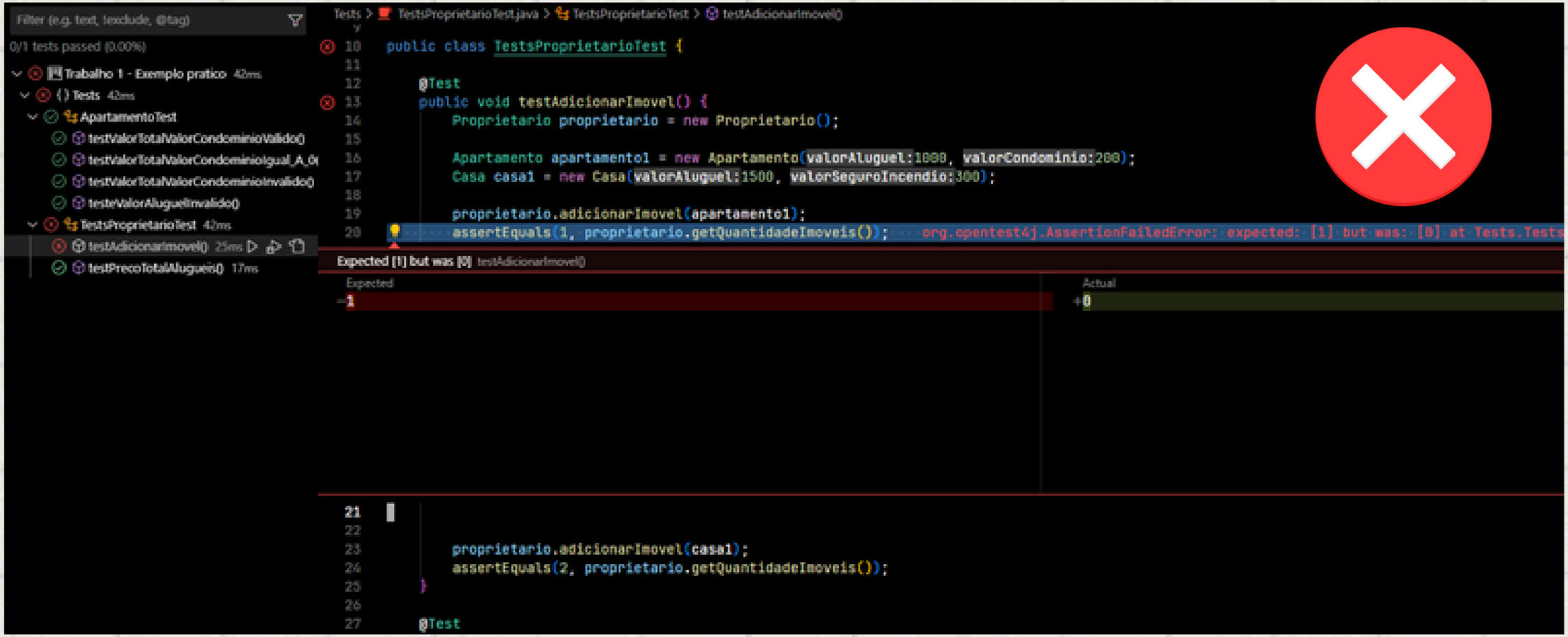
```java
public class Apartamento extends Imovel {

    private double valorCondominio;

    public Apartamento(double valorAluguel, double valorCondominio) {

        super(valorAluguel);

        this.valorCondominio = valorCondominio;

    }

    @Override
    public double valorTotal() {
        return getValorAluguel() + this.valorCondominio;
    }

}
```

```java
package Class;
public class Apartamento extends Imovel{

    private double valorCondominio;

    public Apartamento(double valorAluguel, double valorCondominio){
        super(valorAluguel);

        if (valorCondominio > 0)
            this.valorCondominio = valorCondominio;
        else
            this.valorCondominio = 0;

    }

    @Override
    public double valorTotal() {
        return getValorAluguel() + this.valorCondominio;
    }

}
```

```java
public class ApartamentoTest {

    @Test
    public void testValorTotalValorCondominioValido() {

        Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
        assertEquals(1200, apartamento1.valorTotal(), 0.001);

    }

    @Test
    public void testValorTotalValorCondominioIgual_A_0() {

        Apartamento apartamento3 = new Apartamento(valorAluguel:2000, valorCondominio:0);
        assertEquals(2000, apartamento3.valorTotal(), 0.001);

    }

    @Test
    public void testValorTotalValorCondominioInvalido() {

        Apartamento apartamento2 = new Apartamento(valorAluguel:1500, -100);
        assertEquals(1500, apartamento2.valorTotal(), 0.001);

    }

    @Test
    public void testeValorAluguelInvalido() {

        Apartamento apartamento2 = new Apartamento(-32, valorCondominio:100);
        assertEquals(100, apartamento2.valorTotal(), 0.001);

    }
}
```

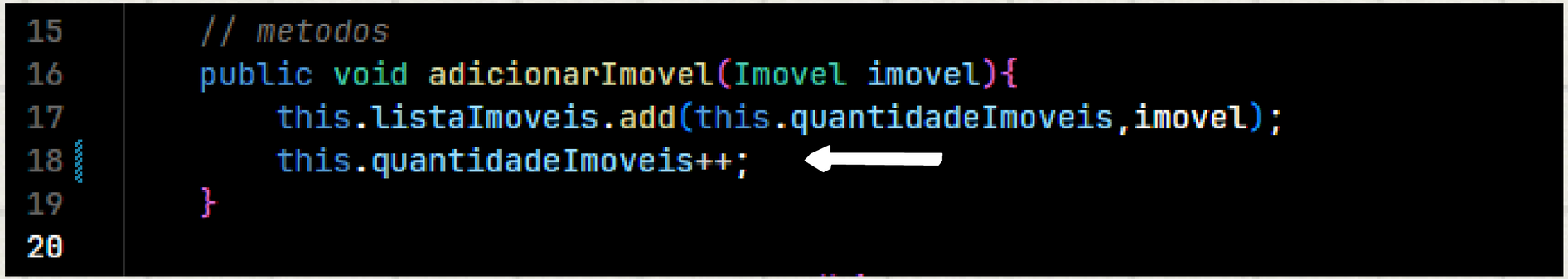
TESTANDO CLASSE PROPRIETARIO

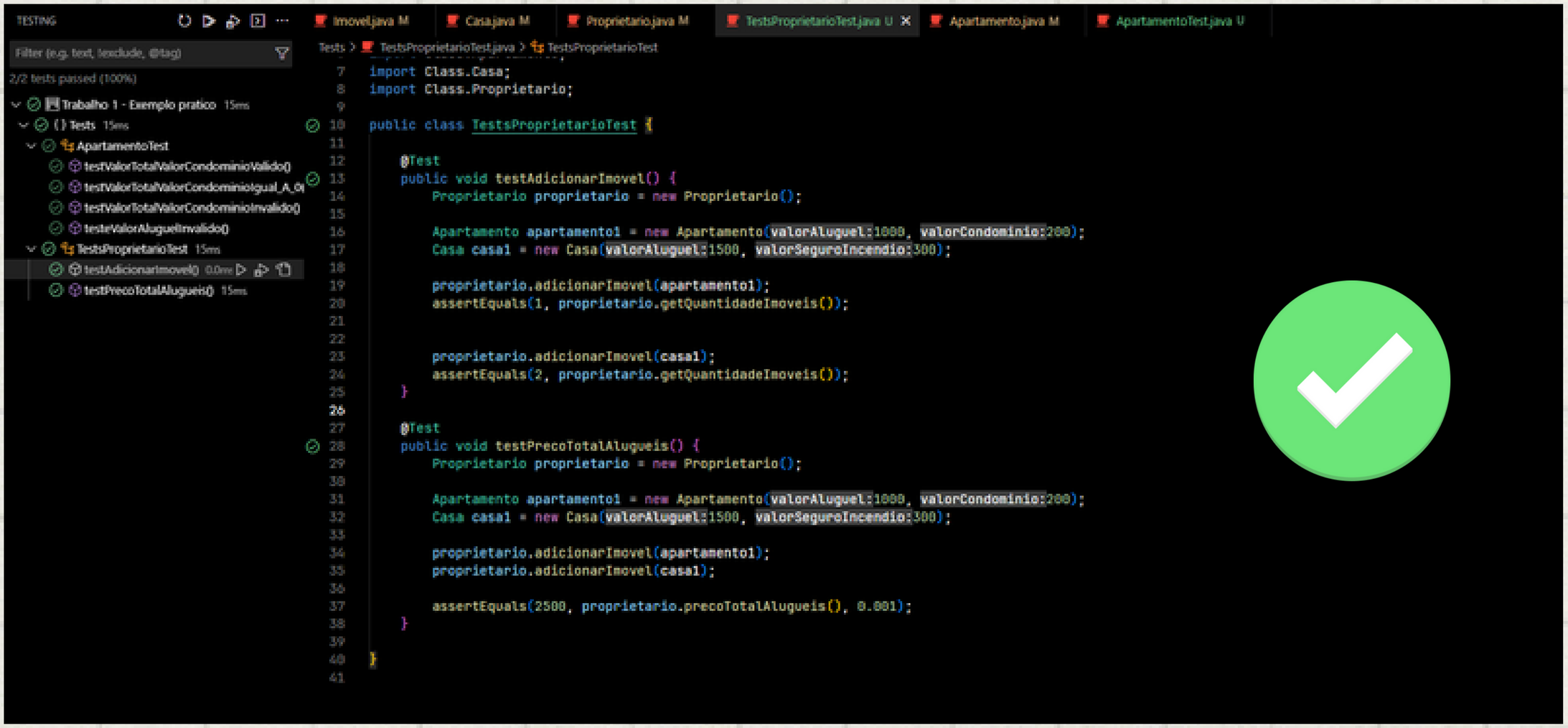
```java
@Test
public void testAdicionarImovel() {
    Proprietario proprietario = new Proprietario();

    Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
    Casa casa1 = new Casa(valorAluguel:1500, valorSeguroIncendio:300);

    proprietario.adicionarImovel(apartamento1);
    assertEquals(1, proprietario.getQuantidadeImoveis());


    proprietario.adicionarImovel(casa1);
    assertEquals(2, proprietario.getQuantidadeImoveis());
}

@Test
public void testPrecoTotalAlugueis() {
    Proprietario proprietario = new Proprietario();

    Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
    Casa casa1 = new Casa(valorAluguel:1500, valorSeguroIncendio:300);

    proprietario.adicionarImovel(apartamento1);
    proprietario.adicionarImovel(casa1);

    assertEquals(2500, proprietario.precoTotalAlugueis(), 0.001);
}
```


Filter (e.g. text, !exclude, @tag)
0/1 tests passed (0.00%)
Trabalho 1 - Exemplo pratico 42ms
Tests 42ms
ApartamentoTest
testValorTotalValorCondominioValido()
testValorTotalValorCondominioIgual_A_0
testValorTotalValorCondominioInvalido()
testeValorAluguelInvalido()
TestsProprietarioTest 42ms
testAdicionarImovel() 25ms
testPrecoTotalAlugueis() 17ms
Tests > TestsProprietarioTest.java > TestsProprietarioTest > testAdicionarImovel()
10 public class TestsProprietarioTest {
12 @Test
13 public void testAdicionarImovel() {
14 Proprietario proprietario = new Proprietario();
16 Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
17 Casa casa1 = new Casa(valorAluguel:1500, valorSeguroIncendio:300);
19 proprietario.adicionarImovel(apartamento1);
20 assertEquals(1, proprietario.getQuantidadeImoveis()); org.opentest4j.AssertionFailedError: expected: [1] but was: [0] at Tests.Tests
Expected [1] but was [0] testAdicionarImovel()
Expected
-1
Actual
+0
21
23 proprietario.adicionarImovel(casa1);
24 assertEquals(2, proprietario.getQuantidadeImoveis());
25 }
27 @Test

Proprietario.java

```java
package Class;
import java.util.ArrayList;

public class Proprietario {

    // atributos
    private ArrayList<Imovel> listaImoveis = new ArrayList<Imovel>();
    private int quantidadeImoveis;

    // construtor
    public Proprietario() {
        this.quantidadeImoveis = 0;
    }

    // metodos
    public void adicionarImovel(Imovel imovel){
        this.listaImoveis.add(this.quantidadeImoveis,imovel);
    }

    public double precoTotalAlugueis(){
        double total = 0;

        for (Imovel imovel : listaImoveis) {
            total += imovel.getValorAluguel();
        }

        return total;
    }

    public int getQuantidadeImoveis() {
        return quantidadeImoveis;
    }
}
```


15 // metodos
16 public void adicionarImovel(Imovel imovel){
17 this.listaImoveis.add(this.quantidadeImoveis,imovel);
18 this.quantidadeImoveis++;
19 }
20


TESTING
Filter (e.g. text, !exclude, @tag)
2/2 tests passed (100%)
Trabalho 1 - Exemplo pratico 15ms
Tests 15ms
ApartamentoTest
testValorTotalValorCondominioValido()
testValorTotalValorCondominioIgual_A_0
testValorTotalValorCondominioInvalido()
testeValorAluguelInvalido()
TestsProprietarioTest 15ms
testAdicionarImovel() 0.0ms
testPrecoTotalAlugueis() 15ms
ImovelJava M
CasaJava M
ProprietarioJava M
TestsProprietarioTest.java U
ApartamentoJava M
ApartamentoTest.java U
Tests > TestsProprietarioTest.java > TestsProprietarioTest
import Class.Casa;
import Class.Proprietario;
public class TestsProprietarioTest {
@Test
public void testAdicionarImovel() {
Proprietario proprietario = new Proprietario();
Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
Casa casa1 = new Casa(valorAluguel:1500, valorSeguroIncendio:300);
proprietario.adicionarImovel(apartamento1);
assertEquals(1, proprietario.getQuantidadeImoveis());
proprietario.adicionarImovel(casa1);
assertEquals(2, proprietario.getQuantidadeImoveis());
}
@Test
public void testPrecoTotalAlugueis() {
Proprietario proprietario = new Proprietario();
Apartamento apartamento1 = new Apartamento(valorAluguel:1000, valorCondominio:200);
Casa casa1 = new Casa(valorAluguel:1500, valorSeguroIncendio:300);
proprietario.adicionarImovel(apartamento1);
proprietario.adicionarImovel(casa1);
assertEquals(2500, proprietario.precoTotalAlugueis(), 0.001);
}
}

FIM

Obrigado pela atenção!