

THE  
DEVELOPER'S  
CONFERENCE

**Trilha**  
***Extreme Programming***  
***#TDCPOA19***



**LEANDRO AKIRA IWASAKI**



51 98194-8443



leandro-akira-iwasaki



**RODRIGO MURARI SEVERO**



51 99714-6600



rmsevero



@rodrigomsevero



quaslati.wordpress.com



# Conceitos básicos TDD (*Test Driven Development*)

Que tal jogar o jogo dos números mágicos

Vamos ver na prática?

1. Adicione um teste

2. Execute o teste e observe o resultado

3. Escreva o código

4. Execute os testes automatizados

5. Refatore os códigos

# TDD – Conceitos

TDD

# TDD...

pequenos ciclos de repetições

um teste criado antes para cada funcionalidade

teste criado inicialmente falha

não temos a implementação da funcionalidade

em seguida, implementamos a funcionalidade para fazer o teste passar

refactoring do código implementado

# O QUE GANHAMOS...

Código mais limpo, já que escrevemos códigos simples para o teste passar

Código menos acoplado

Feedback rápido sobre a nova funcionalidade e sobre as outras funcionalidades existentes no sistema

Segurança no *Refactoring* pois podemos ver o que estamos ou não afetando

Segurança na correção de bugs

Maior produtividade já que o DEV encontra menos bugs e não desperdiça tempo com debugs

```
static void PrintCard()
    Card aCard
    new Card(8, "Clubs");
    Console.WriteLine("Card {0} of {1}", aCard.Number, aCard.Suit);
}
```

# Vamos Jogar?

# Escolha um número

## Em quais cartas o número esta?

A			
1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31
33	35	37	39
41	43	45	47
49	51	53	55
57	59	61	63

B			
2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31
34	35	38	39
42	43	46	47
50	51	54	55
58	59	62	63

C			
4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31
36	37	38	39
44	45	46	47
52	53	54	55
60	61	62	63

D			
8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31
40	41	42	43
44	45	46	47
56	57	58	59
60	61	62	63

E			
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63

F			
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63

## Mágico vai adivinhar...



Como o “mágico” descobre o número escolhido ?

No exemplo o número escolhido foi o **35** que está nas cartas A,B,F.

O “mágico” soma o número superior esquerdo de cada carta

$$1+2+32 = \mathbf{35}$$

A				B				C				D				E				F			
<b>1</b>	3	5	7	<b>2</b>	3	6	7	4	5	6	7	8	9	10	11	16	17	18	19	<b>32</b>	33	34	<b>35</b>
9	11	13	15	10	11	14	15	12	13	14	15	12	13	14	15	20	21	22	23	36	37	38	39
17	19	21	23	18	19	22	23	20	21	22	23	24	25	26	27	24	25	26	27	40	41	42	43
25	27	29	31	26	27	30	31	28	29	30	31	28	29	30	31	28	29	30	31	44	45	46	47
33	<b>35</b>	37	39	34	<b>35</b>	38	39	36	37	38	39	40	41	42	43	48	49	50	51	48	49	50	51
41	43	45	47	42	43	46	47	44	45	46	47	44	45	46	47	52	53	54	55	52	53	54	55
49	51	53	55	50	51	54	55	52	53	54	55	56	57	58	59	56	57	58	59	56	57	58	59
57	59	61	63	58	59	62	63	60	61	62	63	60	61	62	63	60	61	62	63	60	61	62	63

1. Adicione  
um teste

## User story

Como um jogador que gosta de jogos de desafio.

Quero jogar o jogo dos números mágicos com 6 cartas

Para se divertir

2. Execute o teste e observe o resultado

3. Escreva o código

4. Execute os testes automatizados

5. Execute os códigos

1. Adicione  
um teste

2. Execute o  
teste e observe  
o resultado

3. Escreva  
o código

4. Execute  
os testes  
automatizados

5. Refatore

**Cenário 1:**

**Dado que O jogador escolheu o número e indicou em todas as cartas que ele está.**

**Quando o sistema fizer a soma dos primeiros números das cartas**

**Então irá revelar o número que o jogador escolheu**

**Critérios de Aceitação:**

TDD

Criando o projeto

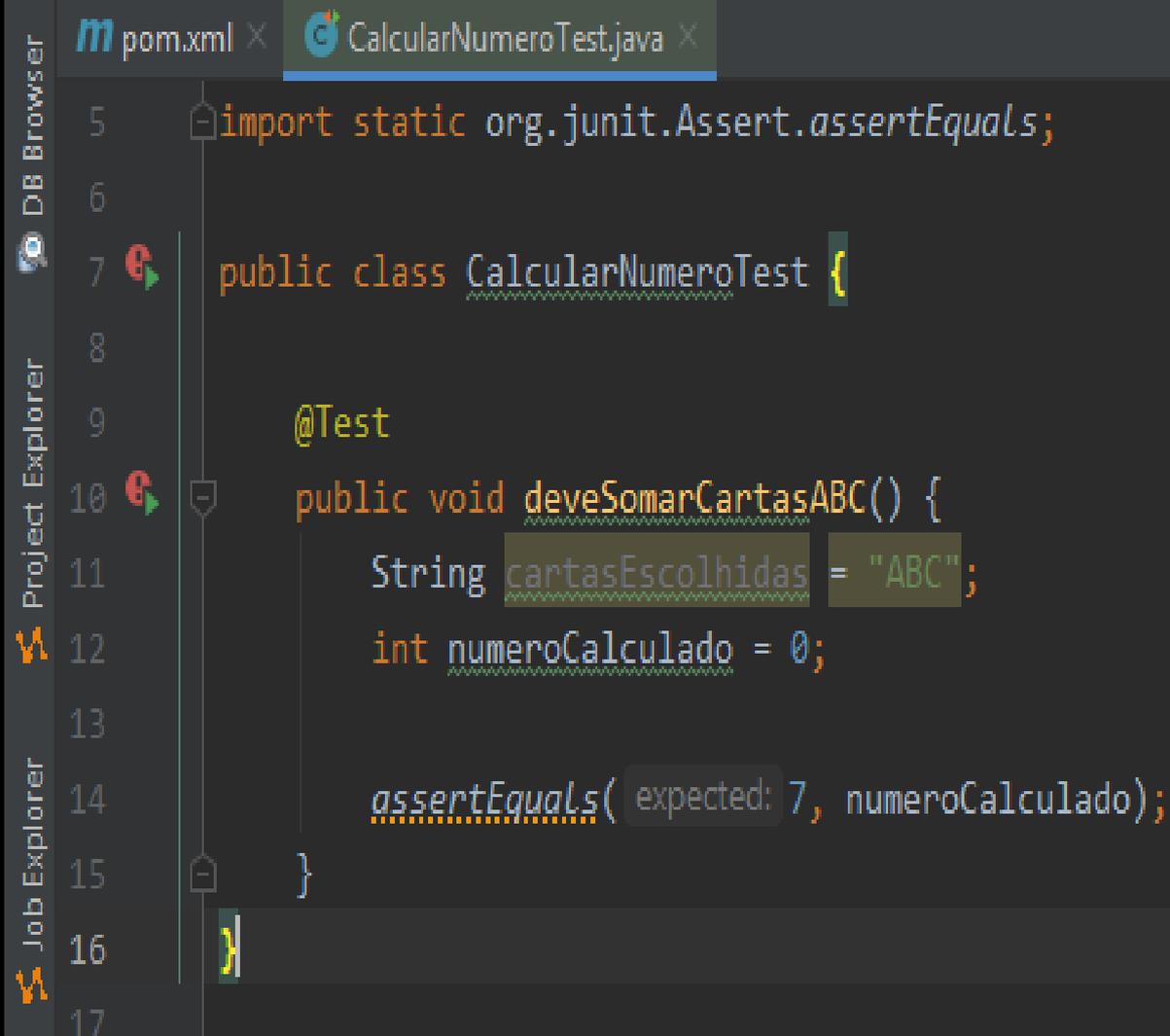
Iniciando pelos testes

The screenshot shows the Project Explorer of an IDE. The project is named 'tdd' and is located at 'C:\Temp\tdd'. The structure is as follows:

- tdd (C:\Temp\tdd)
  - .idea
  - src
    - main
      - java
      - resources
    - test
      - java (highlighted in green)
      - br.com.sicredi.tdd (highlighted in blue)
  - target
  - pom.xml
  - tdd.iml
  - External Libraries
  - Scratches and Consoles

The 'CalculaNúmeroTest' class is highlighted in blue within the 'br.com.sicredi.tdd' package.

## Criando o primeiro teste



```
m pom.xml x CalcularNumeroTest.java x
5 import static org.junit.Assert.assertEquals;
6
7 public class CalcularNumeroTest {
8
9     @Test
10    public void deveSomarCartasABC() {
11        String cartasEscolhidas = "ABC";
12        int numeroCalculado = 0;
13
14        assertEquals("expected: 7, numeroCalculado");
15    }
16 }
17
```



Teste falhando

Run: ◀ CalcularNumeroTest X

▶ ✓ 0 ↓<sup>a</sup> ↓: ⌵ ⌶ ↑ ↓ 🔍 ↶ » ⚠ Tests failed: 1 of 1 test - 22 ms

9 ▼ ⚠ CalcularNumeroTest (br.com.sicredi.tdd) 22 ms

⚠ deveSomarCartasABC 22 ms

java.lang.AssertionError:  
Expected :7  
Actual :0  
[<Click to see difference>](#)

+ <1 internal call>

DB Execution Console Terminal SonarLint Build 0: Messages 4: Run 6: T

Inserindo os testes da  
implementação

Código não compila

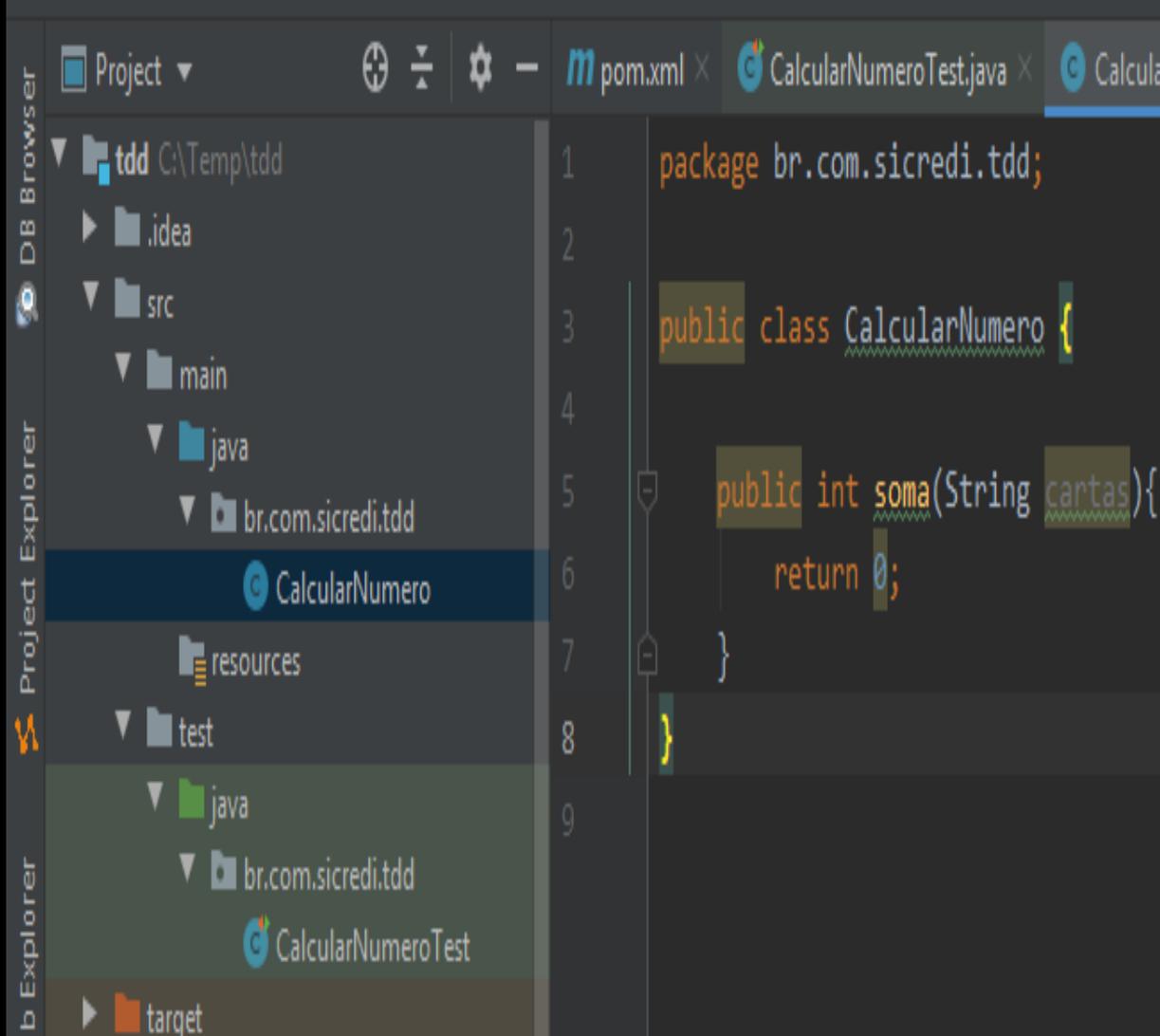
The screenshot shows an IDE window with two tabs: 'pom.xml' and 'CalcularNumeroTest.java'. The code in 'CalcularNumeroTest.java' is as follows:

```
5 import static org.junit.Assert.assertEquals;
6
7 public class CalcularNumeroTest {
8
9     @Test
10    public void deveSomarCartasABC() {
11        String cartasEscolhidas = "ABC";
12        CalcularNumero calcularNumero = new CalcularNumero();
13        int numeroCalculado = calcularNumero.soma(cartasEscolhidas);
14
15        assertEquals( expected: 7, numeroCalculado);
16    }
17 }
```

The IDE's Messages panel at the bottom shows the following build output:

```
Build
Information: java: Errors occurred while compiling module 'tdd'
Information: javac 11.0.2 was used to compile java sources
Information: 21/10/2019 20:54 - Build completed with 2 errors and 0 warnings in 2 s 458 ms
C:\Temp\tdd\src\test\java\br\com\sicredi\tdd\CalcularNumeroTest.java
Error:(12, 9) java: cannot find symbol
symbol: class CalcularNumero
location: class br.com.sicredi.tdd.CalcularNumeroTest
Error:(12, 45) java: cannot find symbol
```

Criando a classe responsável pela implementação



The screenshot shows an IDE interface with a project explorer on the left and a code editor on the right. The project explorer is divided into three panes: DB Browser, Project Explorer, and Explorer. The Project Explorer pane shows a project structure with folders for 'tdd', 'src', 'main', 'java', and 'br.com.sicredi.tdd'. The 'CalculaNumero' class is highlighted in the 'br.com.sicredi.tdd' folder. The Explorer pane shows the 'test' folder with a 'CalculaNumeroTest' class. The code editor shows the following Java code:

```
1 package br.com.sicredi.tdd;
2
3 public class CalculaNumero {
4
5     public int soma(String cartas){
6         return 0;
7     }
8 }
9
```

# Código compila e teste falha

The screenshot shows an IDE with three tabs: pom.xml, CalcularNumeroTest.java, and CalcularNumero.java. The main editor displays the following Java code for CalcularNumeroTest.java:

```
public class CalcularNumeroTest {  
  
    @Test  
    public void deveSomarCartasABC() {  
        String cartasEscolhidas = "ABC";  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(cartasEscolhidas);  
  
        assertEquals( expected: 7, numeroCalculado);  
    }  
}
```

Below the code editor, the Run console shows the execution of the test. The test failed, and the error message is displayed:

```
Run: CalcularNumeroTest x  
Tests failed: 1 of 1 test - 30 ms  
CalcularNumeroTest (br.com.sicredi.tdd) 30 ms  
  deveSomarCartasABC 30 ms  
    java.lang.AssertionError:  
      Expected :7  
      Actual   :0  
      <Click to see difference>
```

Fazendo o teste passar

Implementado o valor  
esperado pelo teste

The screenshot shows an IDE interface with the following components:

- Project Explorer:** Shows a project structure with folders `.idea`, `src`, `main`, `test`, and `target`. Under `src/main/java` is `br.com.sicredi.tdd` containing `CalculadorNumero`. Under `src/test/java` is `br.com.sicredi.tdd` containing `CalculadorNumeroTest`.
- Code Editor:** Displays the source code for `CalculadorNumero.java`:

```
1 package br.com.sicredi.tdd;  
2  
3 public class CalculadorNumero {  
4  
5     public int soma(String cartas){  
6         return 7;  
7     }  
8 }  
9
```
- Run Console:** Shows the execution of `CalculadorNumeroTest` with the message: `Tests passed: 1 of 1 test - 8 ms`.
- Test Results:** A table showing the test results:

Test Name	Duration
CalculadorNumeroTest (br.com.sicredi.tdd)	8 ms
deveSomarCartasABC	8 ms

# Implementando a regra de negócio

```
m pom.xml x CalculadorNumeroTest.java x CalculadorNumero.java x
3 public class CalcularNumero {
4
5 @ public int soma(String cartas){
6     int numeroCalculado = 0;
7     if(cartas.contains("A")){
8         numeroCalculado = numeroCalculado + 1;
9     }
10    if(cartas.contains("B")){
11        numeroCalculado = numeroCalculado + 2;
12    }
13    if(cartas.contains("C")){
14        numeroCalculado = numeroCalculado + 4;
15    }
16    return numeroCalculado;
17 }
18 }
```

Run: CalculadorNumeroTest x

Tests passed: 1 of 1 test -

CalcularNumeroTest (br.com.sicredi.tdd)	9 ms	"C:\Program Files\
deveSomarCartasABC	9 ms	

Criando testes para as demais cartas

Teste falhando

```
m pom.xml x  CalculadorNumeroTest.java x  CalculadorNumero.java x
16      }
17
18      @Test
19      public void deveSomarCartasABCDEF() {
20          String cartasEscolhidas = "ABCDEF";
21          CalculadorNumero calcularNumero = new CalculadorNumero();
22          int numeroCalculado = calcularNumero.soma(cartasEscolhidas);
23
24          assertEquals( expected: 63, numeroCalculado);
25      }
26  }
```

Run: CalculadorNumeroTest x

Tests failed: 1, passed: 1 of 2 tests – 31 ms

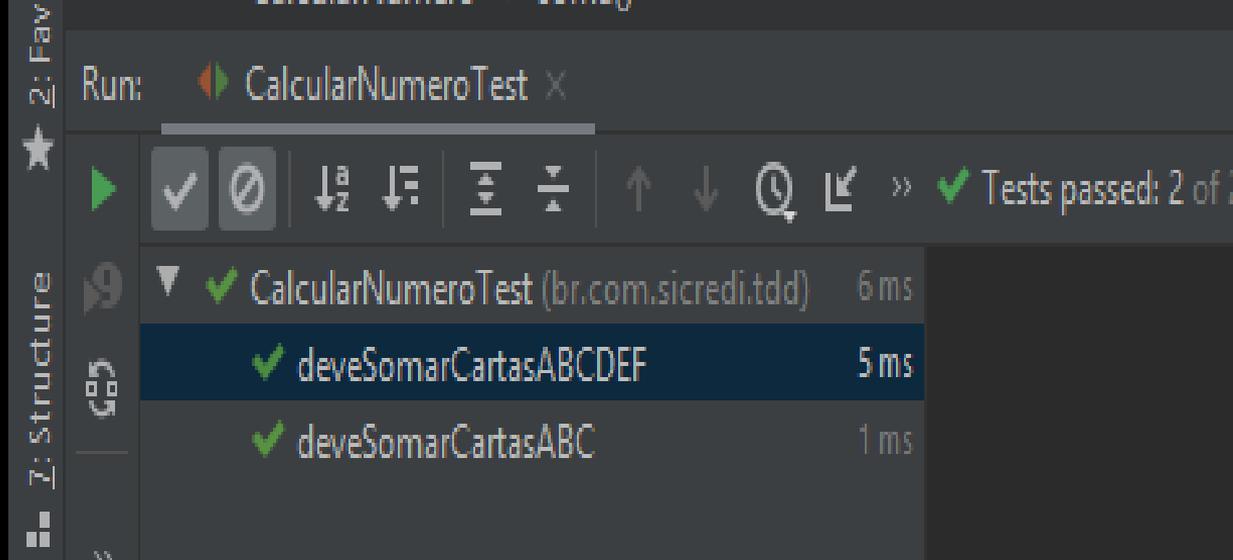
Test Name	Duration	Status
CalculadorNumeroTest (br.com.sicredi.tdd)	31 ms	Failed
deveSomarCartasABCDEF	31 ms	Failed
deveSomarCartasABC	0 ms	Passed

java.lang.AssertionError:  
Expected :63  
Actual :7  
[<Click to see difference>](#)

Criando implementação  
para demais cartas

```
CalculadorNumeroTest.java x CalculadorNumero.java x  
public class CalculadorNumero {  
    public int soma(String cartas){  
        int numeroCalculado = 0;  
        if(cartas.contains("A")){  
            numeroCalculado = numeroCalculado + 1;  
        }  
        if(cartas.contains("B")){  
            numeroCalculado = numeroCalculado + 2;  
        }  
        if(cartas.contains("C")){  
            numeroCalculado = numeroCalculado + 4;  
        }  
        if(cartas.contains("D")){  
            numeroCalculado = numeroCalculado + 8;  
        }  
        if(cartas.contains("E")){  
            numeroCalculado = numeroCalculado + 16;  
        }  
        if(cartas.contains("F")){  
            numeroCalculado = numeroCalculado + 32;  
        }  
        return numeroCalculado;  
    }  
}
```

Teste passando



The screenshot shows the Run console of an IDE. At the top, it says "Run: CalculadorNumeroTest". Below that is a toolbar with various icons for test execution and navigation. The main area displays a list of test results:

- CalculadorNumeroTest (br.com.sicredi.tdd) 6 ms
- deveSomarCartasABCDEF 5 ms
- deveSomarCartasABC 1 ms

All tests are marked with a green checkmark, indicating they passed. The text "Tests passed: 2 of" is visible at the top right of the console.

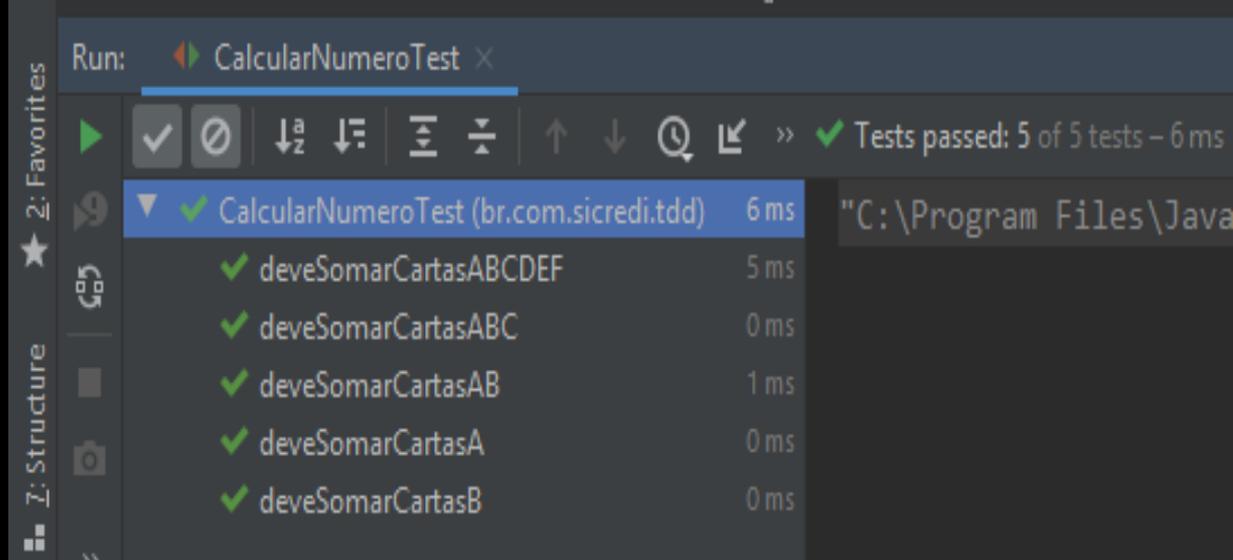
## Criando mais testes

```
CalcularNumeroTest.java x CalcularNumero.java x
@Test
public void deveSomarCartasA() {
    String cartasEscolhidas = "A";
    CalcularNumero calcularNumero = new CalcularNumero();
    int numeroCalculado = calcularNumero.soma(cartasEscolhidas);
    assertEquals( expected: 1, numeroCalculado);
}

@Test
public void deveSomarCartasB() {
    String cartasEscolhidas = "B";
    CalcularNumero calcularNumero = new CalcularNumero();
    int numeroCalculado = calcularNumero.soma(cartasEscolhidas);
    assertEquals( expected: 2, numeroCalculado);
}

@Test
public void deveSomarCartasAB() {
    String cartasEscolhidas = "AB";
    CalcularNumero calcularNumero = new CalcularNumero();
    int numeroCalculado = calcularNumero.soma(cartasEscolhidas);
    assertEquals( expected: 3, numeroCalculado);
}
```

Teste passando



The screenshot shows the Run console of an IDE. At the top, it says "Run: CalculadorNumeroTest". Below that, there are several icons for test actions. The main area shows a tree view of test results:

- CalculadorNumeroTest (br.com.sicredi.tdd) 6 ms
- deveSomarCartasABCDEF 5 ms
- deveSomarCartasABC 0 ms
- deveSomarCartasAB 1 ms
- deveSomarCartasA 0 ms
- deveSomarCartasB 0 ms

At the top right of the console, it says "Tests passed: 5 of 5 tests - 6 ms". On the right side, there is a text box containing the path "C:\Program Files\Java".

# Os testes falam?

Notem que temos uma análise combinatória entre as cartas escolhidas

Será que teremos que escrever 63 testes?

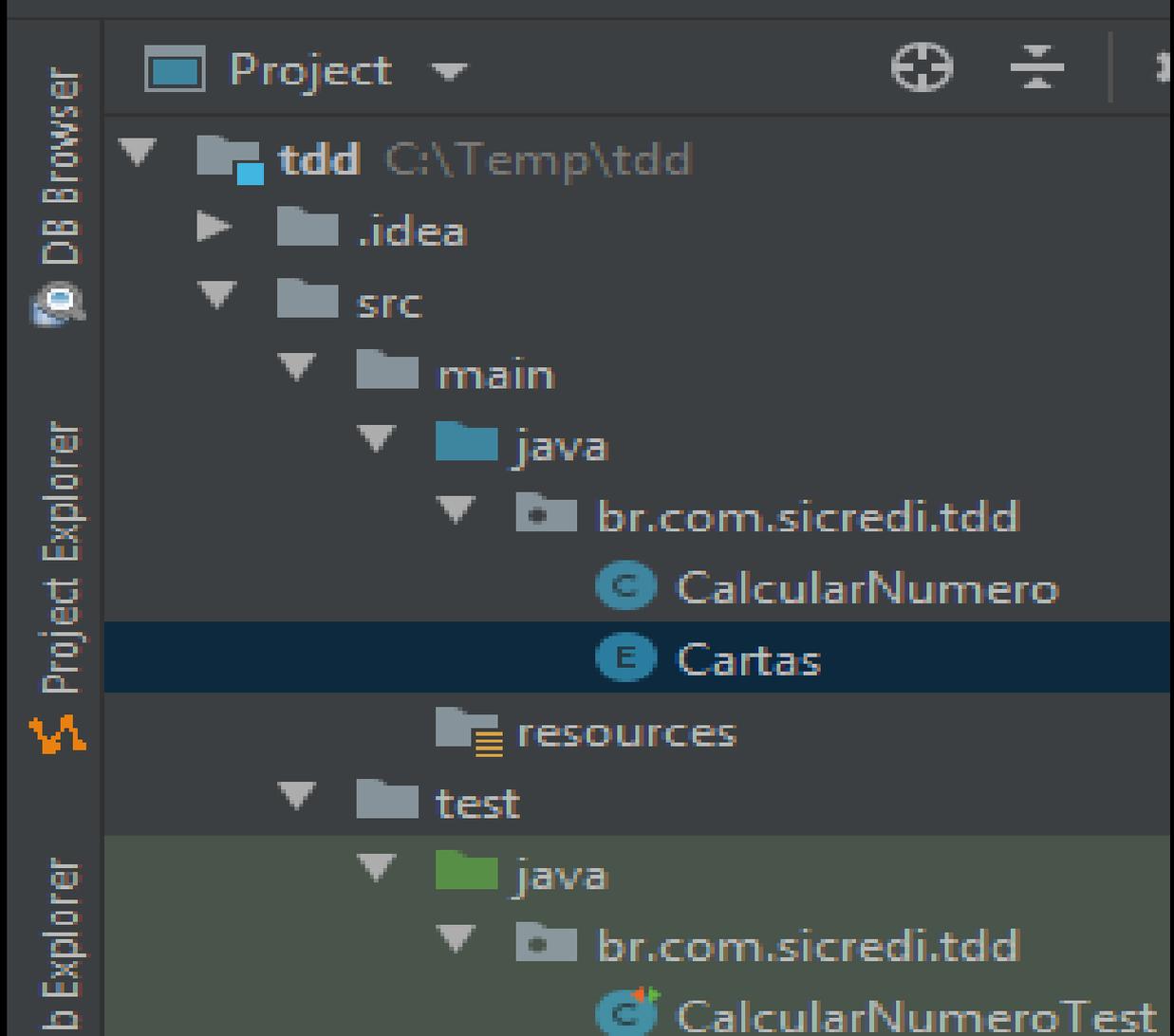
Vamos analisar o código e “refatorar”?

Código está implementando a regra de negócio

Código está complexo, "IFZEIRA"

```
public class CalcularNumero {  
  
    public int soma(String cartas){  
        int numeroCalculado = 0;  
        if(cartas.contains("A")){  
            numeroCalculado = numeroCalculado + 1;  
        }  
        if(cartas.contains("B")){  
            numeroCalculado = numeroCalculado + 2;  
        }  
        if(cartas.contains("C")){  
            numeroCalculado = numeroCalculado + 4;  
        }  
        if(cartas.contains("D")){  
            numeroCalculado = numeroCalculado + 8;  
        }  
        if(cartas.contains("E")){  
            numeroCalculado = numeroCalculado + 16;  
        }  
        if(cartas.contains("F")){  
            numeroCalculado = numeroCalculado + 32;  
        }  
        return numeroCalculado;  
    }  
}
```

Vamos refatorar,  
podemos criar um enum  
para as Cartas



## Enum Cartas com a responsabilidade de saber os valores das Cartas

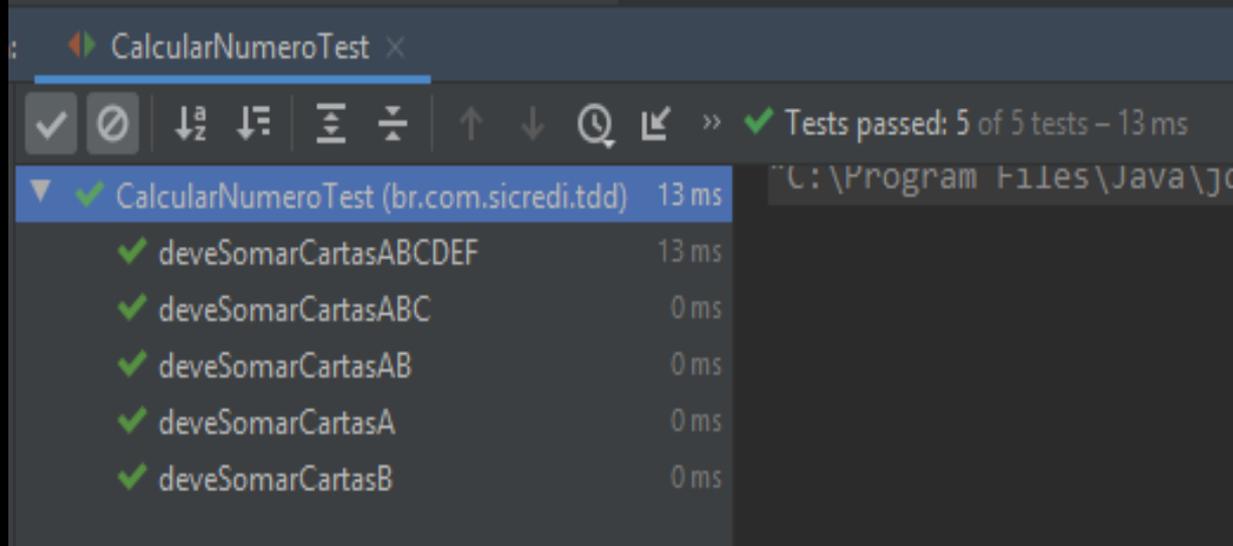
```
public enum Cartas {  
  
    CARTA_A( carta: "A", valor: 1),  
    CARTA_B( carta: "B", valor: 2),  
    CARTA_C( carta: "C", valor: 4),  
    CARTA_D( carta: "D", valor: 8),  
    CARTA_E( carta: "E", valor: 16),  
    CARTA_F( carta: "F", valor: 32);  
  
    private final String carta;  
    private final int valor;  
  
    Cartas(String carta, int valor) { this.carta = carta;  
        this.valor = valor;  
    }  
  
    public int getValor() { return valor; }
```



Retirando a  
responsabilidade da  
soma saber os valores  
das Cartas

```
public int soma(String cartas){  
    int numeroCalculado = 0;  
    if(cartas.contains("A")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_A.getValor();  
    }  
    if(cartas.contains("B")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_B.getValor();  
    }  
    if(cartas.contains("C")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_C.getValor();  
    }  
    if(cartas.contains("D")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_D.getValor();  
    }  
    if(cartas.contains("E")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_E.getValor();  
    }  
    if(cartas.contains("F")){  
        numeroCalculado = numeroCalculado + Cartas.CARTA_F.getValor();  
    }  
    return numeroCalculado;  
}
```

Testes passando



Refatorar com segurança  
de não quebrar nada

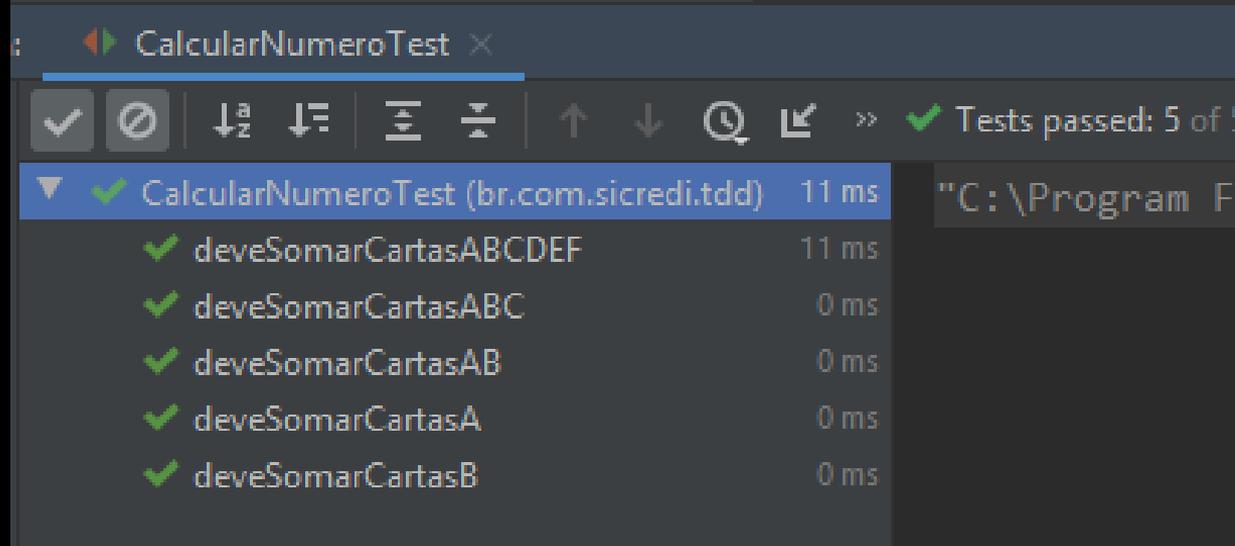
Simplificar a soma para  
retirar a “IFZEIRA”

```
CalculadorNumeroTest.java × CalculadorNumero.java × Cartas.java ×  
package br.com.sicredi.tdd;  
  
import java.util.stream.Stream;  
  
public class CalculadorNumero {  
    public int soma(Cartas... cartas){  
        return Stream.of(cartas).mapToInt(Cartas::getValor).sum();  
    }  
}
```

Ajustando os testes,  
passando para soma as  
Cartas ao invés de uma  
string

```
public class CalcularNumeroTest {  
  
    @Test  
    public void deveSomarCartasABC() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_A, CARTA_B, CARTA_C);  
        assertEquals( expected: 7, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartasABCDEF() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_A, CARTA_B, CARTA_C, CARTA_D, CARTA_E, CARTA_F);  
        assertEquals( expected: 63, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartasA() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_A);  
        assertEquals( expected: 1, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartasB() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_B);  
    }  
}
```

Testes passando



The screenshot shows the Test Explorer window in Visual Studio. The title bar reads "CalculadorNumeroTest". The toolbar includes icons for passing, failing, and running tests, along with navigation and search icons. The test results are as follows:

Test Name	Duration
CalculadorNumeroTest (br.com.sicredi.tdd)	11 ms
deveSomarCartasABCDEF	11 ms
deveSomarCartasABC	0 ms
deveSomarCartasAB	0 ms
deveSomarCartasA	0 ms
deveSomarCartasB	0 ms

On the right side of the window, a snippet of the file path is visible: "C:\Program F".

# Os testes falam?

Será que teremos que escrever 63 testes?

Não, podemos criar somente alguns testes que nos de segurança no código.

Podemos criar um teste que:

- some todas as cartas

- some uma carta

- some 2 cartas

- um teste se não informar uma carta



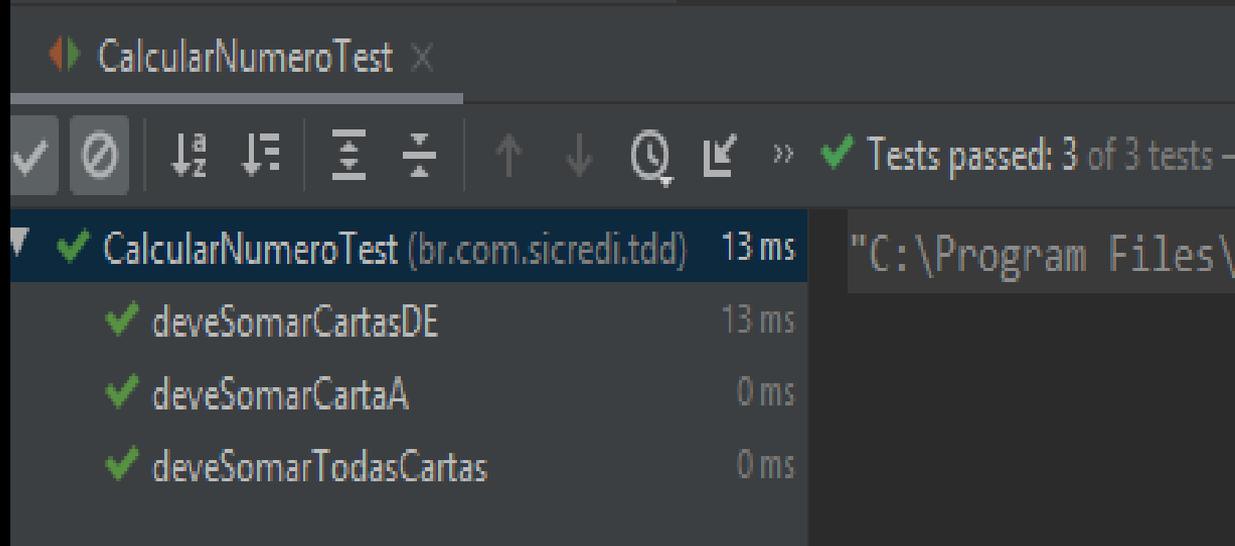
Criando teste que some todas as cartas

Criando teste que some 1 carta

Criando teste que some 2 cartas

```
public class CalcularNumeroTest {  
  
    @Test  
    public void deveSomarTodasCartas() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(Cartas.values());  
        assertEquals( expected: 63, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartaA() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_A);  
        assertEquals( expected: 1, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartasDE() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_D,CARTA_E);  
        assertEquals( expected: 24, numeroCalculado);  
    }  
  
}
```

Testes passando



The screenshot shows a test runner window titled 'CalculadorNumeroTest'. The interface includes a toolbar with various icons for test execution and navigation. The test results are displayed in a list format, showing that all three tests passed successfully. The first test, 'CalculadorNumeroTest (br.com.sicredi.tdd)', took 13 ms. The other two tests, 'deveSomarCartasDE' and 'deveSomarCartaA', both took 0 ms. The overall status is 'Tests passed: 3 of 3 tests'.

Test Name	Duration
CalculadorNumeroTest (br.com.sicredi.tdd)	13 ms
deveSomarCartasDE	13 ms
deveSomarCartaA	0 ms
deveSomarTodasCartas	0 ms

Tests passed: 3 of 3 tests -

Criando teste sem  
informar carta

Teste falha

```
}  
@Test(expected = IllegalArgumentException.class)  
public void deveTratarSomaSemCarta() {  
    CalcularNumero calcularNumero = new CalcularNumero();  
    int numeroCalculado = calcularNumero.soma( ...cartas: null);  
}  
}
```

CalcularNumeroTest > deveTratarSomaSemCarta()

CalculaNumeroTest x

Tests failed: 1, passed: 3 of 4 tests - 37 ms

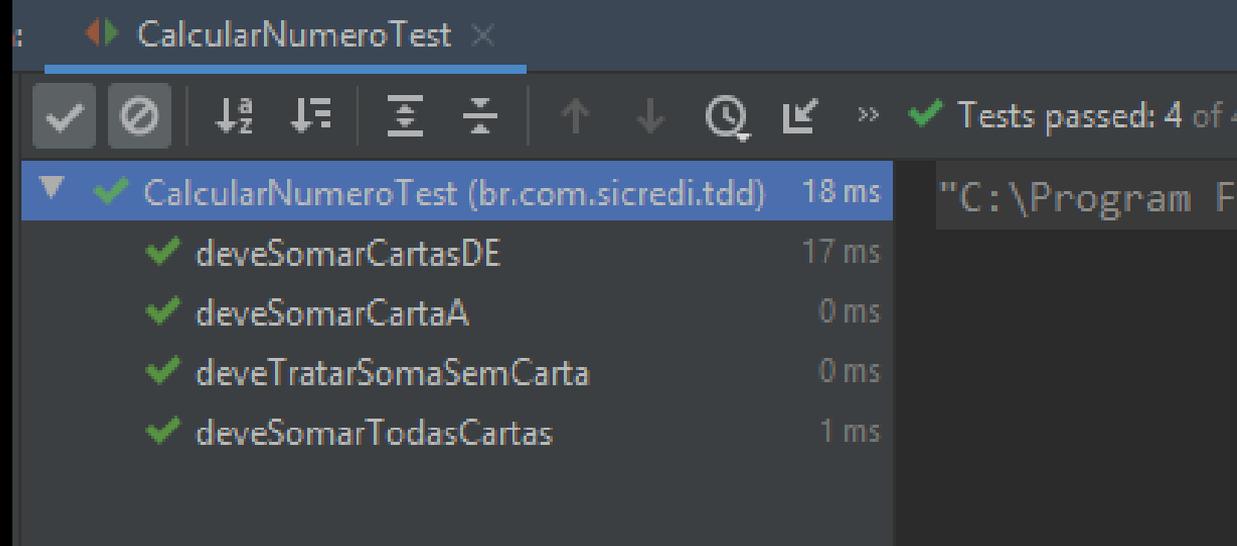
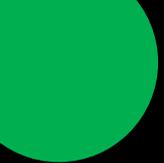
CalculaNumeroTest (br.com.sicredi.tdd)	37 ms
deveSomarCartasDE	10 ms
deveSomarCartaA	0 ms
deveTratarSomaSemCarta	27 ms
deveSomarTodasCartas	0 ms

java.lang.Exception: Unexpected excep  
<15 internal calls>



Implementando  
tratamento para soma  
sem carta

```
public class CalcularNumero {  
  
    public int soma(Cartas... cartas){  
        if(cartas == null || cartas.length <=0){  
            throw new IllegalArgumentException("Deve ter pelo menos uma carta");  
        }  
        return Stream.of(cartas).mapToInt(Cartas::getValor).sum();  
    }  
}
```



Testes passando

1. Adicione  
um teste

## User story

Como um jogador que gosta de jogos de desafio.

Quero jogar o jogo dos números mágicos com 7 cartas

Para se divertir

2. Execute o teste e observe o resultado

3. Escreva o código

4. Execute os testes automatizados

A			
1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31
33	35	37	39
41	43	45	47
49	51	53	55
57	59	61	63
65	67	69	71
73	75	77	79
81	83	85	87
89	91	93	95
97	99	101	103
105	107	109	111
113	115	117	119
121	123	125	127

B			
2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31
34	35	38	39
42	43	46	47
50	51	54	55
58	59	62	63
66	67	70	71
74	75	78	79
82	83	86	87
90	91	94	95
98	99	102	103
106	107	110	111
114	115	118	119
122	123	126	127

C			
4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31
36	37	38	39
44	45	46	47
52	53	54	55
60	61	62	63
68	69	70	71
76	77	78	79
84	85	86	87
92	93	94	95
100	101	102	103
108	109	110	111
116	117	118	119
124	125	126	127

D			
8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31
40	41	42	43
44	45	46	47
56	57	58	59
60	61	62	63
72	73	74	75
76	77	78	79
88	89	90	91
92	93	94	95
104	105	106	107
108	109	110	111
120	121	122	123
124	125	126	127

E			
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
112	113	114	115
116	117	118	119
120	121	122	123
124	125	126	127

F			
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
96	97	98	99
100	101	102	103
104	105	106	107
108	109	110	111
112	113	114	115
116	117	118	119
120	121	122	123
124	125	126	127

G			
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
100	101	102	103
104	105	106	107
108	109	110	111
112	113	114	115
116	117	118	119
120	121	122	123
124	125	126	127

Ajustando testes de soma de todas as cartas

Teste falha, pois não existe a nova carta

```
public class CalcularNumeroTest {  
  
    @Test  
    public void deveSomarTodasCartas() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(Cartas.values());  
        assertEquals( expected: 127, numeroCalculado);  
    }  
  
    @Test  
    public void deveSomarCartaA() {  
        CalcularNumero calcularNumero = new CalcularNumero();  
        int numeroCalculado = calcularNumero.soma(CARTA_A);  
        assertEquals( expected: 1, numeroCalculado);  
    }  
}
```

CalculadorNumeroTest > deveSomarTodasCartas()

CalculadorNumeroTest x

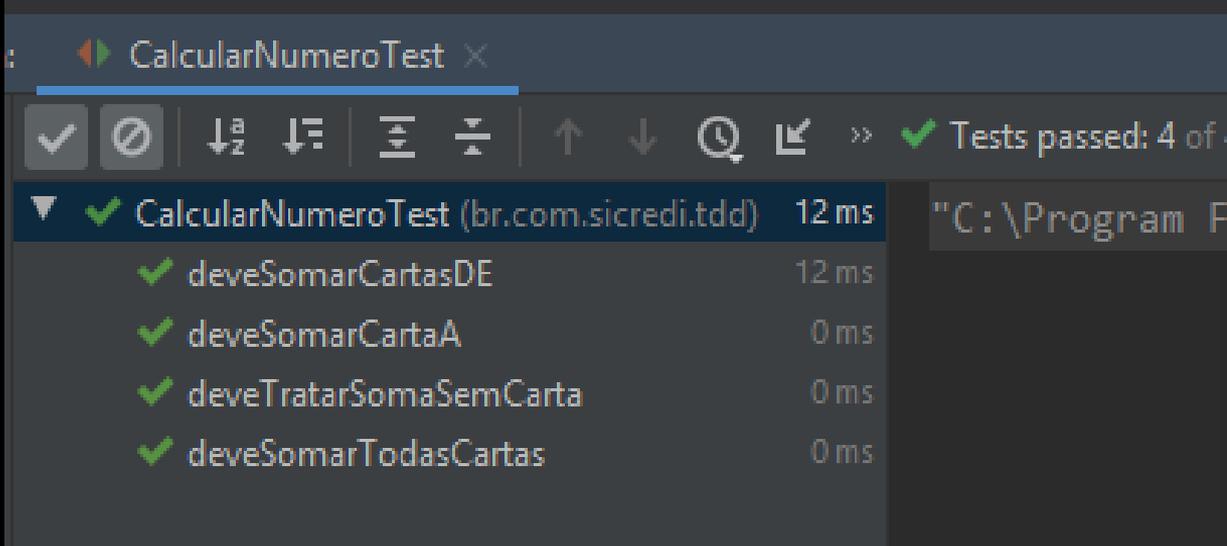
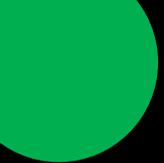
Tests failed: 1, passed: 3 of 4 tests – 30 ms

Test Name	Duration	Status
CalculadorNumeroTest (br.com.sicredi.tdd)	30 ms	Failed
deveSomarCartasDE	12 ms	Passed
deveSomarCartaA	0 ms	Passed
deveTratarSomaSemCarta	0 ms	Passed
deveSomarTodasCartas	18 ms	Failed

java.lang.AssertionError:  
Expected :127  
Actual :63  
<Click to see difference>

## Implementando nova carta

```
public enum Cartas {  
  
    CARTA_A( carta: "A", valor: 1),  
    CARTA_B( carta: "B", valor: 2),  
    CARTA_C( carta: "C", valor: 4),  
    CARTA_D( carta: "D", valor: 8),  
    CARTA_E( carta: "E", valor: 16),  
    CARTA_F( carta: "F", valor: 32),  
    CARTA_G( carta: "G", valor: 64);  
  
    private final String carta;  
    private final int valor;  
  
    Cartas(String carta, int valor) { this.carta = carta;  
        this.valor = valor;  
    }  
  
    public int getValor() { return valor; }  
}
```



Testes passando



Obrigado 😊

Trilha

*Extreme Programming*

*#TDCPOA19*