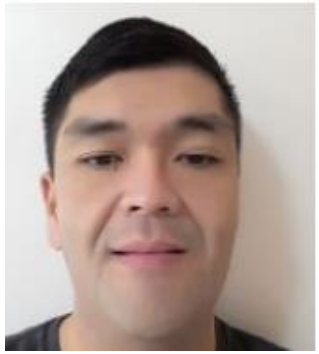

**Trilha**

***Extreme Programming***

***#TDCPOA19***



**LEANDRO AKIRA IWASAKI**

51 98194-8443

leandro-akira-iwasaki

**RODRIGO MURARI SEVERO**

51 99714-6600

rmsevero

@rodrigomsevero

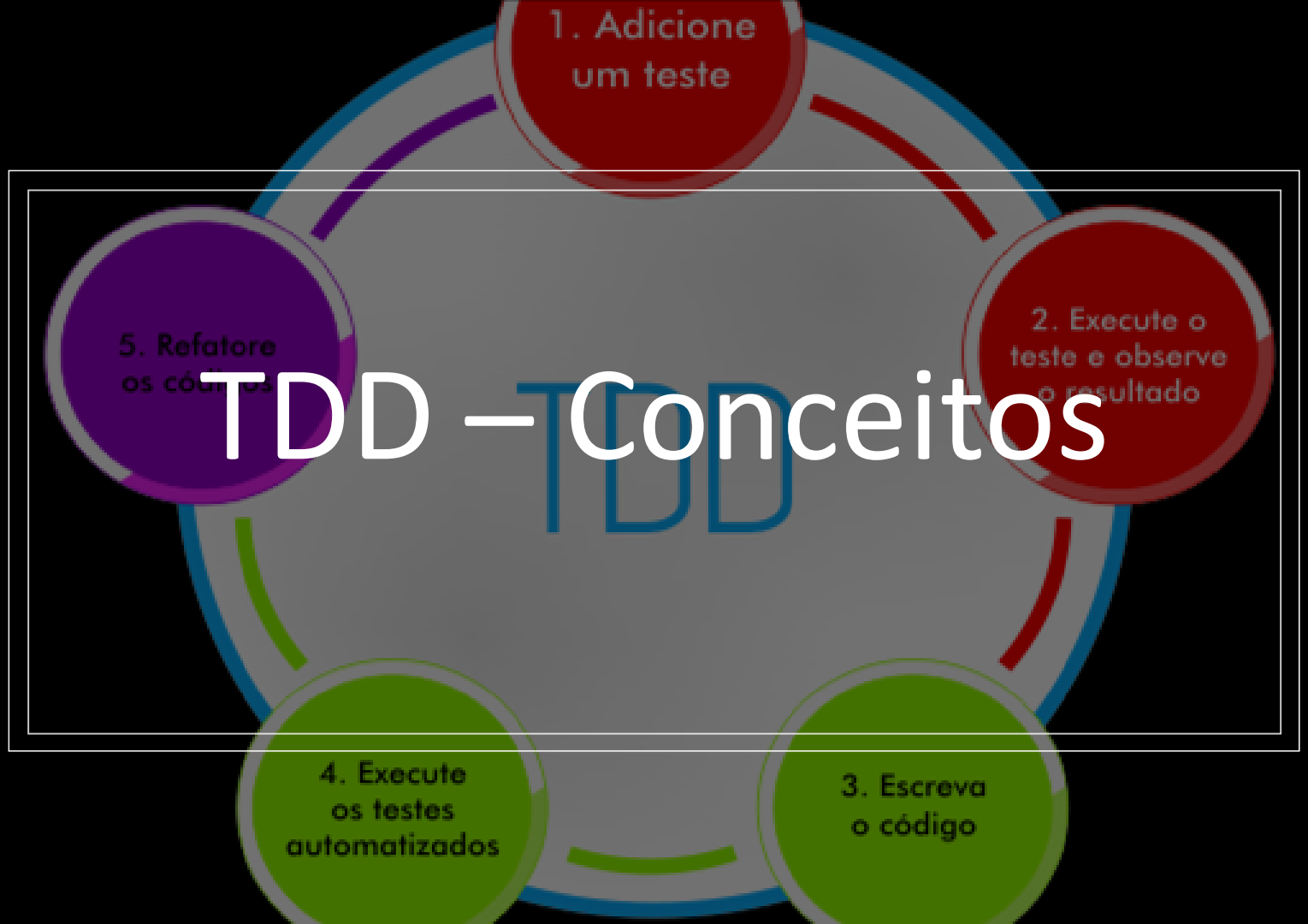
quaslati.wordpress.com

# Conceitos básicos TDD (*Test Driven Development*)

Que tal jogar o jogo dos números mágicos

Vamos ver na prática?

# TDD – Conceitos



# 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

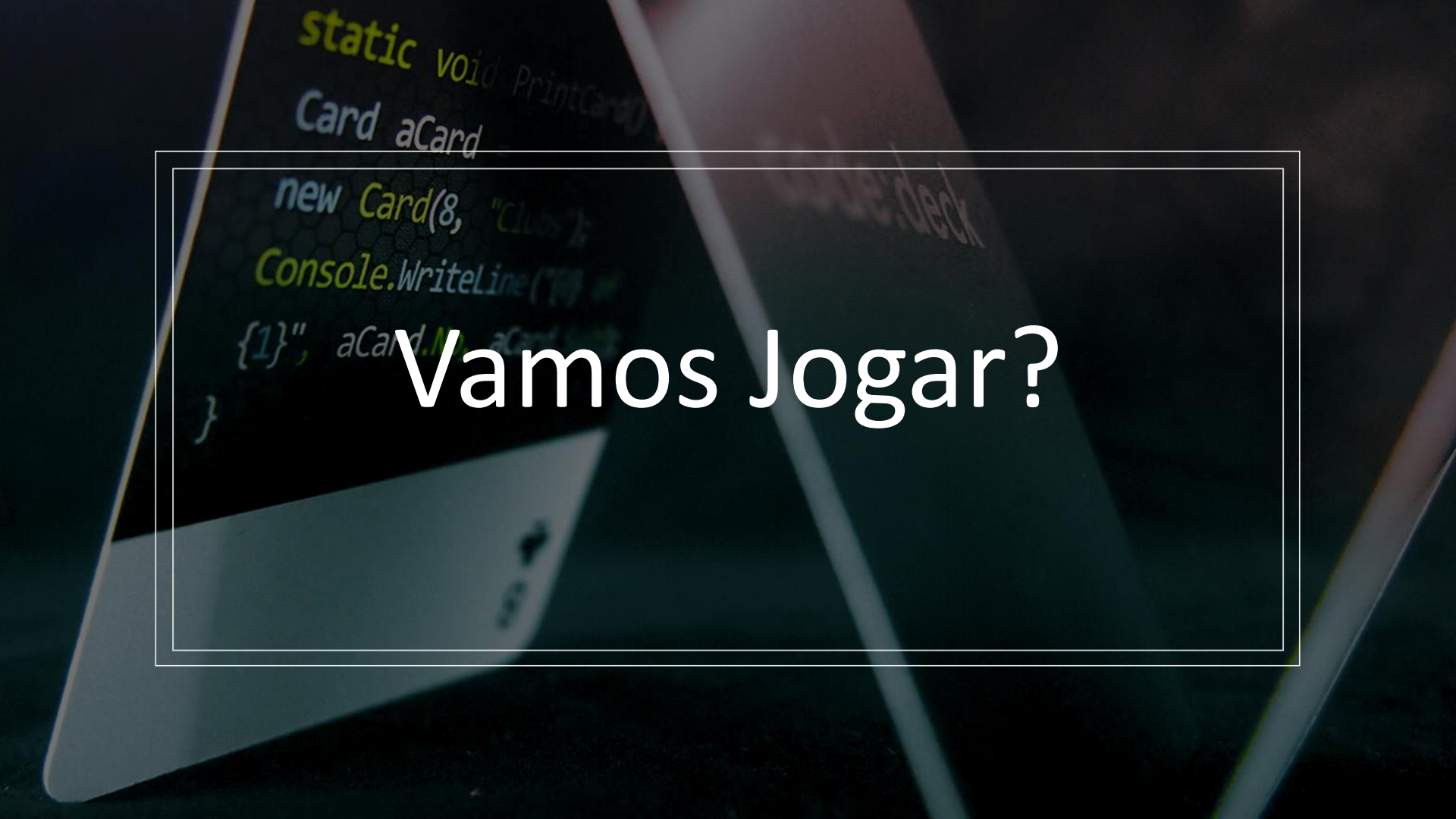
Código menos acoplado

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

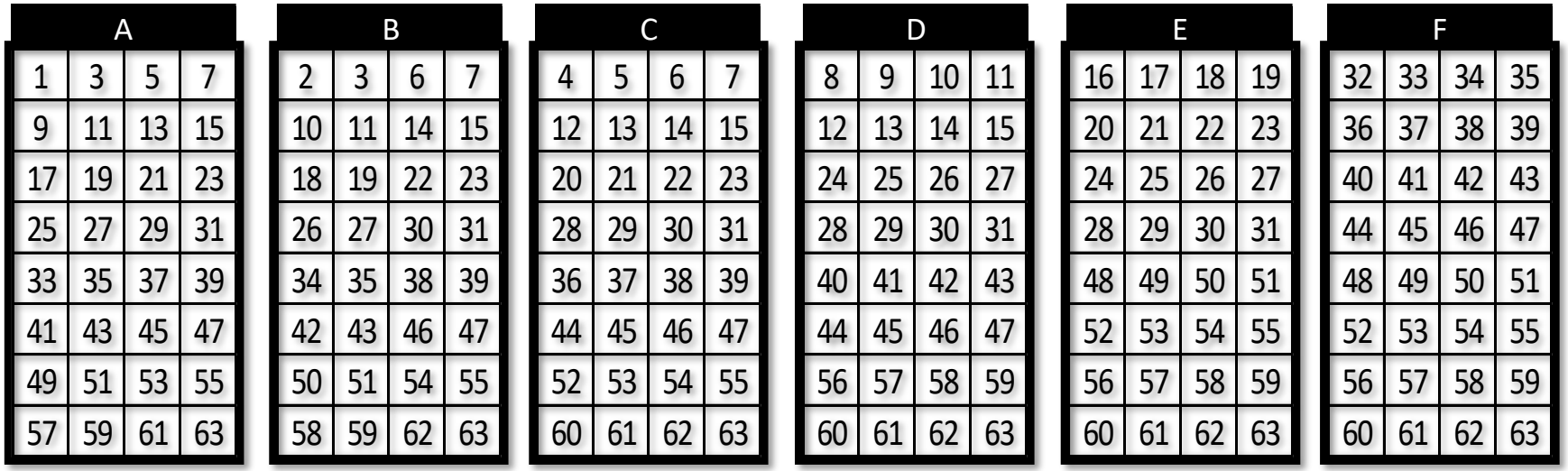
Segurança na correção de bugs

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


static void PrintCard
Card aCard
new Card(8,
Console.WriteLine
{1}",
}

# 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 |

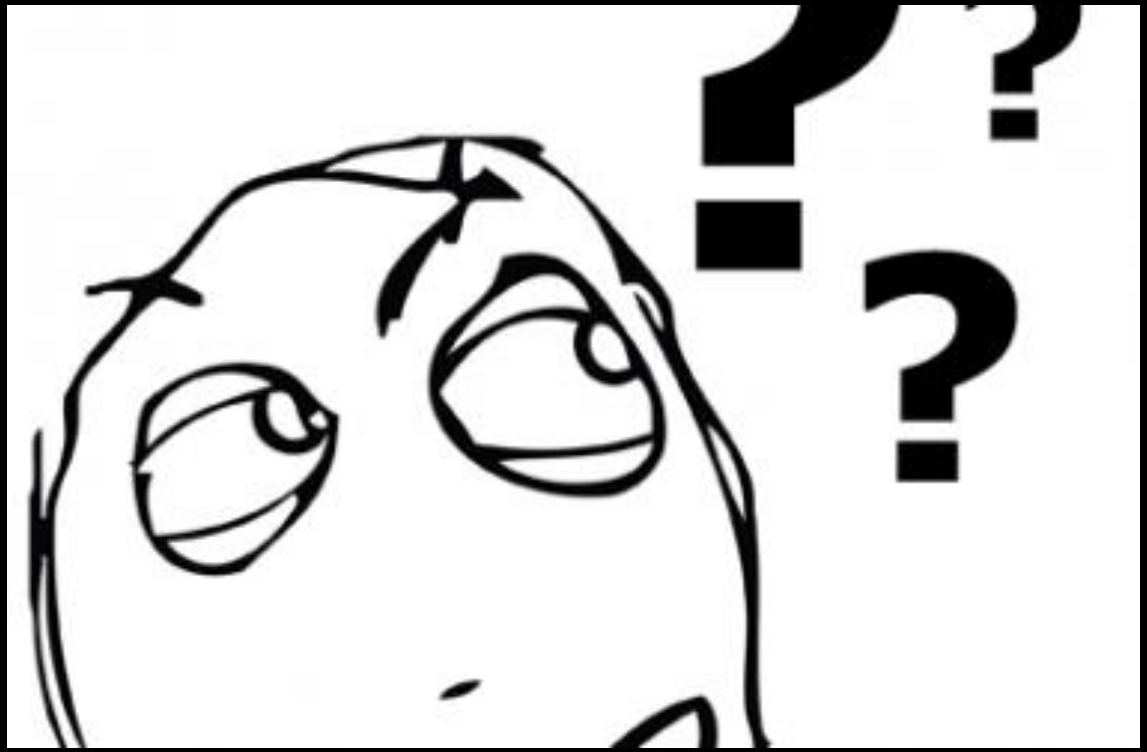
| D | | | |
|---|---|---|---|
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |
| 40 | 41 | 42 | 43 |
| 44 | 45 | 46 | 47 |
| 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 |

| 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 = 35

| 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 |

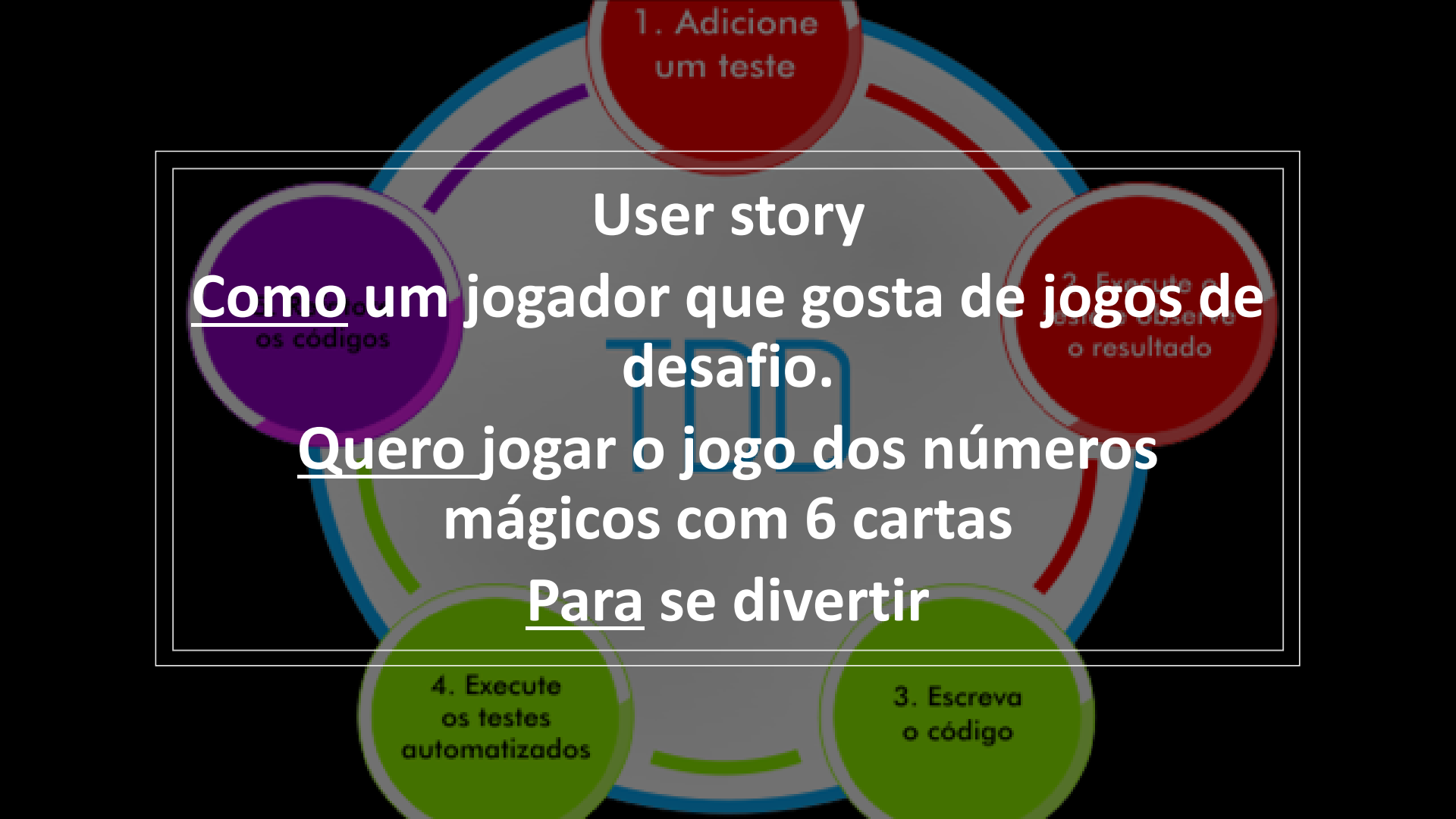
| B | | | |
|---|---|---|---|
| 2 | 3 | 6 | 7 |
| 10 | 11 | 14 | 15 |
| 18 | 19 | 22 | 23 |
| 26 | 27 | 30 | 31 |
| 34 | 35 | 38 | 39 |
| 42 | 43 | 46 | 47 |
| 50 | 51 | 54 | 55 |
| 58 | 59 | 62 | 63 |

| 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 |

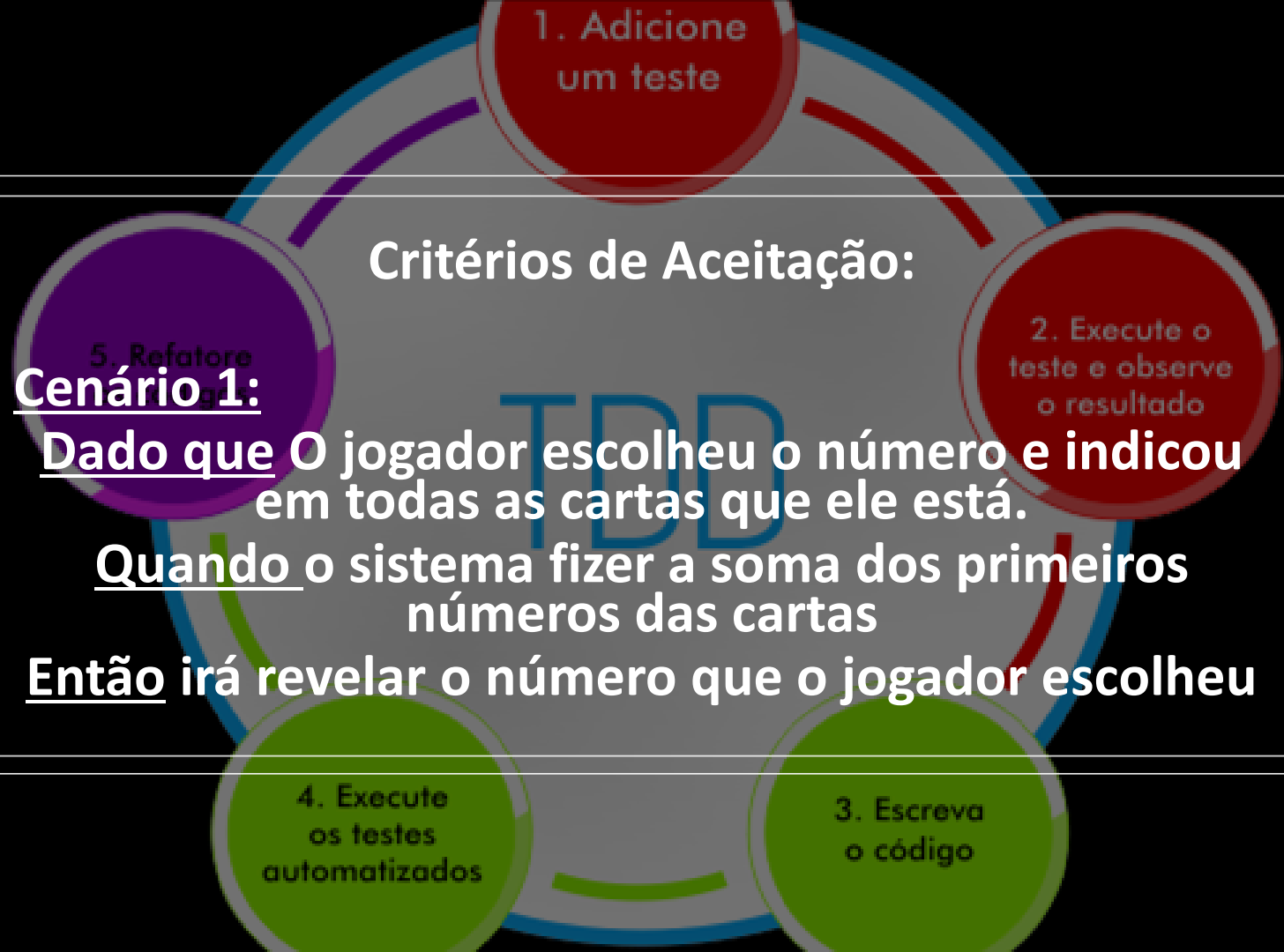



**User story**

**Como um jogador que gosta de jogos de desafio.**

**Quero jogar o jogo dos números mágicos com 6 cartas**

**Para se divertir**




## Critérios de Aceitação:

**Cenário 1:**

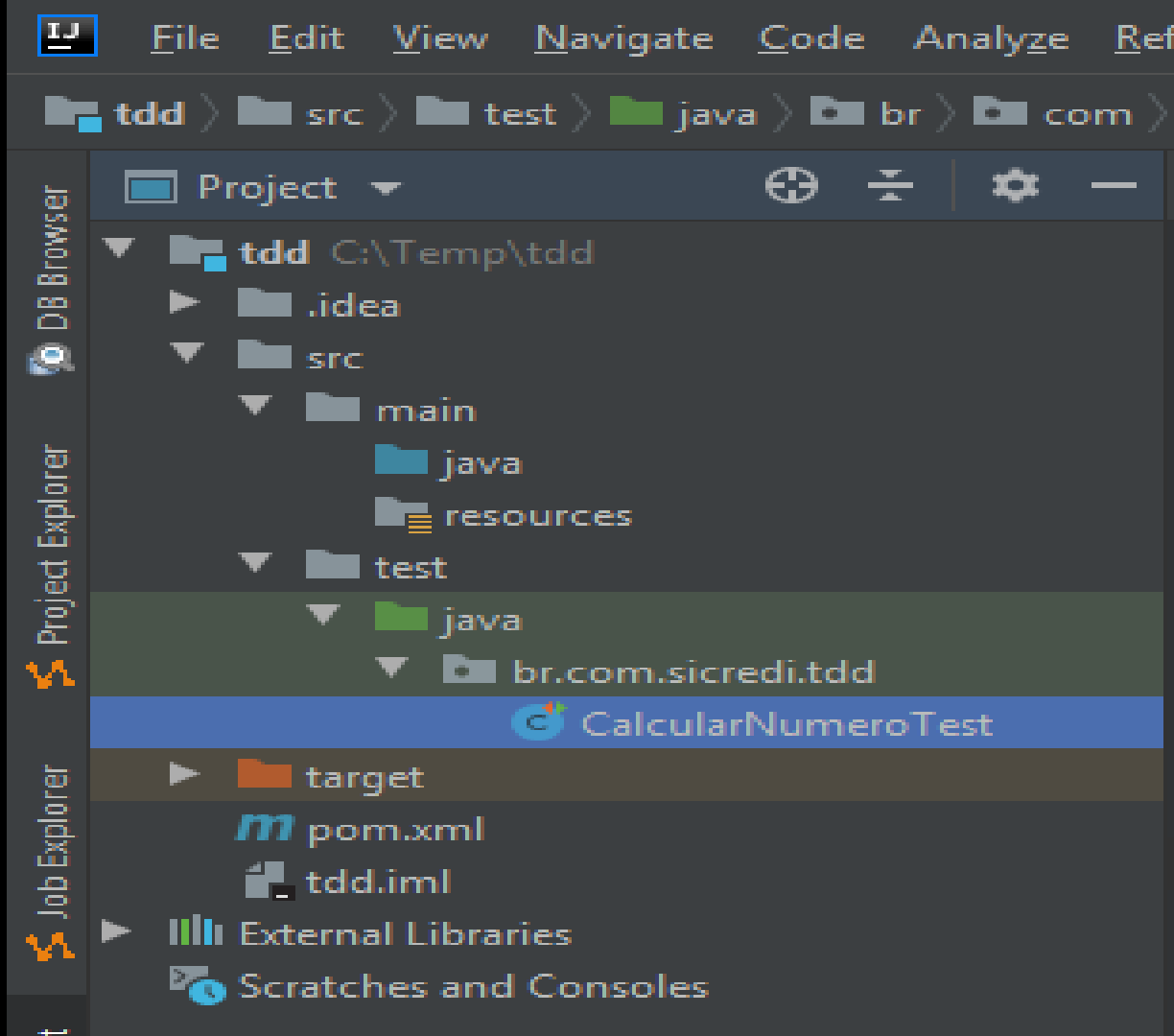
**Dado que** O jogador escolheu o número e indicou em todas as cartas que ele está.

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

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

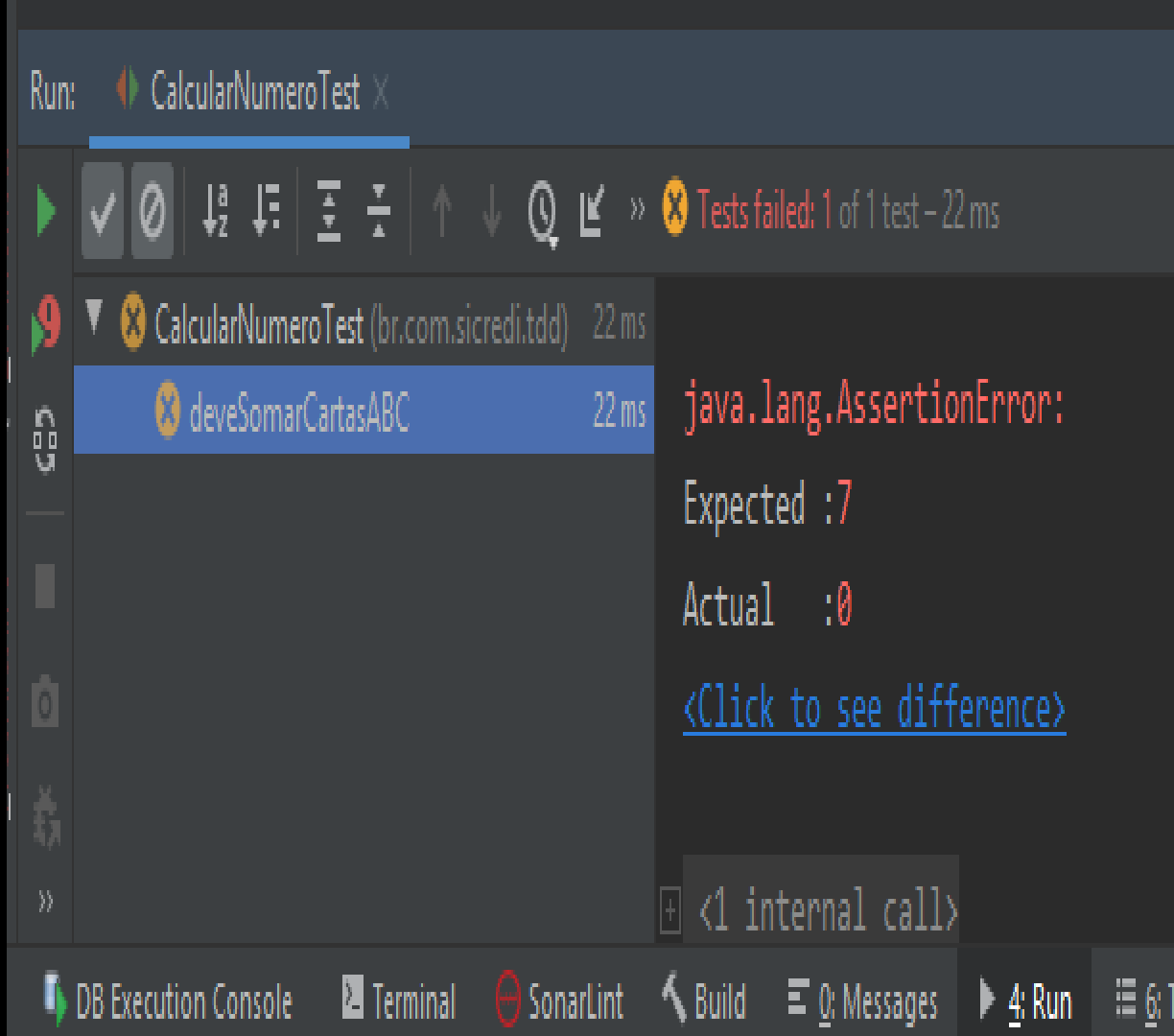
Criando o projeto

Iniciando pelos testes



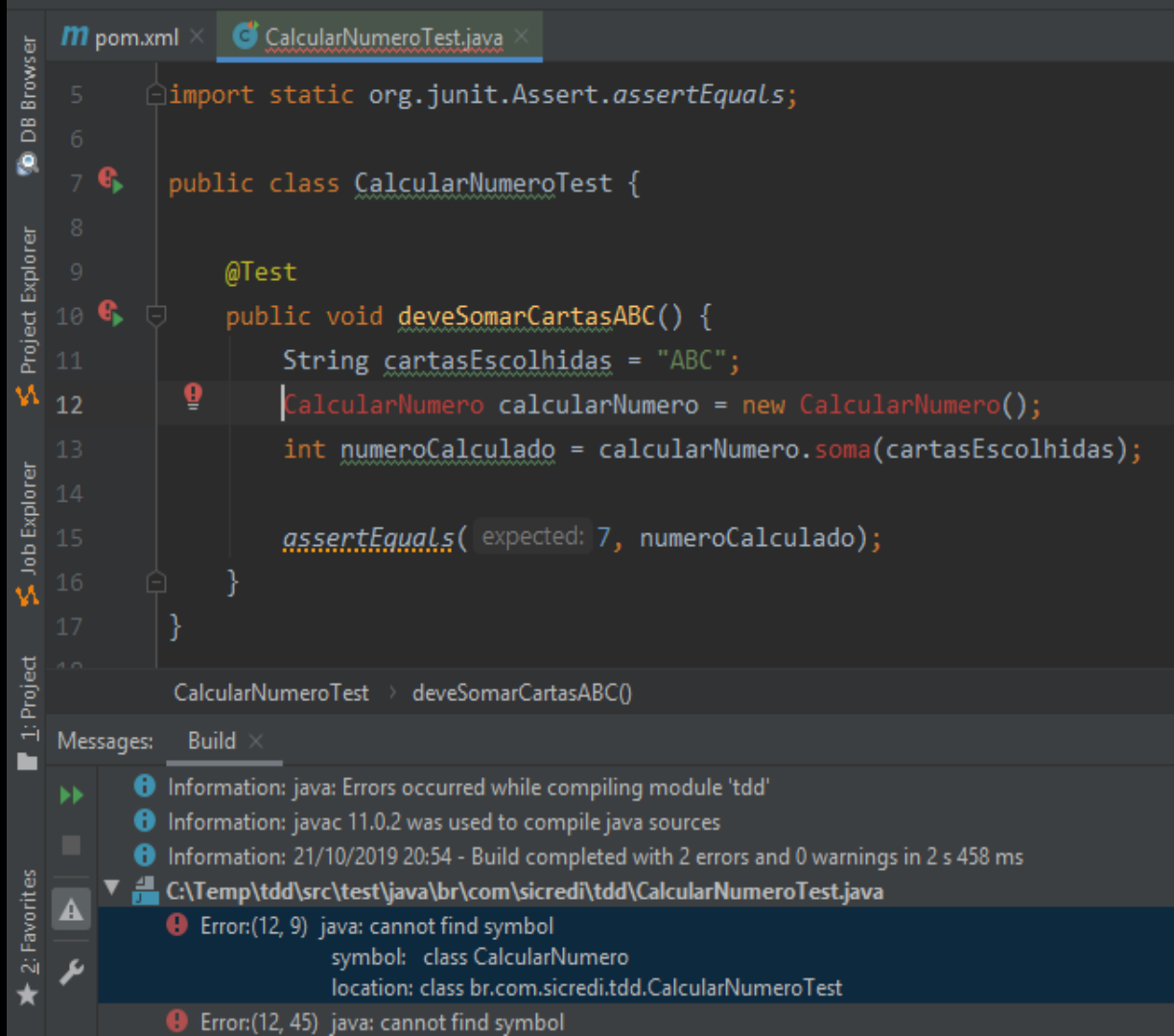
Criando o primeiro teste

Teste falhando

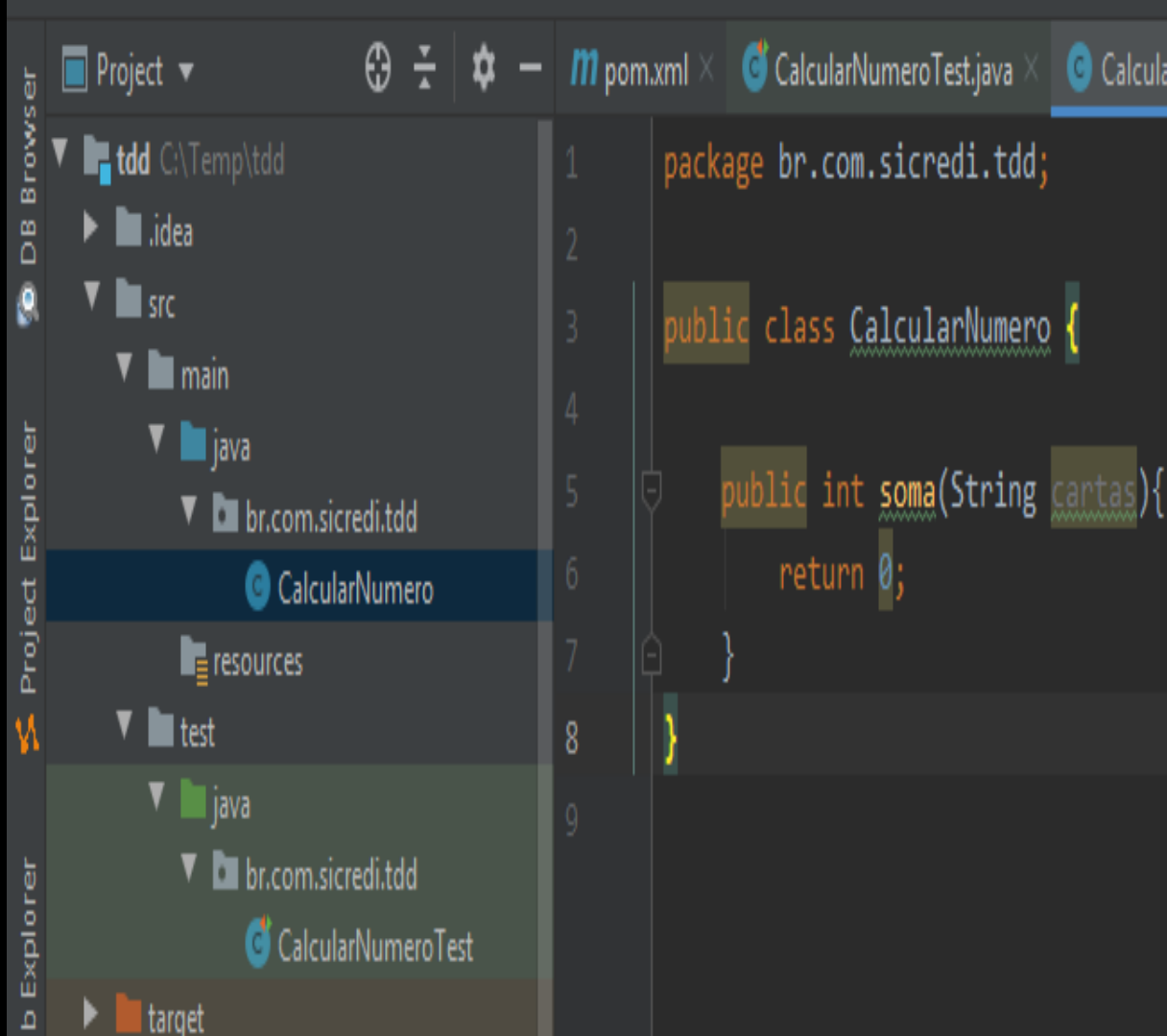


Inserindo os testes da implementação

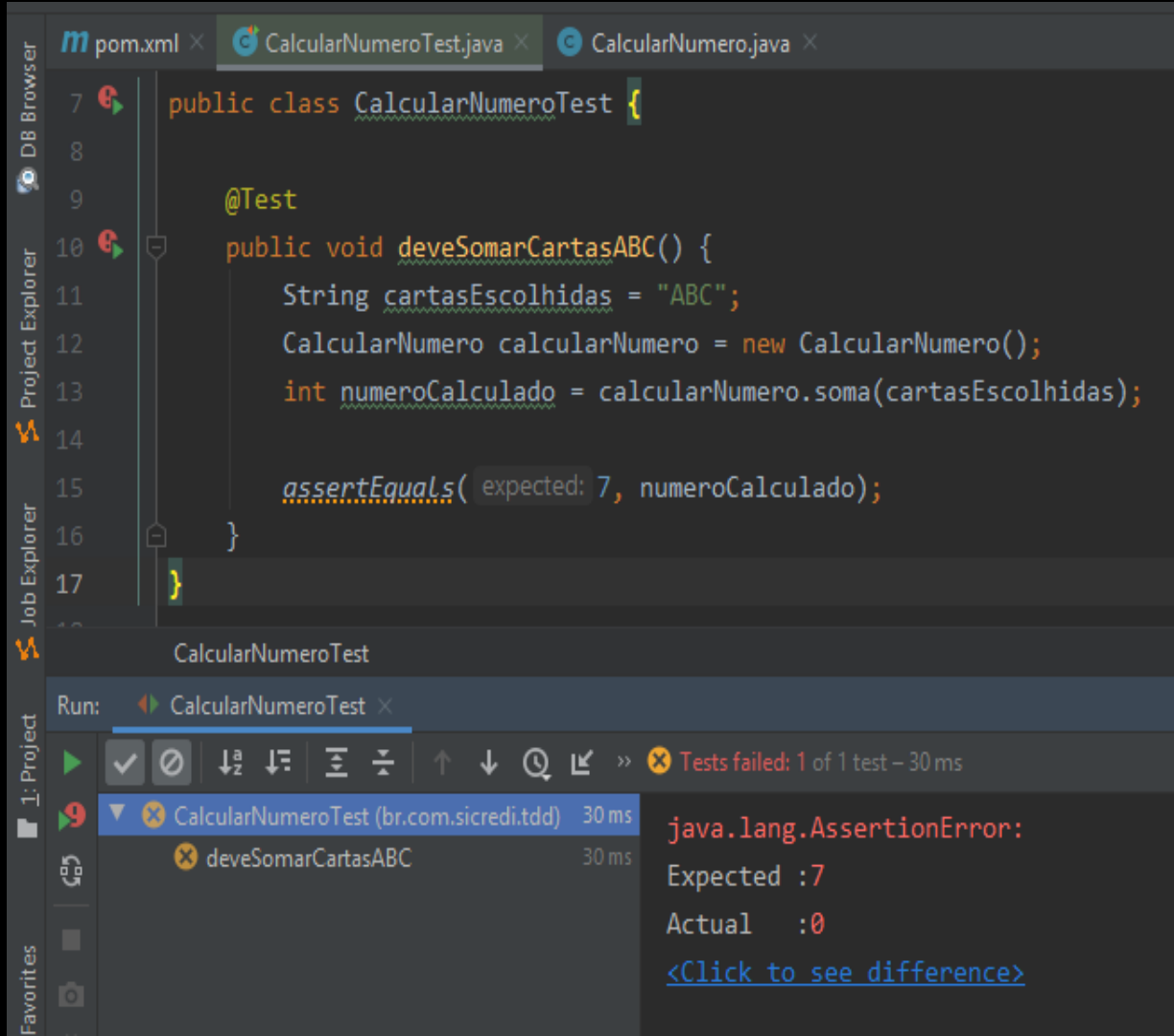
Código não compila



Criando a classe responsável pela implementação

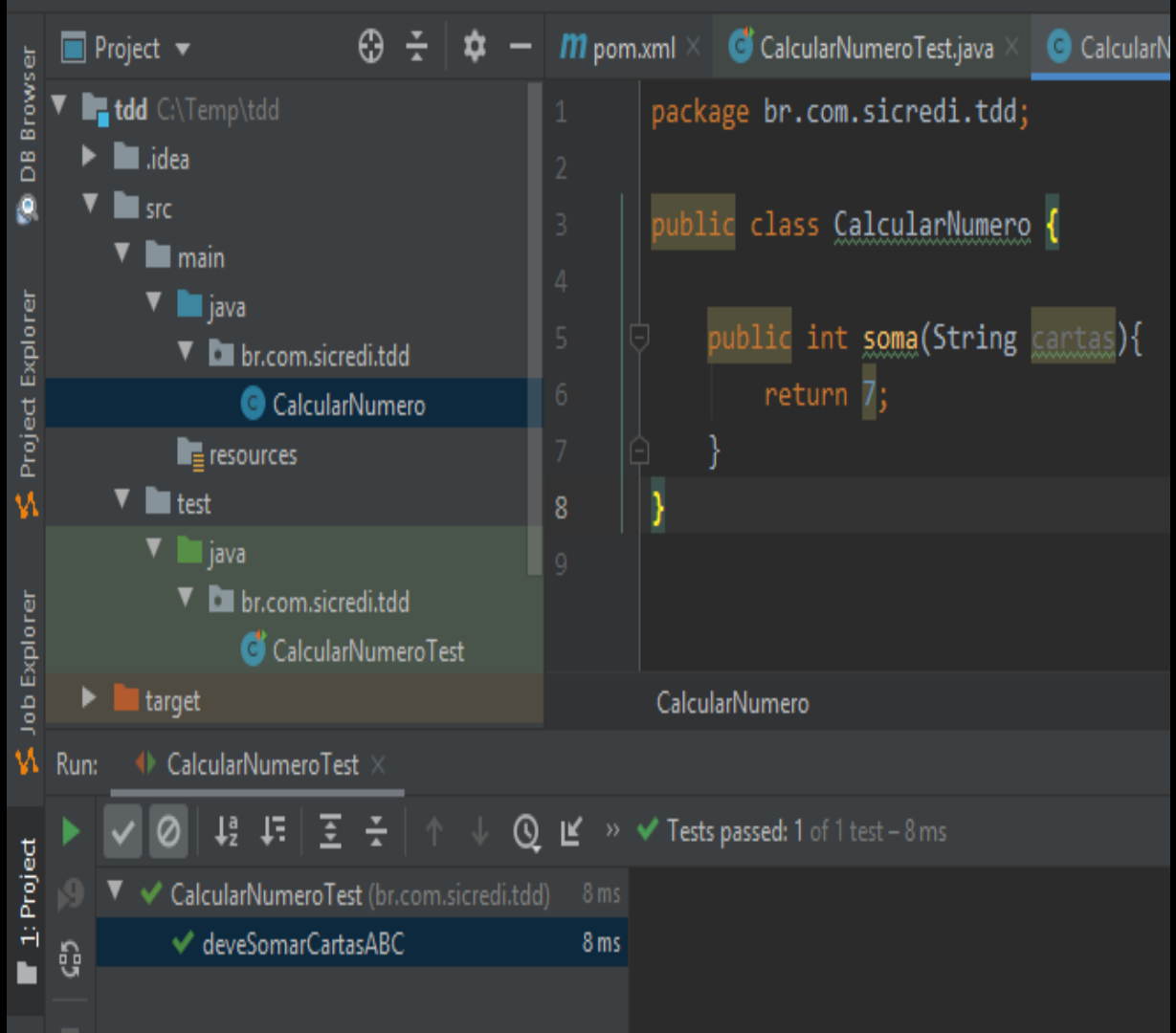


Código compila e teste falha

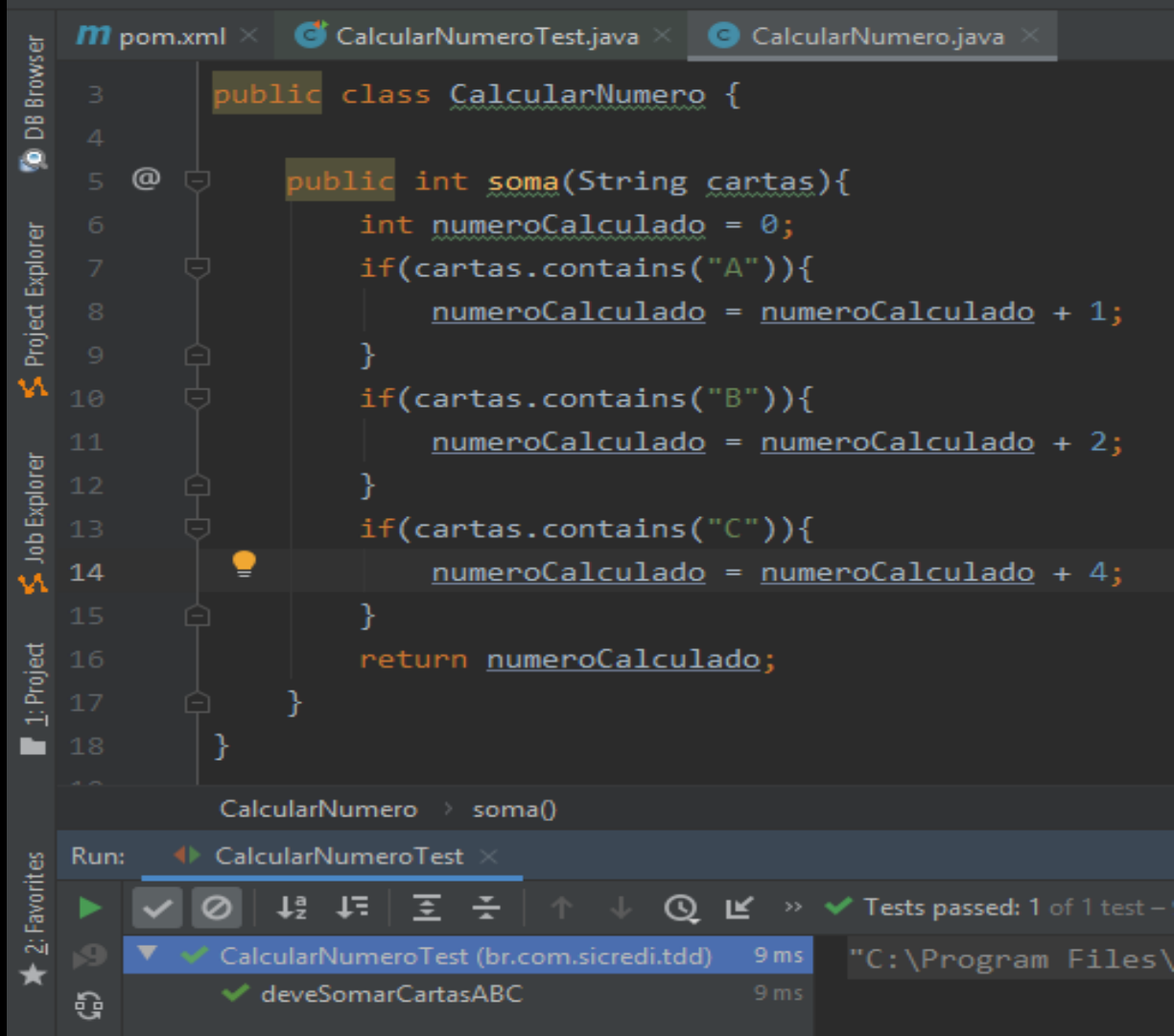


Fazendo o teste passar

Implementado o valor esperado pelo teste

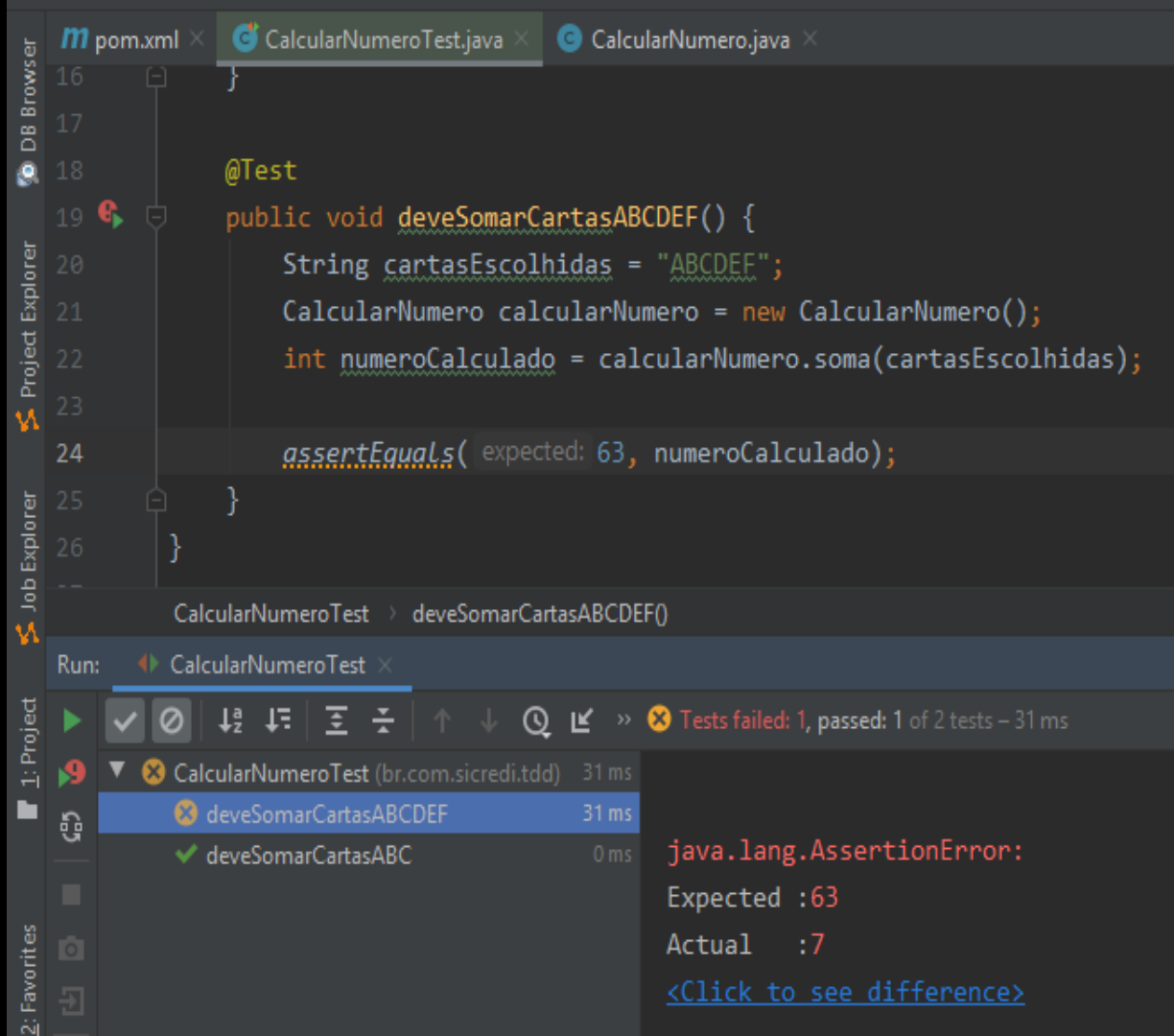


Implementando a regra de negócio

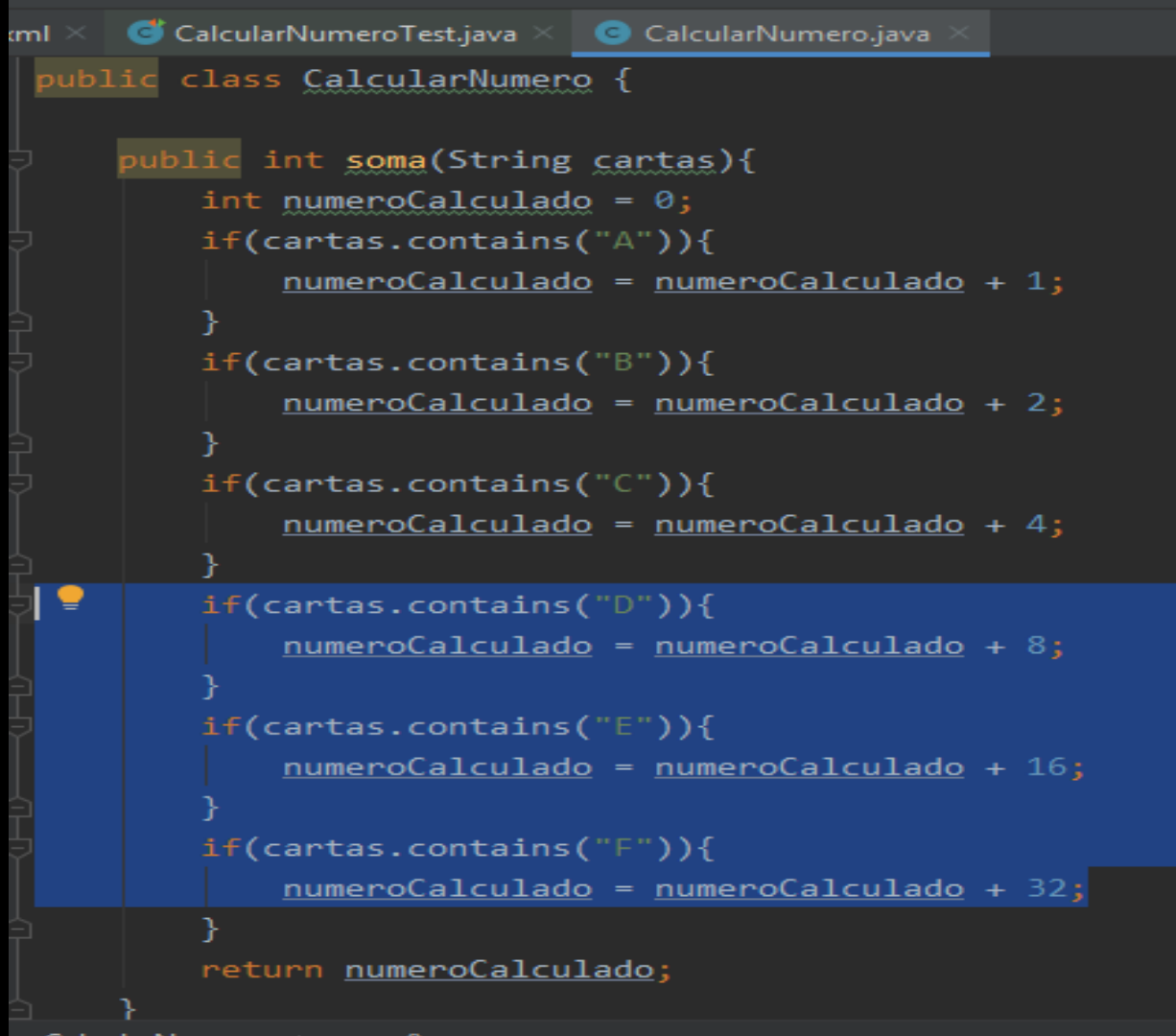


Criando testes para as demais cartas

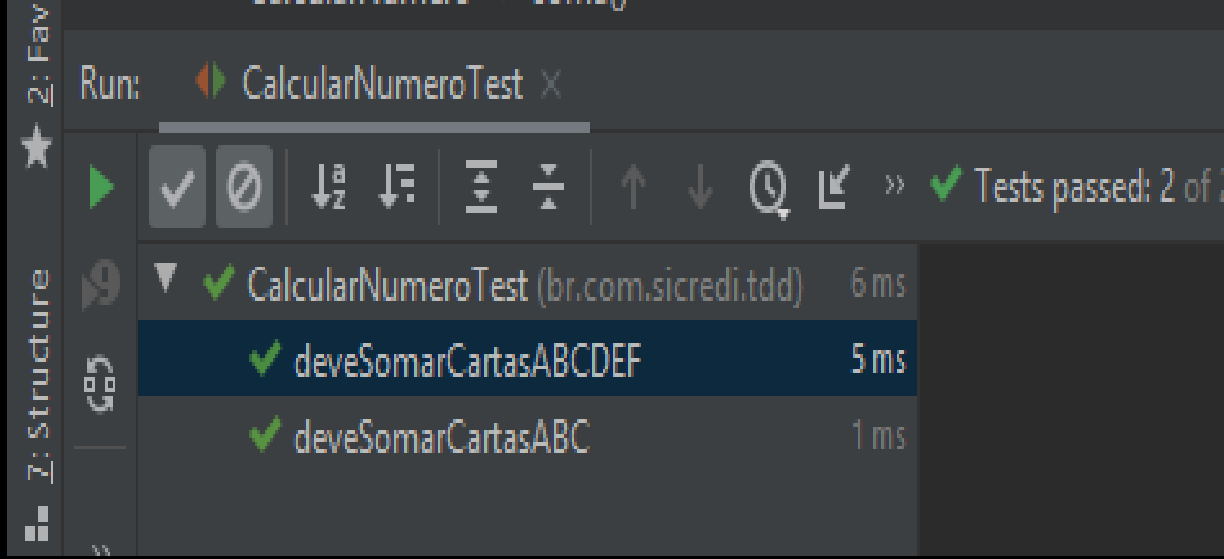
Teste falhando



Criando implementação para demais cartas



Teste passando

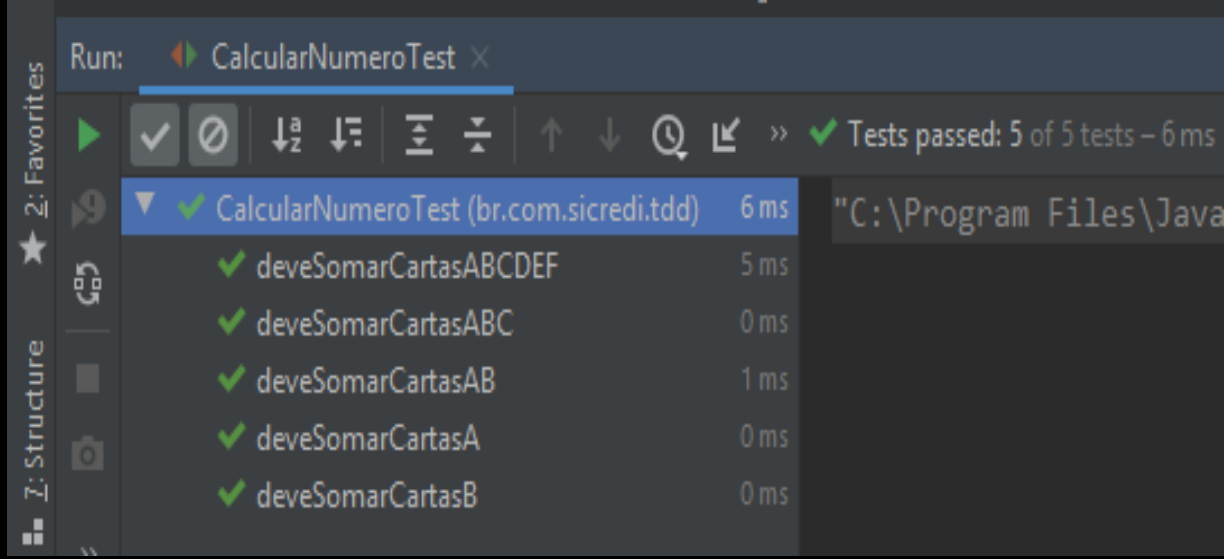


Criando mais testes

CalcularNumeroTest.java | CalcularNumero.java

```java
@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



# 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”?

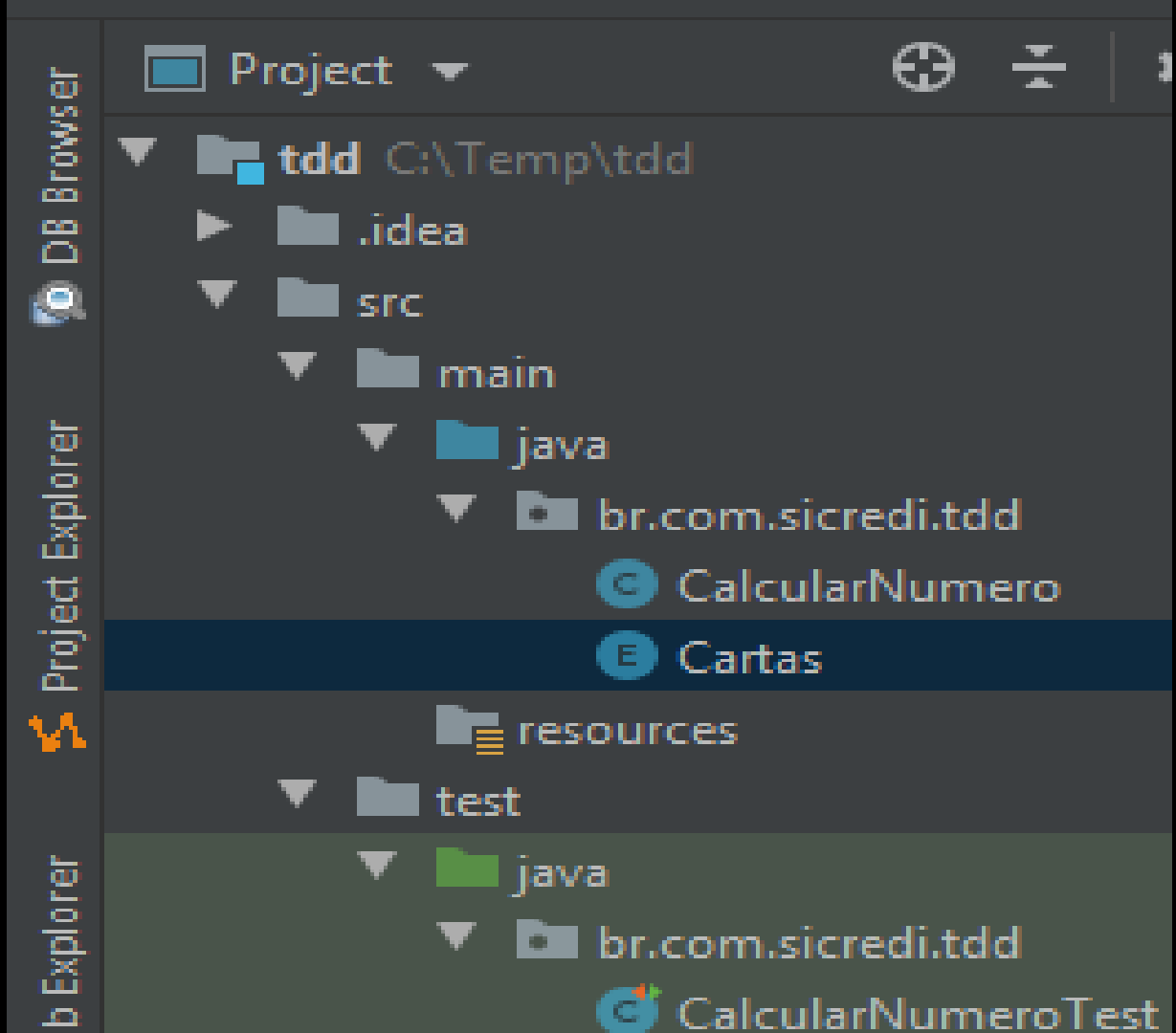
Código está implementando a regra de negócio

Código está complexo, “IFZEIRA”

```java
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

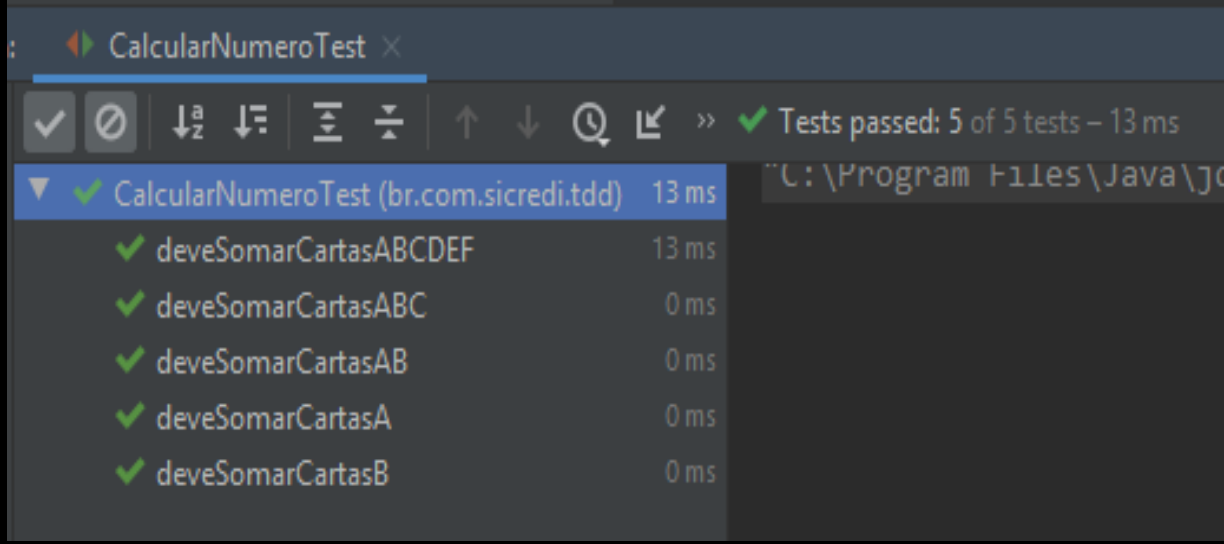
```java
public enum Cartas {

    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

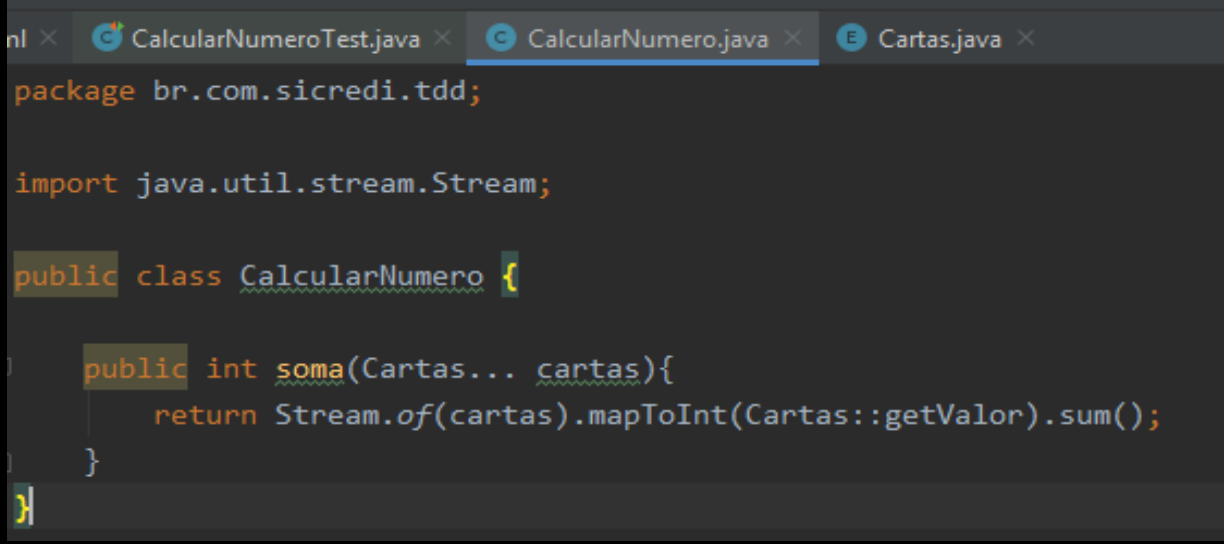
```java
public int soma(String cartas){
    int numeroCalculado = 0;
    if(cartas.contains("A")){
        numeroCalculado = numeroCalculado + Cartas.CARTA_A.getValor();
    }
    if(cartas.contains("B")){
        numeroCalculado = numeroCalculado + Cartas.CARTA_B.getValor();
    }
    if(cartas.contains("C")){
        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



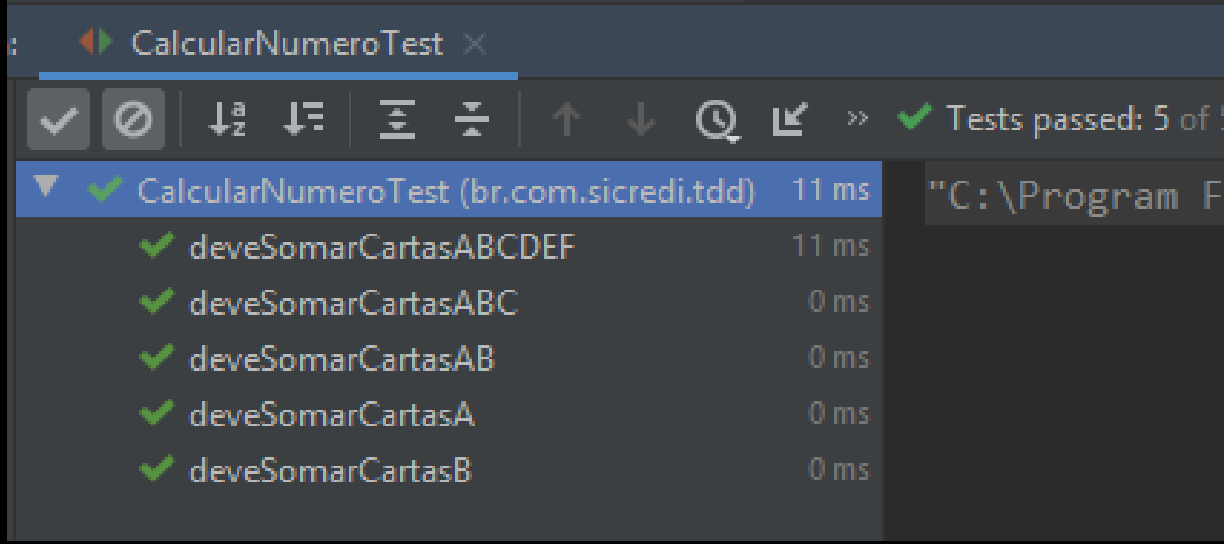
Simplificar a soma para retirar a “IFZEIRA”



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

```java
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



# 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

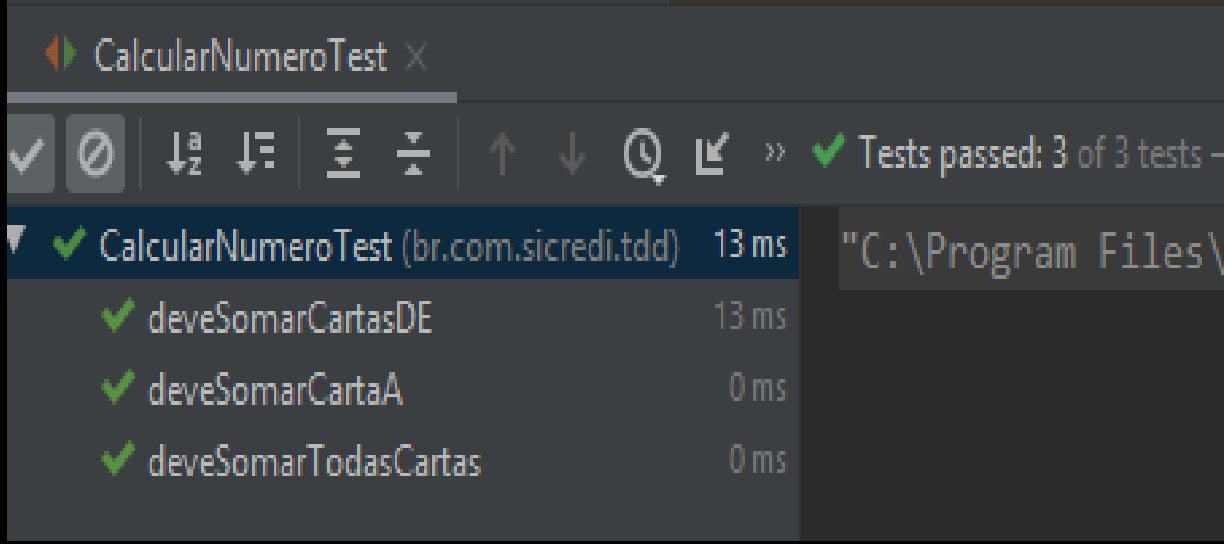
Criando teste que some todas as cartas

Criando teste que some 1 carta

Criando teste que some 2 cartas

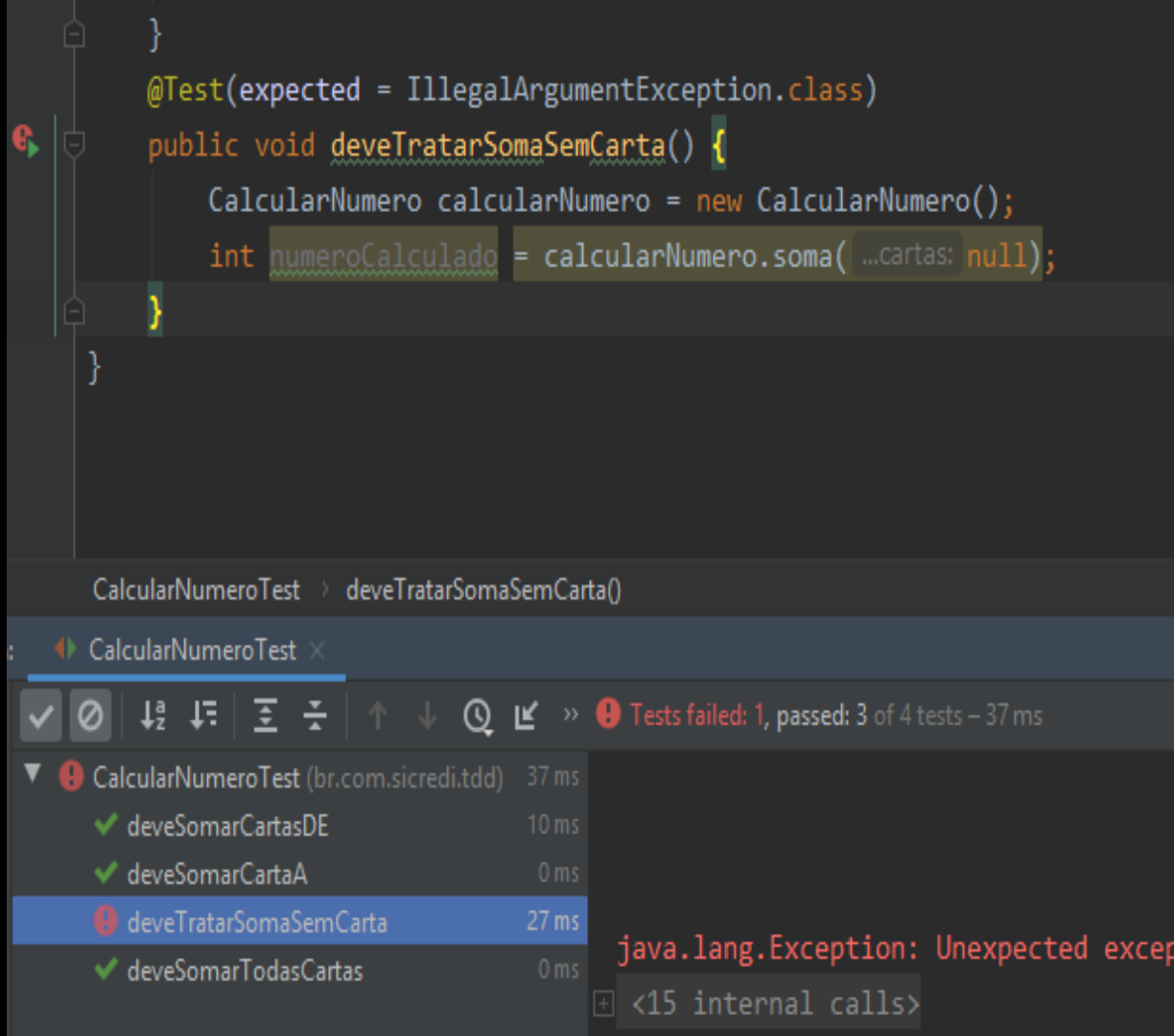
```java
public class CalcularNumeroTest {

    @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



Criando teste sem informar carta

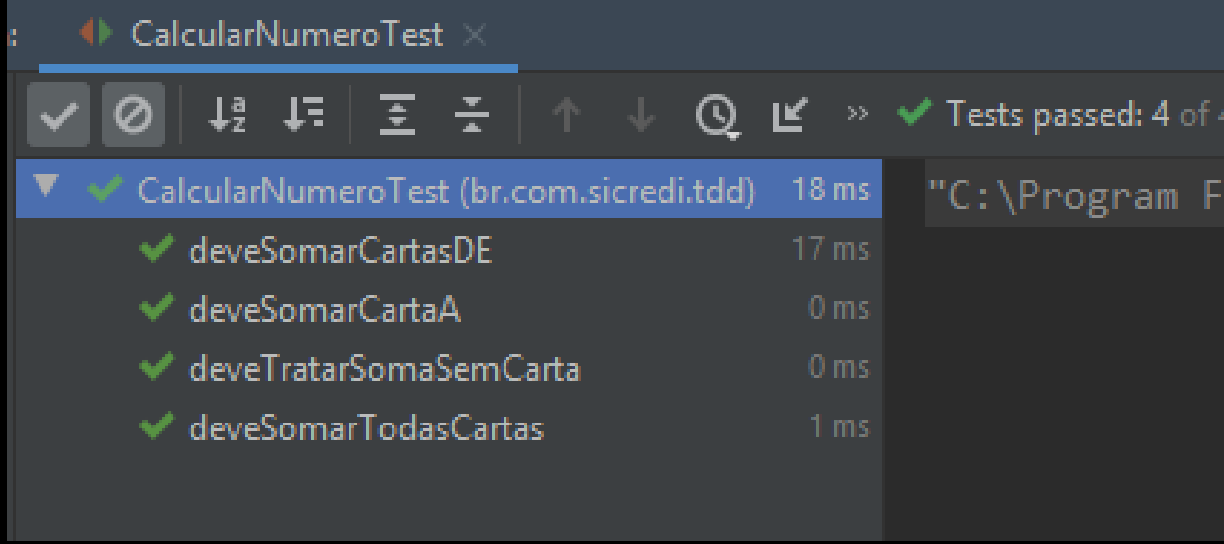
Teste falha

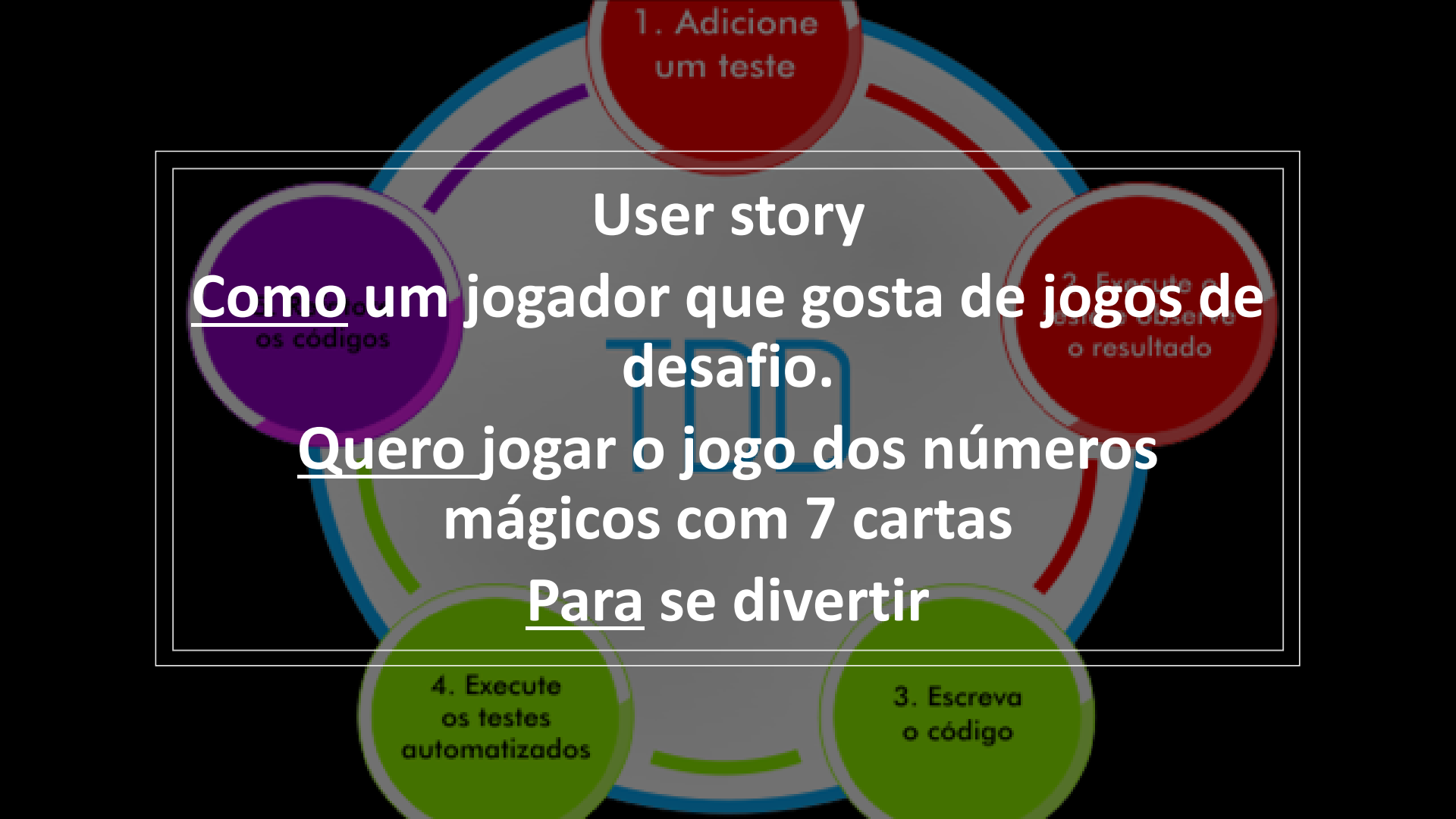


Implementando tratamento para soma sem carta

```java
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






**User story**

**Como um jogador que gosta de jogos de desafio.**

**Quero jogar o jogo dos números mágicos com 7 cartas**

**Para se divertir**

| 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 |

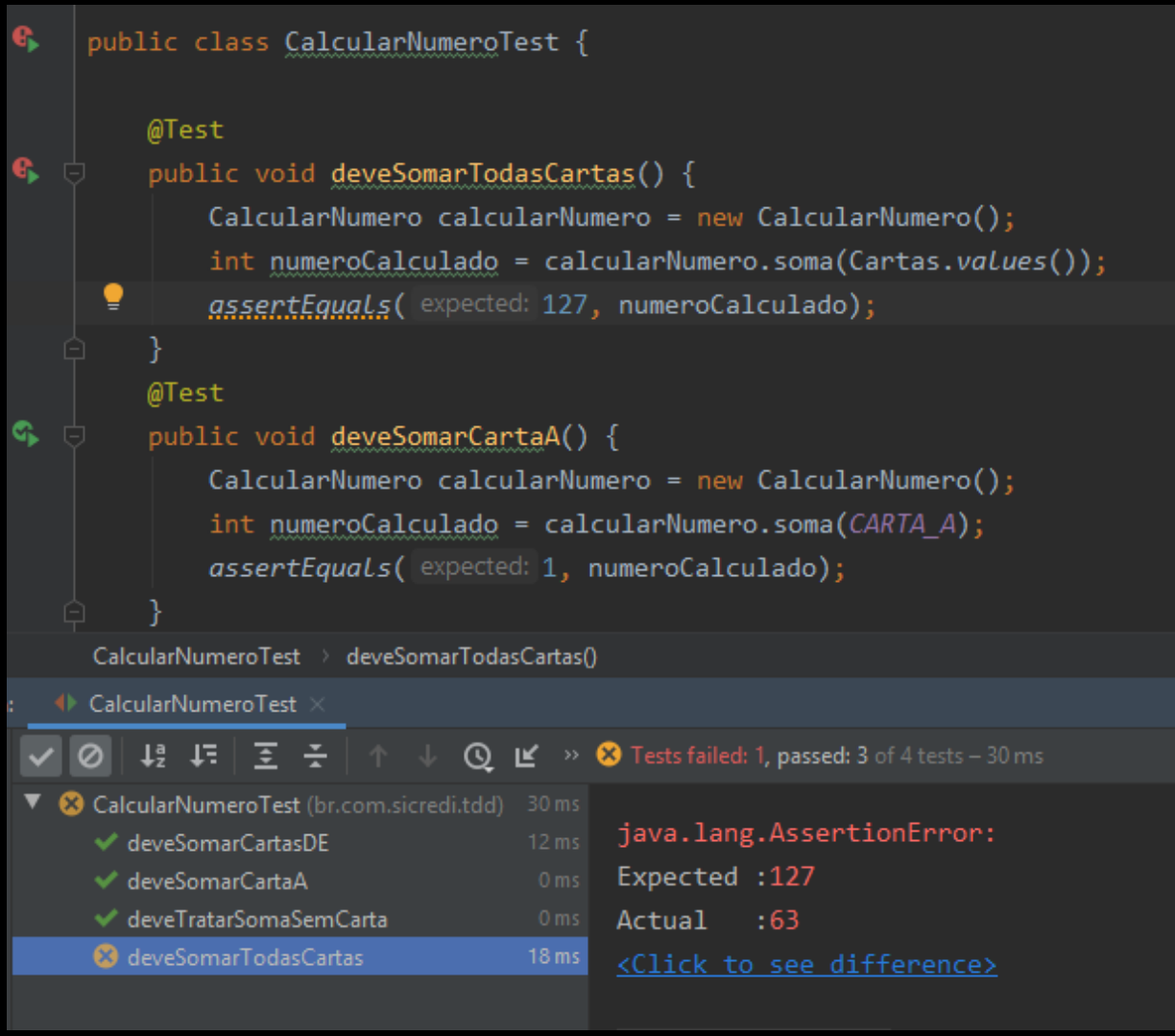
| E | | | |
|---|---|---|---|
| 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |
| 48 | 49 | 50 | 51 |
| 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 |
| 80 | 81 | 82 | 83 |
| 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 |
| 92 | 93 | 94 | 95 |
| 112 | 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

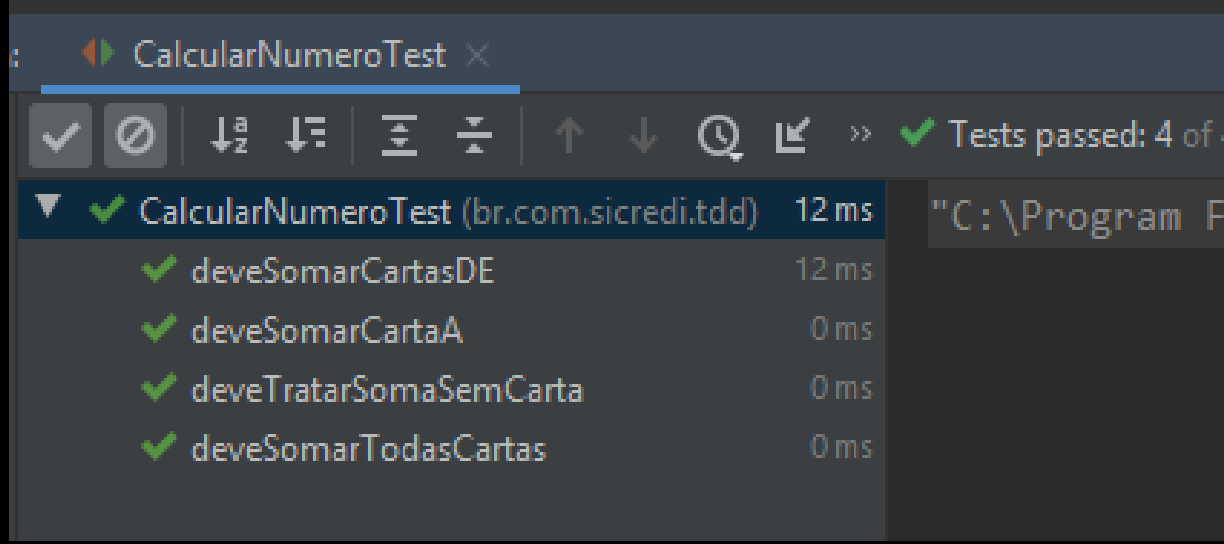
Teste falha, pois não existe a nova carta



Implementando nova carta

```java
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***