

# Tarefa teste de Software com JUnit

## Códigos-fonte das classes implementadas.

### Classe Calculadora

```
1 public class Calculadora {
2
3     public int somar(int a, int b) {
4         return a + b;
5     }
6
7     public int subtrair(int a, int b) {
8         return a - b;
9     }
10
11    public int multiplicar(int a, int b) {
12        return a * b;
13    }
14
15    public double dividir(double a, double b) {
16        if (b != 0) {
17            return a / b;
18        } else {
19            throw new ArithmeticException("Nao divide zero");
20        }
21    }
22 }
```

### Classe NumeroUtil

```
NumeroUtil.java > ...
1 public class NumeroUtil {
2
3     public static boolean ehPar(int numero) {
4         return numero % 2 == 0;
5     }
6
7     public static boolean ehPrimo(int numero) {
8         if (numero <= 1) {
9             return false;
10        }
11        for (int i = 2; i <= Math.sqrt(numero); i++) {
12            if (numero % i == 0) {
13                return false;
14            }
15        }
16        return true;
17    }
18 }
19
```

### Classe ValidadorSenha

```
ValidadorSenha.java > ...
1 public class ValidadorSenha {
2     public boolean validar(String senha) {
3         if (senha == null || senha.length() < 8) {
4             return false;
5         }
6         boolean temMaiuscula = false;
7         boolean temMinuscula = false;
8         boolean temDigito = false;
9         for (char c : senha.toCharArray()) {
10             if (Character.isUpperCase(c)) temMaiuscula = true;
11             if (Character.isLowerCase(c)) temMinuscula = true;
12             if (Character.isDigit(c)) temDigito = true;
13         }
14         return temMaiuscula && temMinuscula && temDigito;
15     }
16 }
```

## Classe CarrinhoDeCompras

```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  public class CarrinhoDeCompras {
5      private Map<String, Double> produtos = new HashMap<>();
6
7      public void adicionarProduto(String nome, double preco) {
8          produtos.put(nome, preco);
9      }
10
11     public void removerProduto(String nome) {
12         produtos.remove(nome);
13     }
14
15     public double calcularTotal() {
16         double total = 0.0;
17         for (double preco : produtos.values()) {
18             total += preco;
19         }
20         return total;
21     }
22 }
```

## Códigos de teste com cobertura adequada.

```
1  import static org.junit.Assert.assertEquals;
2  import static org.junit.Assert.assertThrows;
3
4  import org.junit.Test;
5
6  public class CalculadoraTest {
7      @Test
8      public void testDividir() {
9          Calculadora calc = new Calculadora();
10         assertEquals(expected:2.0, calc.dividir(a:10, b:5), delta:0.0001);
11         assertEquals(-2.5, calc.dividir(-5, b:2), delta:0.0001);
12         assertEquals(expected:0.0, calc.dividir(a:0, b:10), delta:0.0001);
13         assertThrows(expectedThrowable:ArithmeticException.class, () -> {
14             calc.dividir(a:10, b:0);
15         });
16     }
17
18     @Test
19     public void testMultiplicar() {
20         Calculadora calc = new Calculadora();
21         assertEquals(expected:6, calc.multiplicar(a:2, b:3));
22         assertEquals(expected:0, calc.multiplicar(a:0, b:5));
23         assertEquals(-15, calc.multiplicar(-3, b:5));
24         assertEquals(expected:9, calc.multiplicar(-3, -3));
25     }
26
27     @Test
28     public void testSomar() {
29         Calculadora calc = new Calculadora();
30         assertEquals(expected:5, calc.somar(a:2, b:3));
31         assertEquals(expected:0, calc.somar(a:2, -2));
32         assertEquals(-1, calc.somar(-2, b:1));
33         assertEquals(-5, calc.somar(-2, -3));
34     }
35
36     @Test
37     public void testSubtrair() {
38         Calculadora calc = new Calculadora();
39         assertEquals(-1, calc.subtrair(a:2, b:3));
40         assertEquals(expected:4, calc.subtrair(a:2, -2));
41         assertEquals(-3, calc.subtrair(-2, b:1));
42         assertEquals(expected:1, calc.subtrair(-2, -3));
43     }
44 }
```



```

NumeroUtilTest.java > NumeroUtilTest > testEhPrimoComNumerosNaoPrimos()
1  import static org.junit.Assert.assertEquals;
2  import org.junit.Test;
3
4  public class NumeroUtilTest {
5
6      @Test
7      public void testEhParComNumeroPar() {
8          assertEquals(expected:true, NumeroUtil.ehPar(numero:2));
9          assertEquals(expected:true, NumeroUtil.ehPar(numero:0));
10         assertEquals(expected:true, NumeroUtil.ehPar(numero:-4));
11     }
12
13     @Test
14     public void testEhParComNumeroImpar() {
15         assertEquals(expected:false, NumeroUtil.ehPar(numero:3));
16         assertEquals(expected:false, NumeroUtil.ehPar(numero:-1));
17         assertEquals(expected:false, NumeroUtil.ehPar(numero:7));
18     }
19
20     @Test
21     public void testEhPrimoComNumerosPrimos() {
22         assertEquals(expected:true, NumeroUtil.ehPrimo(numero:2));
23         assertEquals(expected:true, NumeroUtil.ehPrimo(numero:3));
24         assertEquals(expected:true, NumeroUtil.ehPrimo(numero:5));
25         assertEquals(expected:true, NumeroUtil.ehPrimo(numero:13));
26         assertEquals(expected:true, NumeroUtil.ehPrimo(numero:17));
27     }
28
29     @Test
30     public void testEhPrimoComNumerosNaoPrimos() {
31         assertEquals(expected:false, NumeroUtil.ehPrimo(numero:1));
32         assertEquals(expected:false, NumeroUtil.ehPrimo(numero:0));
33         assertEquals(expected:false, NumeroUtil.ehPrimo(numero:-3));
34         assertEquals(expected:false, NumeroUtil.ehPrimo(numero:4));
35     }
36 }

```

```

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
import org.junit.Before;
import org.junit.Test;

public class ValidadorSenhaTest {
    private ValidadorSenha validador;

    @Before
    public void setUp() {
        validador = new ValidadorSenha();
    }

    @Test
    public void testSenhaValida() {
        assertTrue(validador.validar(senha:"Senha123"));
        assertTrue(validador.validar(senha:"Abcdef1g"));
    }

    @Test
    public void testSenhaCurta() {
        assertFalse(validador.validar(senha:"Abc12"));
    }

    @Test
    public void testSemMaiuscula() {
        assertFalse(validador.validar(senha:"senha1234"));
    }

    @Test
    public void testSemMinuscula() {
        assertFalse(validador.validar(senha:"SENHA1234"));
    }

    @Test
    public void testSemDigito() {
        assertFalse(validador.validar(senha:"Senhaaaaa"));
    }

    @Test
    public void testSenhaNula() {
        assertFalse(validador.validar(senha:null));
    }
}

```

```

import static org.junit.Assert.assertEquals;
import org.junit.Before;
import org.junit.Test;

public class CarrinhoDeComprasTest {
    private CarrinhoDeCompras carrinho;

    @Before
    public void setUp() {
        carrinho = new CarrinhoDeCompras();
    }

    @Test
    public void testAdicionarProduto() {
        carrinho.adicionarProduto(nome:"Arroz", preco:10.0);
        assertEquals(expected:10.0, carrinho.calcularTotal(), delta:0.0001);
    }

    @Test
    public void testAdicionarVariosProdutos() {
        carrinho.adicionarProduto(nome:"Arroz", preco:10.0);
        carrinho.adicionarProduto(nome:"Feijao", preco:8.5);
        carrinho.adicionarProduto(nome:"Macarrao", preco:5.0);
        assertEquals(expected:23.5, carrinho.calcularTotal(), delta:0.0001);
    }

    @Test
    public void testRemoverProduto() {
        carrinho.adicionarProduto(nome:"Arroz", preco:10.0);
        carrinho.adicionarProduto(nome:"Feijao", preco:8.5);
        carrinho.removerProduto(nome:"Arroz");
        assertEquals(expected:8.5, carrinho.calcularTotal(), delta:0.0001);
    }

    @Test
    public void testRemoverProdutoInexistente() {
        carrinho.adicionarProduto(nome:"Arroz", preco:10.0);
        carrinho.removerProduto(nome:"Feijao");
        assertEquals(expected:10.0, carrinho.calcularTotal(), delta:0.0001);
    }

    @Test
    public void testCalcularTotalVazio() {
        assertEquals(expected:0.0, carrinho.calcularTotal(), delta:0.0001);
    }
}

```

## Prints gerado pelas ferramentas de cobertura (opcional).

### Testes classe Calculadora

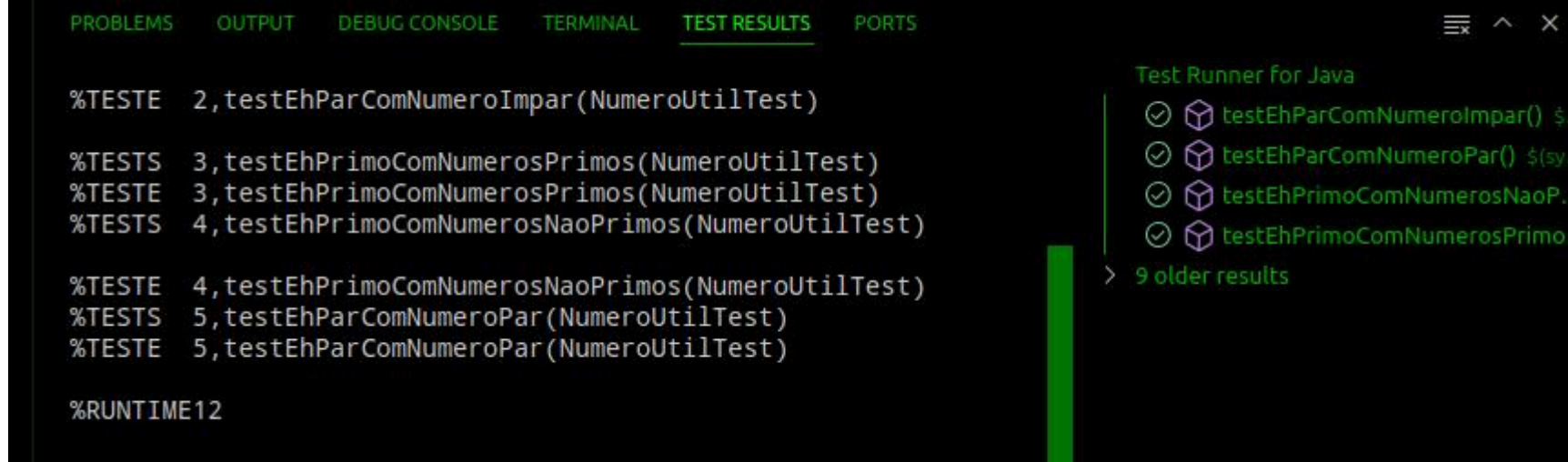


The screenshot displays the test results for the `CalculadoraTest` class. The left pane shows a list of test results with status, count, and method name. The right pane shows a detailed view of the test runner for Java, listing the methods tested and their status.

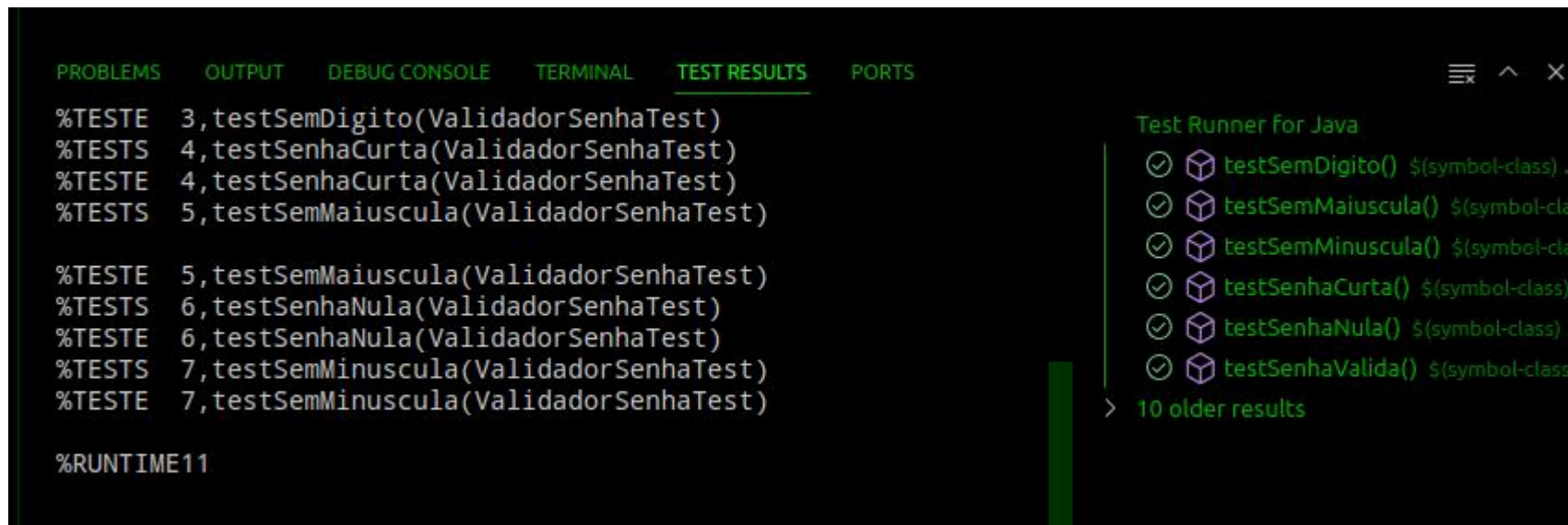
Test Results	Test Runner for Java
%TESTE 2, testSubtrair(CalculadoraTest)	✓ testDividir() \$(symbol-class) Calculadora
%TESTS 3, testSomar(CalculadoraTest)	✓ testMultiplicar() \$(symbol-class) Calculadora
%TESTE 3, testSomar(CalculadoraTest)	✓ testSomar() \$(symbol-class) Calculadora
%TESTS 4, testMultiplicar(CalculadoraTest)	✓ testSubtrair() \$(symbol-class) Calculadora
%TESTE 4, testMultiplicar(CalculadoraTest)	
%TESTS 5, testDividir(CalculadoraTest)	> 8 older results
%TESTE 5, testDividir(CalculadoraTest)	
%RUNTIME11	

### Testes classe Numero Util





## Testes classe Validador Senhas



## Teste Carrinho De Compras

