

Testes parametrizáveis – Java

Geralmente escrevemos testes com valores fixos nos parâmetros, afinal de contas precisamos saber exatamente o que esperar. Mas pode ser interessante usar **diversos parâmetros em um teste**, por exemplo, usar **valores de extremidade** para validar mais ainda o teste:

Indica que o teste vai ser parametrizado

```
@ParameterizedTest
@ValueSource(doubles = {89.0, 99.9, 120.2})
public void should_return_true_when_diet_recommended(Double codersWeight){
    double weight = codersWeight;
    double height = 1.72;

    boolean recommended = BMICalculator.isDietRecommended(weight,height);

    assertTrue(recommended);
}
```

Os valores são definidos na source e injetados no teste através do parâmetro

O nome dos parâmetros podem ser especificados de acordo com a posição do Array com a source

```
@ParameterizedTest(name = "weight={0}, height={1}")
@CsvSource(value = {"89.0, 1.72", "95.2, 1.75", "110.0, 1.78"})
public void should_return_true_when_diet_recommended(Double codersWeight, Double codersHeight){
    double weight = codersWeight;
    double height = codersHeight;

    boolean recommended = BMICalculator.isDietRecommended(weight,height);

    assertTrue(recommended);
}
```

Cada Argumento da string é quebrada em uma posição de um array, com a "," sendo o delimitador. Cada nova String é um novo array com argumentos

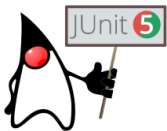
O teste é executado para cada parâmetro informado

```
BMICalculatorTest (test.java) 94 ms
  ✓ should_return_true_when_diet_recom 34 ms
    ✓ [1] 89.0 10 ms
    ✓ [2] 99.9 2 ms
    ✓ [3] 120.2 2 ms
```

Cada conjunto de parâmetros é respectivamente testado

```
BMICalculatorTest (test.java) 147 ms
  ✓ should_return_true_when_diet_recor 147 ms
    ✓ weight=89.0, height=1.72 134 ms
    ✓ weight=95.2, height=1.75 4 ms
    ✓ weight=110.0, height=1.78 9 ms
```

Da pra **pegar parâmetros de arquivos csv** usando a anotação **@CsvFileSource** que recebe o caminho do arquivo como valor, além de poder fazer algumas operações de linha, tipo skipar.



Teste de performance e Assumptions – Java

JUnit tem uma **assertiva bem útil para testar a performance de um método**, testes de performance são geralmente executados em ambientes diferentes do de dev e por isso o Junit tem uma **“assumption”** para ajudar a especificar ambientes em que testes devem ou não ser executados:

```
private String environment = "dev";
```

Em ambiente de desenvolvimento eu não quero executar um teste de performance

```
@Test
public void should_returnCoderWithWorstBMIIn1ms_when_codersListHas10000Elements(){
    assumeTrue(this.environment.equals("prod"));

    List<Coder> coderList = new ArrayList<>();
    for(int i = 0; i <= 10000; i++){
        coderList.add(new Coder( height: 1.0+i, weight: 10.0 +i));
    }

    assertTimeout(Duration.ofMillis(500), () -> BMICalculator.findCoderWithWorstBMI(coderList));
}
```

O limite da operação em 10k de elemento tem que ser meio segundo, 500 ms

```
BMICalculatorTest (test.java)
  ✓ should_throw_ArithmeticException_when_heightZero()
  ✓ should_returnCorrectBMIScoreArray_when_codersListNotEmpty()
  ✓ should_return_false_when_diet_recommended()
  ✗ should_returnCoderWithWorstBMIIn1ms_when_codersListHas10000Elements()
  ✓ should_returnNull_when_codersListIsEmpty()
  > should_return_true_when_diet_recommended(Double, Double)
  ✓ should_returnCoderWithWorstBMI_when_codersListNotEmpty()
```



Nested classes – Java

São **classes internas que podem agrupar um montante de teste**, é usado de forma semântica:

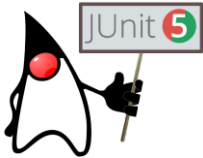
```
Nested
class isDietRecommendedTest{

    @ParameterizedTest(name = "weight={0}, height={1}")
    @CsvSource(value = {"89.0, 1.72", "95.2, 1.75", "110.0, 1.78"})
    public void should_return_true_when_diet_recommended(Double codersWeight, Double codersHeight){...}

    @Test
    public void should_return_false_when_diet_recommended(){...}

    @Test
    public void should_throw_ArithmeticException_when_heightZero(){...}
}

BMI Calculator Test $ isDietRecommendedTest (test.java)
  BMI Calculator Test
    isDietRecommendedTest
      ✓ should_throw_ArithmeticException_when_heightZero()
      ✓ should_return_false_when_diet_recommended()
      ✓ should_return_true_when_diet_recommended(Double, Double)
        ✓ weight=89.0, height=1.72
        ✓ weight=95.2, height=1.75
        ✓ weight=110.0, height=1.78
```



Disable – Java

Serve **para desabilitar algum teste**, é interessante que podem ter condições específicas para desabilitar um teste, como por exemplo um SO diferente:

```
@Test
@Disabled
public void should_returnCoderWithWorstBMI_when_codersListNotEmpty(){...}

@Test
@DisabledOnOs(OS.LINUX)
public void should_returnCoderWithWorstBMI_when_codersListNotEmpty(){...}
```