

Disciplina: **Teste**

Antonio Carlos Ferreira de Almeida - RA:2438194

Aberto: segunda, 13 fev 2023, 08:00

Vencimento: domingo, 26 fev 2023, 23:59

Seguindo um ciclo *Test Driven Development* (TDD), desenvolva as classes abaixo usando JUnit para automatizar os testes do programa que gera um número aleatório em um intervalo

(1) MyRandomNumber.java

```
public class MyRandomNumber {
    Random random = new Random();

    /**
     *
     * @param begin inicio do intervalo
     * @param end fim do intervalo
     * @return retornar um numero aleatorio entre [begin, end]
     * o numero aleatorio retornado nao pode ser igual ao anterior
     * @throws IntervaloInvalidoException
     * essa execucao eh lancada quando begin >= end ou (begin<0 || end<0)
     */

    public int nextRandomNumber(int begin, int end) throws
    IntervaloInvalidoException {

        if(begin < 0)

            throw new IntervaloInvalidoException("begin eh menor que zero");

            //TODO implementar o codigo aqui
            return -1;
        }
    }
}
```

(2) IntervaloInvalidoException.java

```
public class IntervaloInvalidoException extends Exception {

    public IntervaloInvalidoException(String msg) {
        super(msg);
    }
}
```

Entregue o código fonte para responder os itens requeridos em um arquivo único .pdf (coloque o código necessário para responder os itens usando seu editor de texto favorito e salve em um arquivo .pdf para entrega).

Respostas:

01.

```
MyRandomNumberTest.java  *Main.java x  *MyRandomNumber.java
1 package br.gerador.aleatorio;
2
3 public class Main {
4     public static void main(String[] args) throws IntervaloInvalidoException {
5         MyRandomNumber myRandomNumber = new MyRandomNumber();
6         myRandomNumber.nextRandomNumber(-5, 8);
7     }
8 }
9
```

2.

```
1 package br.gerador.aleatorio;
2
3 import java.util.Random;
4
5 /** @param begin inicio do intervalo
6  * @param end fim do intervalo
7  * @return retornar um numero aleatorio entre [begin, end] o numero aleatorio
8  *         retornado nao pode ser igual ao anterior
9  * @throws IntervaloInvalidoException essa excecao eh lancada quando begin >=
10  * end ou (begin<0 || end<0)*/
11
12 public class MyRandomNumber {
13     Random random = new Random();
14
15     public int nextRandomNumber(int begin, int end) throws IntervaloInvalidoException {
16         if (begin < 0)
17             throw new IntervaloInvalidoException("begin eh menor que zero");
18         // TODO implementar o codigo aqui
19         if (begin >= end || begin < 0 || end < 0)
20             throw new IntervaloInvalidoException("Intervalo inválido!");
21
22         int randomNumber = -1;
23         int previusRandomNumber = 0;
24
25         while (randomNumber == -1) {
26             int newRandomNumber = random.nextInt((end - begin) + 1) + begin;
27             if (newRandomNumber != previusRandomNumber) {
28                 randomNumber = newRandomNumber;
29                 previusRandomNumber = newRandomNumber;
30             }
31         }
32         return randomNumber;
33     }
34 }
35
```

3.



```
*MyRandomNumberTest.java  *Main.java  *MyRandomNumber.java  IntervaloInvalidoException.java x
1 package br.gerador.aleatorio;
2
3 public class IntervaloInvalidoException extends Exception{
4
5     public IntervaloInvalidoException(String msg) {
6         super(msg);
7     }
8
9 }
10
```


4.


```
MyRandomNumberTest.java × MyRandomNumber.java IntervalInvalidoException.java
1 package br.gerador.aleatorio;
2
3 import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 public class MyRandomNumberTest {
8     private MyRandomNumber randomNumberGenerator;
9
10    @BeforeEach
11    void setup() {
12        randomNumberGenerator = new MyRandomNumber();
13    }
14
15    @Test
16    void shouldThrowIntervaloInvalidoExceptionWhenBeginIsNegative() {
17        Assertions.assertThrows(IntervaloInvalidoException.class, ()
18            -> randomNumberGenerator.nextRandomNumber(-1, 10));
19    }
20
21    @Test
22    void shouldThrowIntervaloInvalidoExceptionWhenEndIsNegative() {
23        Assertions.assertThrows(IntervaloInvalidoException.class, ()
24            -> randomNumberGenerator.nextRandomNumber(10, -1));
25    }
26
27    @Test
28    void shouldThrowIntervaloInvalidoExceptionWhenBeginIsGreaterThanOrEqualToEnd() {
29        Assertions.assertThrows(IntervaloInvalidoException.class, ()
30            -> randomNumberGenerator.nextRandomNumber(10, 10));
31        Assertions.assertThrows(IntervaloInvalidoException.class, ()
32            -> randomNumberGenerator.nextRandomNumber(10, 5));
33    }
34
35    @Test
36    void shouldReturnRandomNumberInRange() throws IntervaloInvalidoException{
37        int randomNumber = randomNumberGenerator.nextRandomNumber(10, 20);
38        Assertions.assertTrue(randomNumber >= 10 && randomNumber <= 20);
39    }
40
41    @Test
42    void shouldReturnDifferentRandomNumberOnEachInvocation() throws IntervaloInvalidoException{
43        int randomNumber1 = randomNumberGenerator.nextRandomNumber(10, 20);
44        int randomNumber2 = randomNumberGenerator.nextRandomNumber(10, 20);
45        int randomNumber3 = randomNumberGenerator.nextRandomNumber(10, 20);
46        Assertions.assertNotEquals(randomNumber1, randomNumber2);
47        Assertions.assertNotEquals(randomNumber1, randomNumber3);
48        Assertions.assertNotEquals(randomNumber2, randomNumber3);
49    }
50 }
```


6.

Finished after 0,304 seconds

Runs: 5/5  Errors: 0  Failures: 0



>  MyRandomNumberTest [Runner: JUnit 5] (0,066 s)

 Failure Trace 