MÓDULO 1 TESTE 2

1) Considere o código abaixo:

```
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;

public class FahrenheitCelciusConverterTest {

    @Test
    public void shouldConvertCelciusToFahrenheit() {
        assertEquals(32, FahrenheitCelciusConverter.toFahrenheit(0));
        assertEquals(98, FahrenheitCelciusConverter.toFahrenheit(37));
        assertEquals(212, FahrenheitCelciusConverter.toFahrenheit(100));
    }

    @Test
    public void shouldConvertFahrenheitToCelcius() {
        assertEquals(0, FahrenheitCelciusConverter.toCelcius(32));
        assertEquals(37, FahrenheitCelciusConverter.toCelcius(100));
        assertEquals(100, FahrenheitCelciusConverter.toCelcius(212));
    }
}
```

Implemente a classe FahrenheitCelciusConverter. Empregue **throw** para lançar exceções em caso de serem informados argumentos inválidos para os métodos **toCelsius** e **toFahrenheit**. Execute FahrenheitCelciusConverterTest após fazer sua implementação. A resposta para essa questão é o fonte da classe FahrenheitCelciusConverter e o report gerado pelo JUnit 5.

2.Considere o projeto que se segue:

https://drive.google.com/file/d/1kPKlpkTDXK72bP43dYfpbXS1meXlHP2a/view?usp=share_link

Programe testes com JUnit 5 para testar os seguinte cenários:

- Quando criado, o carrinho de compras tem 0 itens;
- Quando vazio, o carrinho de compras tem 0 itens;
- Quando um novo produto é adicionado, o número de itens deve ser incrementado;
- Quando um novo produto é adicionado, o novo saldo deve ser a soma do

saldo anterior mais o custo do novo produto;
- Quando um item é removido, o número de itens deve ser diminuído;
- Quando um produto que não está no carrinho é removido, uma ProductNotFoundException deve ser lançada.

3. Considere as classes que se seguem:

```
public class Calculator
{
public
   double add(double number1, double number2)
   {
       return number1 + number2;
   }
}

public class CalculatorTest {

   //Bloco 1
   private int nbErrors = 0;

   void testAdd() {
       Calculator calculator = new Calculator();
       double result = calculator.add(10, 50);
       if (result != 60) {
           throw new IllegalStateException("Bad result: " + result);
       }
   }

   //Bloco 2
   public static void main(String[] args) {
       CalculatorTest test = new CalculatorTest();
       try {
           test.testAdd();
       } catch (Throwable e) {
           test.nbErrors++;
           e.printStackTrace();
       }
       if (test.nbErrors > 0) {
                   throw new IllegalStateException("There were " +
test.nbErrors
                  + " error(s)");
       }
   }
```

```
}
```

Reescreva CalculatorTest com JUnit 5.

4. Escolha ou crie uma aplicação com mais de 5 classes escritas em Java (pode ser a mesma que você utilizou no teste passado). Programe pelo menos 5 testes com JUnit para a suas classes. Empregue pelo menos dois tipos de métodos assert apresentados na tabela que se segue:

Método assert	Propósito
assertAll	Avalia se nenhum dos executáveis fornecidos lança exceções. Um executável é um objeto do tipo org.junit.jupiter.api.function.Executable.
assertArrayEquals	Avalia se o array esperado e o array de entrada são iguais.
assertEquals	Avalia se os valores esperados e os valores de entrada são iguais.
assertX(..., String message)	Asserção que entrega a mensagem fornecida à estrutura de teste se a asserção falhar.
assertX(..., Supplier<String>messageSupplier)	Asserção que entrega a mensagem fornecida à estrutura de teste se a asserção falhar.