Desafio: É comum encontrarmos classes cheias de ifs, com regras de negócio complicadas. Veja a classe abaixo:

```
import java.util.ArrayList;
import java.util.List;

public class FiltroDeLances {

    public List<Lance> filtra(List<Lance> lances) {
        ArrayList<Lance> resultado = new ArrayList<Lance>();

        for(Lance lance : lances) {
            if(lance.getValor() > 1000 && lance.getValor() < 3000)
                    resultado.add(lance);
            else if(lance.getValor() > 500 && lance.getValor() < 700)
                    resultado.add(lance);
            else if(lance.getValor() > 5000)
                    resultado.add(lance);
        }

        return resultado;
    }
}
```

Já começamos a escrever os testes para essa classe, mas ela está incompleta. Levante os cenários que faltam e complete a classe de testes abaixo:

```
import static org.junit.Assert.assertEquals;

import java.util.Arrays;
import java.util.List;

import org.junit.Test;

public class FiltroDeLancesTest {

    @Test
    public void deveSelecionarLancesEntre1000E3000() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao,2000),
                new Lance(joao,1000),
                new Lance(joao,3000),
                new Lance(joao, 800)));

        assertEquals(1, resultado.size());
        assertEquals(2000, resultado.get(0).getValor(), 0.00001);
    }

    @Test
    public void deveSelecionarLancesEntre500E700() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao,600),
                new Lance(joao,500),
                new Lance(joao,700),
                new Lance(joao, 800)));

        assertEquals(1, resultado.size());
        assertEquals(600, resultado.get(0).getValor(), 0.00001);
    }
}
```

Opinião do instrutor

Os testes que faltam cobrem os seguintes cenários: - Lances maiores que 5000 devem ser selecionados; - Lances menores que 500 devem ser eliminados; - Lances entre 3000 e 5000 devem ser eliminados.

```
    @Test
    public void deveSelecionarLancesMaioresQue5000() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao,10000),
                new Lance(joao, 800)));

        assertEquals(1, resultado.size());
        assertEquals(10000, resultado.get(0).getValor(), 0.00001);
    }

    @Test
    public void deveEliminarMenoresQue500() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao,400),
                new Lance(joao, 300)));

        assertEquals(0, resultado.size());
    }

    @Test
    public void deveEliminarEntre700E1000() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao, 800),
                new Lance(joao, 1000),
                new Lance(joao, 700),
                new Lance(joao, 900)));

        assertEquals(0, resultado.size());
    }

    @Test
    public void deveEliminarEntre3000E5000() {
        Usuario joao = new Usuario("Joao");

        FiltroDeLances filtro = new FiltroDeLances();
        List<Lance> resultado = filtro.filtra(Arrays.asList(
                new Lance(joao,4000),
                new Lance(joao, 3500)));

        assertEquals(0, resultado.size());
    }
```

Veja que podemos escrever mais testes, afinal essa classe é realmente complexa.