Lista de Exercícios 10 – Testes Automatizados

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

Herculano De Biasi
herculano.debiasi@unoesc.edu.br

Lista de Exercícios 10

- Na última aula foi criada a classe `Calculadora` com o método `somar()`

modulo2 - Calculadora.java

```java
public class Calculadora {
    public int somar(int num1, int num2) {
        return num1 + num2;
    }
}
```

I. Acrescente agora os métodos `subtrair()`,`multiplicar()` e `dividir()` implementando as respectivas operações

Lista de Exercícios 10

- Na última aula foi criada a classe `CalculadoraTest` contendo 4 casos de teste para o método `somar()`

2. Siga o modelo ao lado e acrescente testes semelhantes para os métodos `subtrair()` e `multiplicar()`

3. Implemente também testes para o método `dividir()`, exceto as situações de divisão por zero, que serão vistas na próxima semana

modulo2 - CalculadoraTest.java

```java
public class CalculadoraTest {
    @Test
    @DisplayName("Deveria somar dois números positivos")
    public void deveriaSomarDoisNumerosPositivos() {
        // Cenário (arrange)
        Calculadora calc = new Calculadora();

        // Execução (act)
        int soma = calc.somar(41, 1);

        // Verificação (assert)
        Assert.assertEquals(42, soma);
    }

    @Test
    public void deveriaSomarUmNumeroPositivoEZero() {
        Calculadora calc = new Calculadora();
        int soma = calc.somar(42, 0);
        Assert.assertEquals(42, soma);
    }

    @Test
    public void deveriaSomarUmNumeroPositivoEUmNegativo() {
        Calculadora calc = new Calculadora();
        int soma = calc.somar(5, -5);
        Assert.assertEquals(0, soma);
    }

    @Test
    public void deveriaSomarDoisNumerosNegativos() {
        Calculadora calc = new Calculadora();
        int soma = calc.somar(-2, -3);
        Assert.assertEquals(-5, soma);
    }
}
```

UNOESC

Lista de Exercícios 10

4. No roteiro do problema do Leilão, foi criada a classe `TestaLeilao`. Modifique o nome da classe (e, consequentemente o nome do arquivo Java) para `LeilaoTest` de forma a seguir boas práticas de desenvolvimento

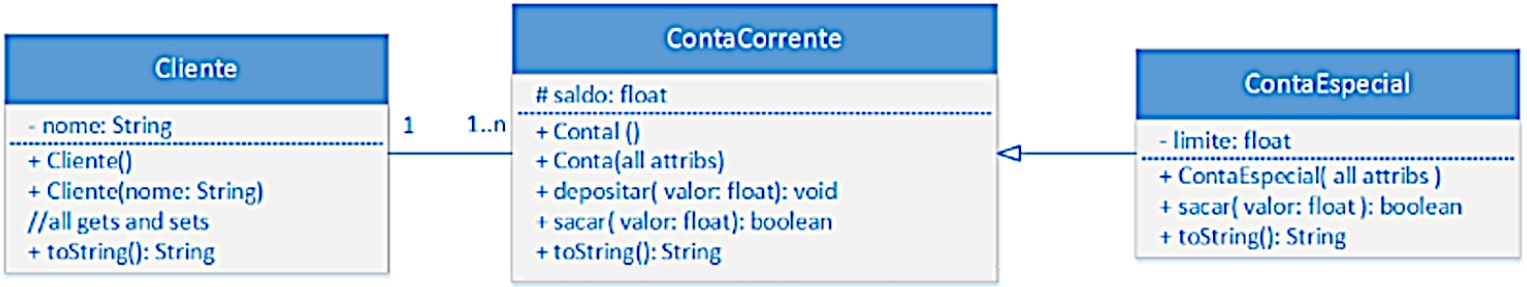
5. Crie outros métodos de teste para o programa do Leilão, como
   - Lances em ordem decrescente
   - Lances em ordem aleatória (sem nenhuma ordem específica)
   - Todos os lances com o mesmo valor
   - Apenas um lance na lista

Lista de Exercícios 10

6. Como provocação, pense agora no cenário em que nenhum lance seja feito
   - Crie um caso de teste para este cenário e não inclua nenhum lance nele
   - Execute a unidade de teste, o teste irá falhar pois o resultado esperado não é o resultado obtido pelo método `avalia()`
   - Quais você acha que seriam os valores máximos e mínimos esperados?
   - Como modificar o método `avalia()` da classe `Avaliador` para tratar essa situação?
   - Descreva como seria a sua solução em um arquivo texto

Lista de Exercícios 10

7. Crie classes de forma a representar o diagrama a seguir



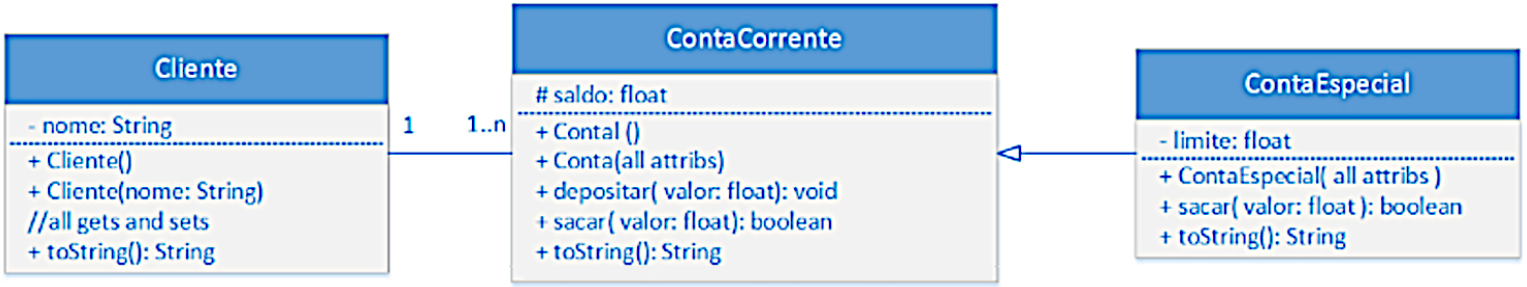
- Clientes podem ter uma ou mais contas, implemente isso através de uma lista
- Clientes de `ContaCorrente` não podem ter saldo negativo, ou seja, o método `sacar` deve prever essa situação
- A classe `ContaEspecial` herda da classe `ContaCorrente`
- Clientes que possuem conta especial possuem um limite de crédito; dessa forma, podem fazer saques até esse valor limite, mesmo que não possuam saldo suficiente na conta
- O construtor da classe `ContaEspecial` deve receber como parâmetro, além dos parâmetros da superclasse, o limite que o banco disponibiliza para o cliente
- Sobrescreva o método `sacar` na classe `ContaEspecial`, de modo que o cliente possa ficar com saldo negativo até o valor de seu limite - note que o atributo saldo da classe `ContaCorrente` deve ser do tipo `protected` para que possa ser modificado na subclasse

UNOESC

Apoiadores: UNOESC, CITEOSC, INOVALE, Centro de Inovação Videira, DEV-TI, SC TEC, fapesc, Governo de Santa Catarina

Lista de Exercícios 10

7. Crie classes de forma a representar o diagrama a seguir



- Com as classes de domínio prontas, crie agora duas classes de teste, uma para `ContaCorrente` e outra para `ContaEspecial`
- Na classe de teste da `ContaCorrente` faça testes que verifiquem se os métodos `depositar` e `sacar` estão funcionando corretamente (o saldo não pode ser negativo)
- Na classe de teste da `ContaEspecial` faça testes que verifiquem se os métodos `depositar` e `sacar` estão funcionando corretamente (o saldo não pode ser menor do que o limite de crédito)