FIAP GRADUAÇÃO

Prof. Renato Jardim Parducci

GOVERNANÇA E QUALIDADE DE SOFTWARE

DISCIPLINA: **COMPLIANCE & QUALITY ASSURANCE**
PROJETO DE SISTEMAS APLICADO AS MELHORES PRÁTICAS EM QUALIDADE DE SOFTWARE E GOVERNANÇA DE TI

AULA:
4 – EXEMPLIFICAÇÃO DE COMO FERRAMENTAS E PROCESSOS IMPACTAM A QUALIDADE E GOVERNANÇA – JUNIT

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR

AGENDA DA AULA

- ✓ Exemplificação do impacto de processos de trabalho e ferramentas sobre a Governança e a Qualidade
- ✓ Uso do JUNIT para criar testes para programas JAVA

GERENCIAMENTO DO TESTE DE SOFTWARE

Teste é uma atividade exaustiva e muitas vezes pouco interessante aos olhos do programador mas,... Quando testar se torna também uma atividade de programação, tudo fica mais divertido (os programadores ficam estimulados a testar)!

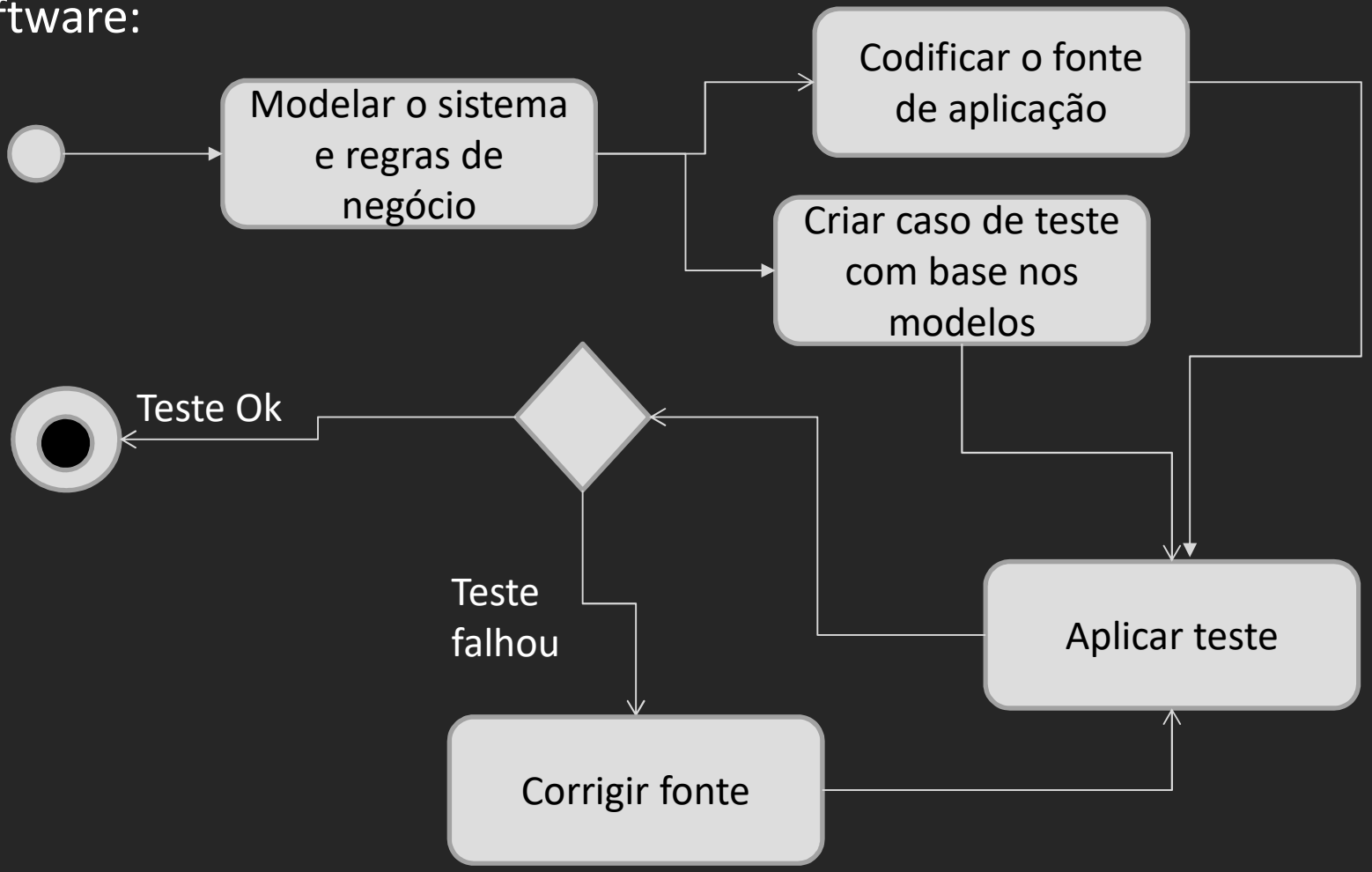
GERENCIAMENTO DO TESTE DE SOFTWARE

Utilizar técnicas apropriadas e ferramentas adequadas para testar programas de aplicação permite:

- Disciplinar o formato dos testes;
- Documentar os casos de teste;
- Aplicar um mesmo teste múltiplas vezes, se necessário (repetir);
- Aproveitar um teste criado por um programador por outras pessoas do time de desenvolvimento (reusar);
- Agilizar a execução dos casos de testes e avalição dos resultados;
- Avaliar se os testes criados cobrem as situações previstas na lógica de um programa de aplicação (análise de cobertura dos testes).

GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos trabalhar neste momento com o processo convencional de teste de software:



GERENCIAMENTO DO TESTE DE SOFTWARE

Para criar os casos de testes, vamos utilizar a ferramenta aplicada a programas JAVA:

GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos usar JUNIT dentro do Eclipse, aprendendo na prática a criar os testes automatizados:

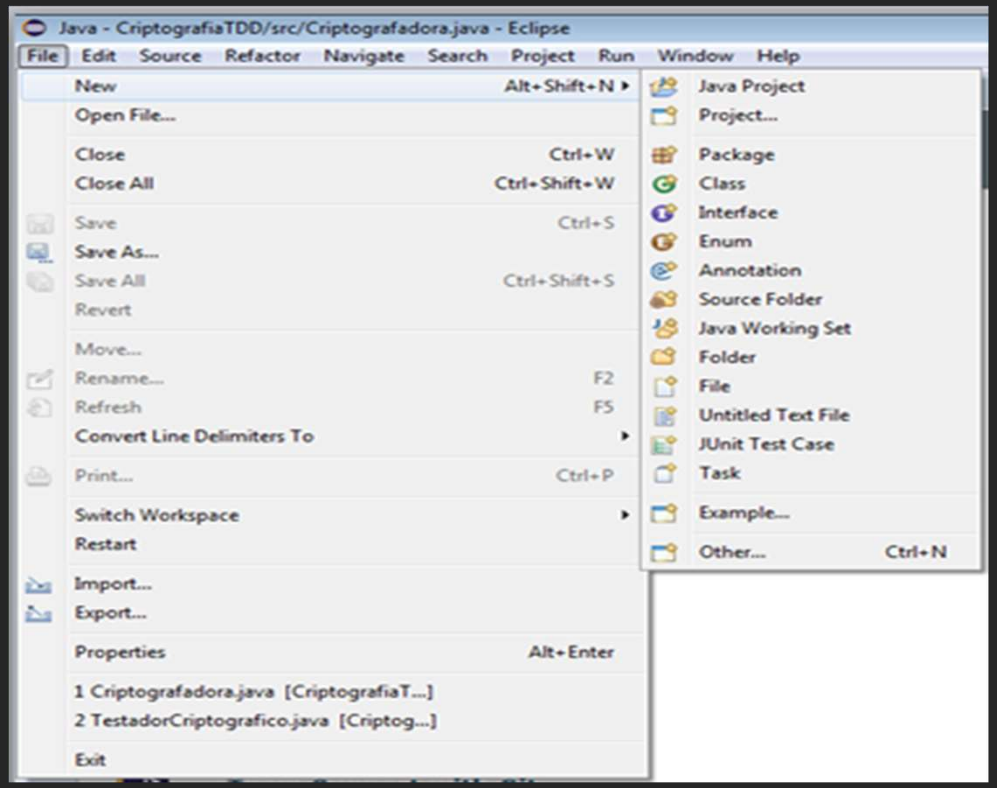
GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos iniciar pela criação de uma Classe JAVA para trabalharmos os testes.

Nos exemplos a seguir, é considerado que a Classe foi escrita exatamente conforme os algoritmos de especificação e modelo UML que foram definidos para seus atributos e métodos.

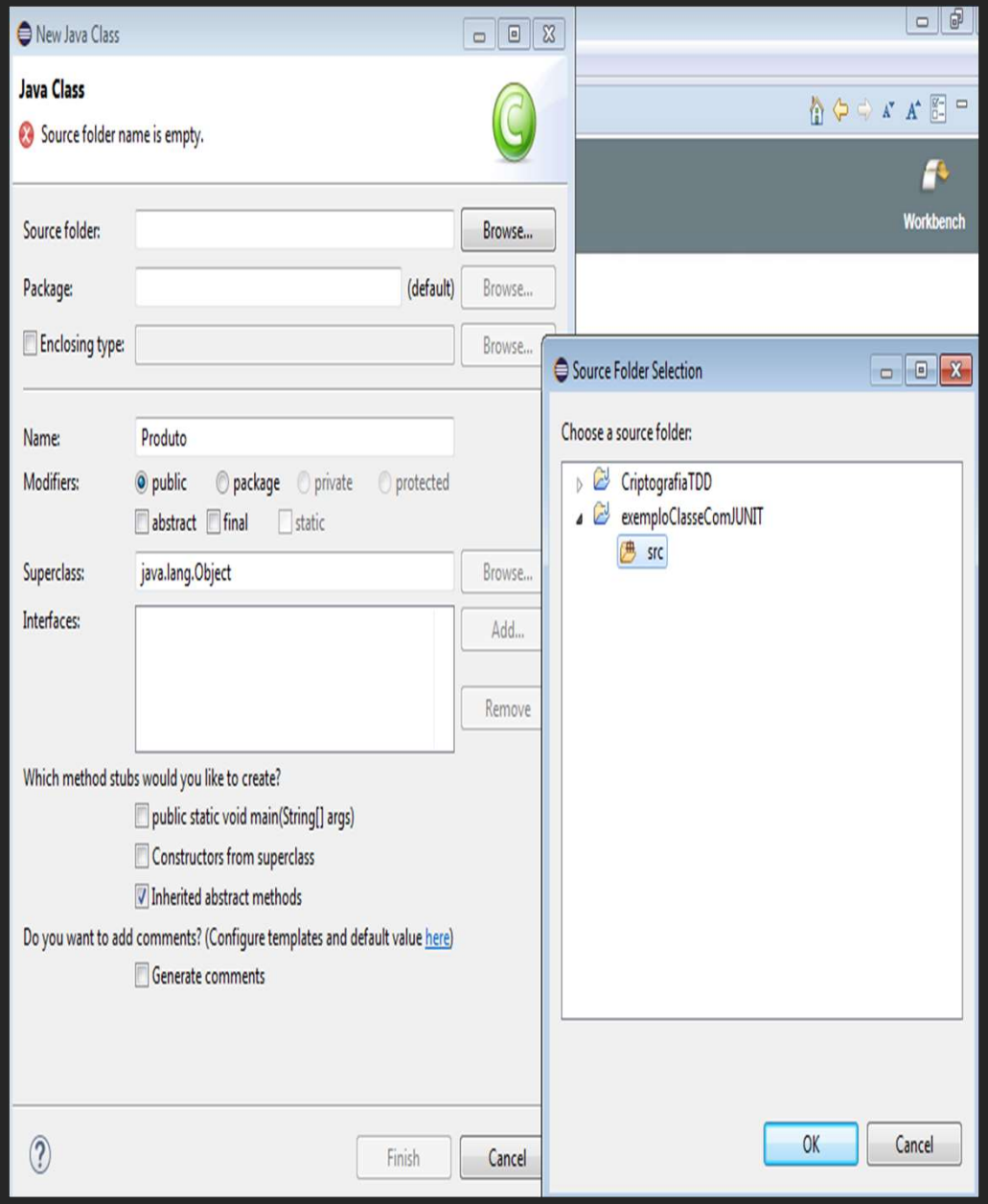
Queremos confirmar que ela funciona adequadamente, através dos testes.

Crie um projeto no Eclipse e depois, ... a Classe JAVA Calculadora, descrita ao lado *(new JAVA Class EJB)*



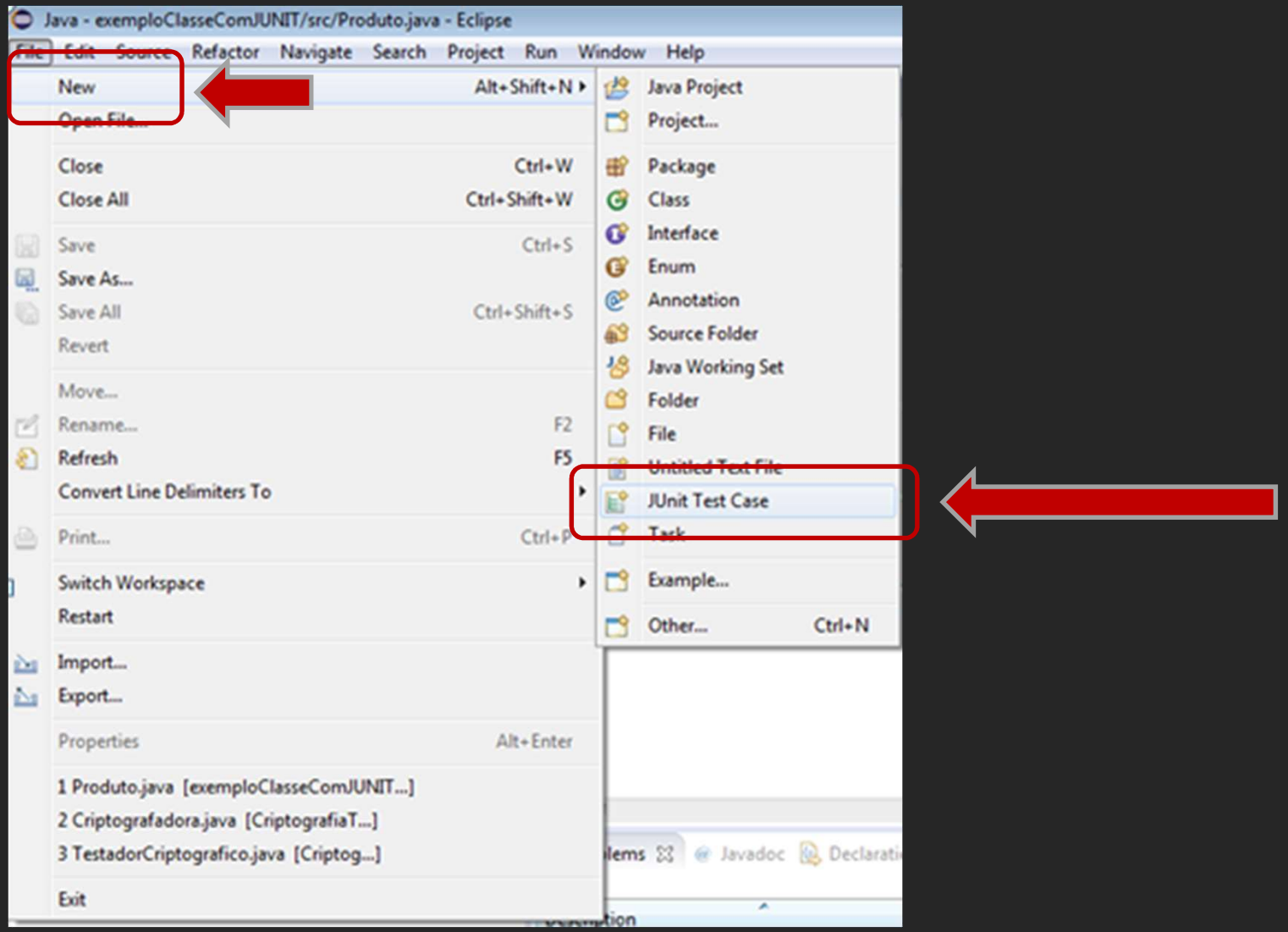

```
public class Calculadora{

    // atributo
    private int resultado = 0;

    // método somar
    public int somar( int n1, int n2 ){

       resultado = n1 + n2;
       return resultado;
    }

    // método subtrair
    public int subtrair( int n1, int n2 ){

       resultado = n1 - n2;
       return resultado;
    }

    // método multiplicar
    public int multiplicar( int n1, int n2 ){

       resultado = n1 * n2;
       return resultado;
    }

    // método dividir
    public int dividir( int n1, int n2 ){

       resultado = n1 / n2;
       return resultado;
    }
}
```

GOVERNANÇA E QUALIDADE DE SOFTWARE

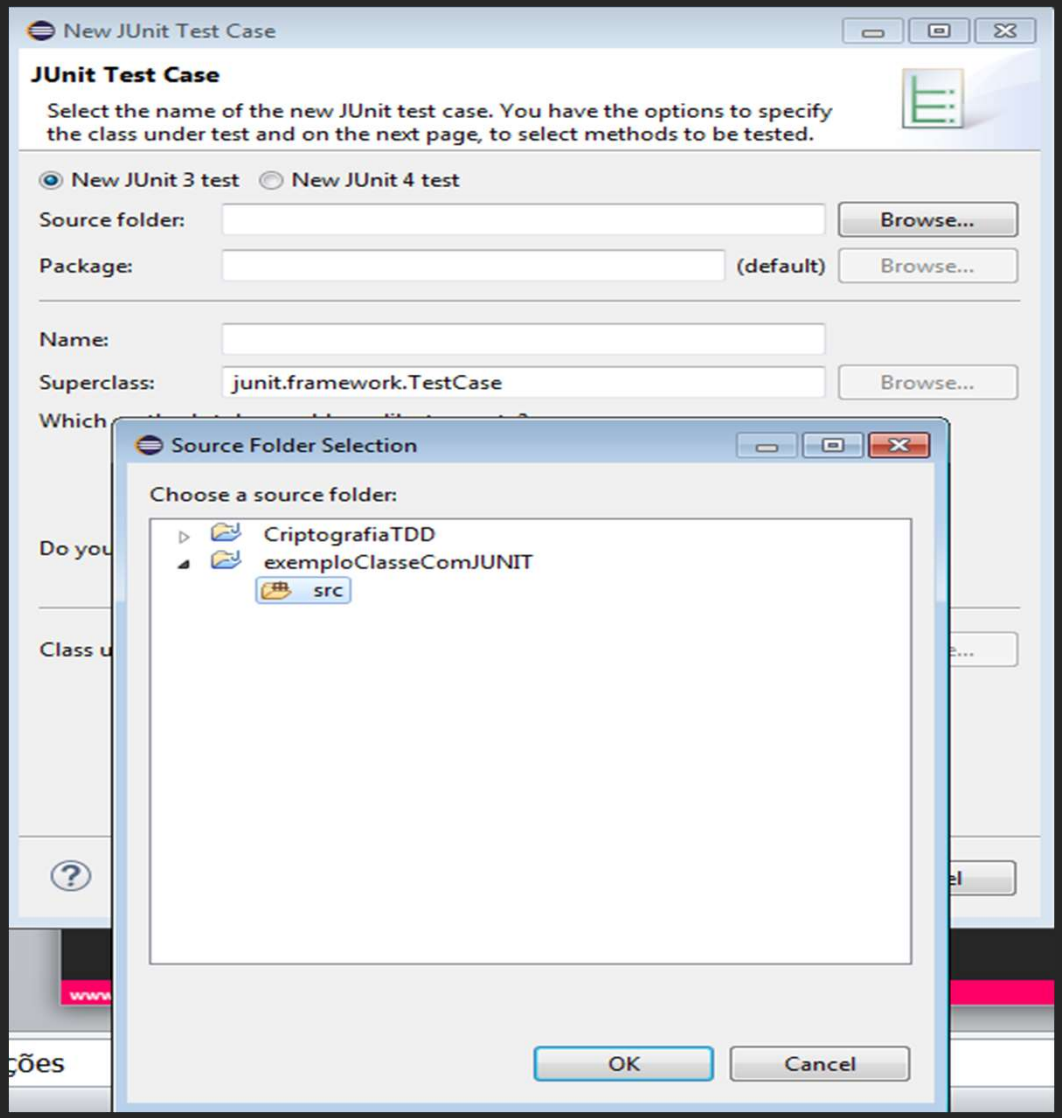



```
public class Calculadora{

    // atributo
    private int resultado = 0;

    // método somar
    public int somar( int n1, int n2 ){

       resultado = n1 + n2;
       return resultado;
    }

    // método subtrair
    public int subtrair( int n1, int n2 ){

       resultado = n1 - n2;
       return resultado;
    }

    // método multiplicar
    public int multiplicar( int n1, int n2 ){

       resultado = n1 * n2;
       return resultado;
    }

    // método dividir
    public int dividir( int n1, int n2 ){

       resultado = n1 / n2;
       return resultado;
    }
}
```

Agora, vamos aos testes...
Crie uma JUNIT Test Case (new JUNIT)



Dê o nome de TesteCalculadora para a Classe de teste que será criada.



GERENCIAMENTO DO TESTE DE SOFTWARE

O processo que vamos seguir para realizar a criação dos testes será:

1º) Criar uma Classe de Teste para Classe de Implementação;

2º) Criar um Método de Teste diferente dentro da Classe de Teste para cada simulação de comportamento que for necessária (um Método para cada Caso de Teste).

3º) Criar e aplicar um Método de Teste por vez, isolando problemas (keep it simple – faça as coisas de forma simples).

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

import junit.framework.TestCase;

public class TesteCalculadoraTest extends TestCase {

    /**
     * Teste de somar na Calculadora.
     */
    @Test
    public void testeSomar() {
        int nro1 = 5;
        int nro2 = 5;
        Calculadora calc= new Calculadora();
        int resultadoEsperado = 10;
        int resultadoReal= calc.somar(nro1, nro2);
        assertEquals(resultadoEsperado, resultadoReal);
    }

}
```

Implementação do Método de teste SOMA, dentro da Classe de Teste de um Objeto Calculadora

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

public class TesteCalculadoraTest extends TestCase {

        ...

    /**
     * Teste de subtrair na  Calculadora.
     */
    @Test
    public void testeSubtrair() {
        int nro1 = 5;
        int nro2 = 3;
        Calculadora calc = new Calculadora();
        int resultadoEsperado= 2;
        int resultadoReal= calc.subtrair(nro1, nro2);
        assertEquals(resultadoEsperado resultadoReal);
    }

}
```

Acrescente esse Método de teste da SUBTRAÇÃO logo em seguida do Método de teste de SOMA que você fez anteriormente

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

public class TesteCalculadoraTest extends TestCase {

   • • •

   /**
    * Teste de multiplicar na Calculadora.
    */
   @Test
   public void testeMultiplicar() {
      int nro1 = 3;
      int nro2 = 3;
      Calculadora calc = new Calculadora();
      int resultadoEsperado = 9;
      int resultadoReal = calc.multiplicar(nro1, nro2);
      assertEquals(resultadoEsperado, resultadoReal);
   }
}
```

Acrescente esse Método de teste da MULTIPLICAÇÃO logo em seguida do Método de teste de SUBTRAÇÃO que você fez anteriormente

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

public class TesteCalculadoraTest extends TestCase {

   • • •

    /**
     * Teste de dividir na Calculadora.
     */
    @Test
    public void testeDividir() {
        int nro1 = 6;
        int nro2 = 2;
        Calculadora calc = new Calculadora();
        int resultadoEsperado= 3;
        int resultadoReal = calc.dividir(nro1, nro2);
        assertEquals(resultadoEsperado, resultadoReal);
    }

}
```

Acrescente esse Método de teste da DIVISÃO logo em seguida do Método de teste de MULTIPLICAÇÃO que você fez anteriormente

Agora, crie a Classe descrita ao lado e a JUNIT para testar todos os métodos da Classe.

Faça os testes para criar um objeto e instanciá-lo e depois testar a recuperação de dados.

```
public class  Produto{

    private double peso;
    private double altura;

    public double getPeso() {
      return peso;
    }

    public void setPeso(double peso) {
      this.peso = peso;
    }

    public double getAltura() {
      return altura;
    }

    public void setAltura(double altura) {
      this.altura = altura;
    }

}
```

Boa diversão!

Agora você conhece um jeito inteligente de controlar suas versões de arquivos com total segurança!

Aproveite esse ambiente em todas as suas disciplinas de prograação!

D ú v i d a s

EXEMPLIFICAÇÃO DE COMO FERRAMENTAS E PROCESSOS IMPACTAM A QUALIDADE E GOVERNANÇA – JUNIT

FIM

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR