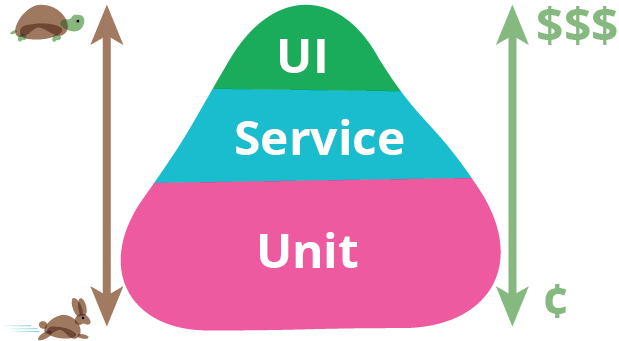
Teste Unitário.

**O que são teste unitário? Para que server? Como desenvolvê-los?**

O teste unitário consiste no teste da menor parte testável do seu código. Por exemplo se você programa em uma linguagem que suporte orientação a objeto, a menor parte testável do seu código é um método. Então o seu teste unitário seria um teste sobre qualquer método criado.

**Para que serve?**

O teste unitário serve para que o programador consiga testar seu código de forma rápida, deixando de gastar muito dinheiro com softwares gigantes que levam muito tempo para configurar e testar as funcionalidades de todo seu código.



Nesta imagem, ao analisa-la no primeiro instante, não entendi muita coisa, vi alguns métodos de teste umas setas com animais do lado e outra com algumas cifras. Então aos pouco essa imagem me mostrava o que realmente um teste unitário é, uma lebre rápida e barata. É melhor uma lebre rápida que custa pouco a uma tartaruga lenta que curta muito dinheiro.

Mas para entender melhor o teste unitário, deveremos mostrar na prática.

**Teste unitário de métodos de uma calculadora.**

**public** **class** Calculadora {

**private** **double** soma;

**private** **double** subtrair;

**private** **double** multiplicar;

**private** **double** dividir;

**public** **double** somar(**double** a, **double** b) {

**this**.soma = a+ b;

System.***out***.println("Soma: " + **this**.soma);

**return** **this**.soma;

}

**public** **double** subtrair(**double** a, **double** b) {

**this**.subtrair = a - b;

System.***out***.println("Subtração: " + **this**.subtrair);

**return** **this**.subtrair;

}

**public** **double** multiplicar(**double** a, **double** b) {

**this**.multiplicar = a \* b;

System.***out***.println("Multiplicação: " + **this**.multiplicar);

**return** **this**.multiplicar;

}

**public** **double** dividir(**double** a, **double** b) {

**double** d1 = Double.*valueOf*(a);

**double** d2 = Double.*valueOf*(b);

**this**.dividir = d1/d2;

System.***out***.println("Divisão: " + **this**.dividir);

**return** **this**.dividir;

}

}

**public** **class** TestaCalculadora {

**public** **static** **void** main(String[] args) {

Calculadora calc = **new** Calculadora();

calc.somar(5, 3);

calc.subtrair(5, 3);

calc.multiplicar(5, 3);

calc.dividir(5, 2);

}

}

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**public** **class** TesteUnitarioCalculadora **extends** Calculadora{

@Test

**public** **void** testSoma() {

System.***out***.println("Somar...");

**int** a = 5;

**int** b = 3;

Calculadora instance = **new** Calculadora();

**double** expResult = 8;

**double** result = instance.somar(a, b);

*assertEquals*(expResult, result,0);

}

@Test

**public** **void** testSubtrair() {

System.***out***.println("Subtrair...");

**int** a = 5;

**int** b = 3;

Calculadora instance = **new** Calculadora();

**double** expResult = 2;

**double** result = instance.subtrair(a, b);

*assertEquals*(expResult, result, 0);

}

@Test

**public** **void** testMultiplicar() {

System.***out***.println("Multiplicar...");

**int** a = 5;

**int** b = 3;

Calculadora instance = **new** Calculadora();

**double** expResult = 15;

**double** result = instance.multiplicar(a, b);

*assertEquals*(expResult, result, 0);

}

@Test

**public** **void** testDividir() {

System.***out***.println("Dividir...");

**int** a = 5;

**int** b = 2;

Calculadora instance = **new** Calculadora();

**double** expResult = 2.5;

**double** result = instance.dividir(a, b);

*assertEquals*(expResult, result, 0);

}

}