Lab JUnit – Semana 01

WebQuest

Introdução

Exercitar o uso de testes automatizados com o JUnit.

Tarefa

Testar classe e métodos com JUnit.

Use nomes com significado para os métodos de Teste

- Não é necessário usar a palavra "test" no nome do método. Alem de pertencer a uma classe de teste, existe já a anotação @Test logo na linha anterior!
  - Atenção: usar "test" no nome dos métodos de teste **era** necessário em **versões antigas** do JUnit. Utilize JUnit 4, e não JUnit 3 ou anterior!

- Utilize um equema padronizado: WhenThisHappensThenThatHappens

  veja exemplos de variações da mesma ideia:
  https://dzone.com/articles/7-popular-unit-test-naming

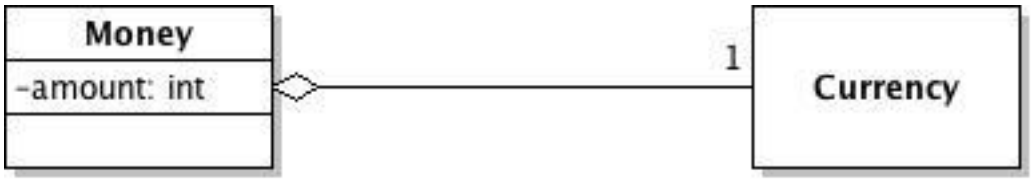
  O importante é explicar no nome qual situação está sendo testada e o que se espera, exemplos:
  whenAddingUSDExpectSumUSD
  whenAddingDifferentMoniesReturnMoneyBag

  é importante pois:
  - torna o codigo da classe teste legivel com muito menos documentação em forma de comentarios!
  - quando o JUnit mostrar os testes passando ou não passando, a propria leitura dos nomes dos métodos já informa o que deu certo ou errado!

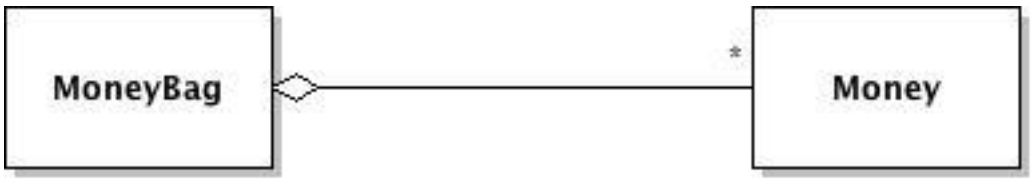
Processo

1. Criar a classe Money abaixo no seu ambiente Java, inicialização de objeto da classe Money por meio de setters públicos. Um objeto Money possui amount inteiro (para simplificar) e tipo da moeda (currency) em string de 3 letras: USD, CHF, EUR, BRL etc. Se tiver interesse, veja a lista toda em ISO 4217 Currency Codes: http://www.xe.com/iso4217.php!
2. Fazer um teste para o método add(Money money) com base no assertTrue, onde se supões que a soma ocorre apenas na mesma moeda.
3. Fazer um teste para o método add(Money money) com base no assertEquals.

4. Usar @Before e extrair o fixture num método do tipo setup(). Fazer os testes passarem de novo!
5. Dada a associação da classe Money com a classe Currency, mostrada abaixo, construir a classe Currency e fazer injeção de dependência do tipo construtor na classe Money. Substitua um string _currency na classe Money por um Currency _currency!



6. Refatore os testes de modo que funcionem novamente!
7. Crie uma classe MoneyBag que vai conter quantidades de dinheiro nos vários tipos de moedas, de acordo com a associação abaixo. Por exemplo: 10 USD com 20 CHF com 15 EUR.



8. Teste todos métodos não getters ou setters de MoneyBag no JUnit.
9. Apresente uma aplicação que mostre as várias moedas num objeto MoneyBag.
10. Mude o método add() de Money, de modo que ele volte um valor money se a soma ocorre entre moedas iguais ou volte um valor moneybag se a soma ocorre entre moedas diferentes. Estruturar as atividades necessárias para fazer isso passo-a-passo!
11. Acrescente um método em MoneyBag que volte o valor total em Reais (BRL) das várias moedas que se encontram lá. Isso vai fazer com que MoneyBag precise converter valores de moedas diferentes e, se já não estiverem em BRL, deverão converter seus valores para BRL. Estruturar as atividades necessárias para fazer isso passo-a-passo! Experimente para as seguintes moedas, estabelecendo taxas de conversão que acabem sempre em números inteiros: BRL, USD (vale 3 vez a BRL) e CHF (vale 2 vezes a BRL).

```
class Money {
      public Money() {
      }
      public Money add(Money m) {
            Money money = new Money();
            money.setAmount(this.getAmount() + m.getAmount());
            money.setCurrency(this.getCurrency());
            return money;
      }
      public int getAmount()
            { return _amount;
      }
```

```
	public void setAmount(int amount)
		{ this._amount = amount;
	}
	public String getCurrency()
		{ return _currency;
	}
	public void setCurrency(String currency)
		{ this._currency = currency;
	}
	private int _amount;
	private String _currency;
}
```