JUnit

O **JUnit** é um *framework open-source*, criado por Erich Gamma e Kent Beck, com suporte à criação de testes automatizados na linguagem de programação Java.

Esse framework facilita a criação de código para a automação de testes com apresentação dos resultados. Com ele, pode ser verificado se cada método de uma classe funciona da forma esperada, exibindo possíveis erros ou falhas podendo ser utilizado tanto para a execução de baterias de testes como para extensão.

Com JUnit, o programador tem a possibilidade de usar esta ferramenta para criar um modelo padrão de testes, muitas vezes de forma automatizada.

O teste de unidade testa o menor dos componentes de um sistema de maneira isolada. Cada uma dessas unidades define um conjunto de estímulos (chamada de métodos), e de dados de entrada e saída associados a cada estímulo. As entradas são parâmetros e as saídas são o valor de retorno, exceções ou o estado do objeto. Tipicamente um teste unitário executa um método individualmente e compara uma saída conhecida após o processamento da mesma.

A expressão acima verifica se a saída de algumMetodo() é 2 quando esse método recebe o parâmetro 1. Normalmente o desenvolvedor já realiza testes semelhantes a esse pequeno exemplo, o que é chamado de testes unitários em linha. Assim sendo, o conceito chave de um teste de unidade é exercitar um código e qual o resultado esperado.

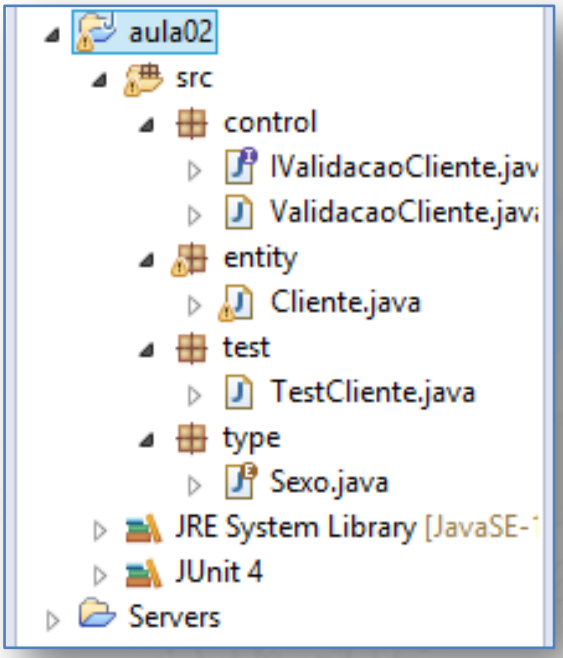
O JUnit permite a realização de **testes de unidades**, conhecidos como "caixa branca", facilitando assim a correção de métodos e objetos.

Algumas vantagens de se utilizar JUnit:

1. Permite a criação rápida de código de teste enquanto possibilita um aumento na qualidade do sistema sendo desenvolvido e testado;
2. Não é necessário escrever o próprio framework;
3. Amplamente utilizado pelos desenvolvedores da comunidade código-aberto, possuindo um grande número de exemplos;
4. Uma vez escritos, os testes são executados rapidamente sem que, para isso, seja interrompido o processo de desenvolvimento;
5. JUnit checa os resultados dos testes e fornece uma resposta imediata;
6. Pode-se criar uma hierarquia de testes que permitirá testar apenas uma parte do sistema ou todo ele;

7. Escrever testes com JUnit permite que o programador perca menos tempo depurando seu código;
8. JUnit é LIVRE.

Estrutra do projeto...




```
package entity;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;
import java.util.TreeSet;

import type.Sexo;

public class Cliente {

//Comparable (Ordenar a Lista de Objetos) ....
```

```
	private Integer idCliente;
	private String  nome;
	private String  email;
	private String  sexo;

//Cliente está Preso a uma Venda


//Construtor Vazio
	public Cliente() {
	}

//Cheio
	public Cliente(Integer idCliente, String nome, String
email, String sexo) {
		this.idCliente = idCliente;
		this.nome = nome;
		this.email = email;
		this.sexo = sexo;
	}

//ToString
	@Override
	public String toString() {
		return "Cliente [idCliente=" + idCliente + ", nome="
+ nome + ", email=" + email + ", sexo=" + sexo + "]";
	}

	public Integer getIdCliente() {
		return idCliente;
	}
	public void setIdCliente(Integer idCliente) {
		this.idCliente = idCliente;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
```

```
	}
	public String getSexo() {
		return sexo;
	}
	public void setSexo(String sexo) {
		this.sexo = sexo;
	}

	public static void main(String[] args) {

//  chave   conteudo
Map <Integer, Cliente> mapa = new HashMap<Integer,Cliente>();
		try{
	Cliente c1 = new Cliente(1,"leo","leo@gmail.com",
Sexo.MASCULINO.getSexo());
	Cliente c2 = new Cliente(3,"ben","ben@gmail.com","m");
	Cliente c3 = new Cliente(1,"pool","pool@gmail.com","m");
	Cliente c4 = new
Cliente(4,"juliana","juliana@gmail.com",Sexo.FEMININO.getSexo())
;
		mapa.put(c1.getIdCliente() , c1);
		mapa.put(c2.getIdCliente() , c2);
		mapa.put(c3.getIdCliente() , c3);

//Classe que possui métodos estáticos ..
		Collections.sort(null);
//Grava com Mapa (put)
//primeiro é chave, segundo é conteúdo ...
//Chave e Conteudo
		System.out.println("Todos :" + mapa);

//busca
		System.out.println("Busca :" + mapa.get(4));
//Busca com a Chave ...
	System.out.println("Varrendo o Lado Esquerdo do Mapa");
		for(Integer chave : mapa.keySet()){
			System.out.println(chave); //o id
		}

//Varrendo o Lado direito do Map
		for(Cliente c : mapa.values()){
			System.out.println(c);
// O Objeto do Cliente....
		}
```

```
//Collection -> List, Set e Queue
//List
	Collection<Cliente> dc = new ArrayList<Cliente>();
//Queue
	Collection<Cliente>    dc2 = new LinkedList<Cliente>();
//Set
	Collection <Cliente> dc3= new TreeSet<Cliente>();
//Mapa não é uma Collection ...
//Collections(Classe com Métodos Estáticos) <>
//Collection (Interface SuperInterface)
		}catch(Exception ex){
			ex.printStackTrace();
		}
	}
}
```

No console...

```
Todos :{1=Cliente [idCliente=1, nome=pool, email=pool@gmail.com,
sexo=m], 3=Cliente [idCliente=3, nome=ben, email=ben@gmail.com,
sexo=m]}
Busca :null
Varrendo o Lado Esquerdo do Mapa
1
3
Cliente [idCliente=1, nome=pool, email=pool@gmail.com, sexo=m]
Cliente [idCliente=3, nome=ben, email=ben@gmail.com, sexo=m]
```

```
package type;

public enum Sexo {
	MASCULINO("m"), FEMININO("f");

	private String sexo;

//construtor é do tipo do Enum, não tem
	Sexo(String sexo) {
		this.sexo = sexo;
	}
	public String getSexo() {
		return sexo;
```

```
	}
}
```

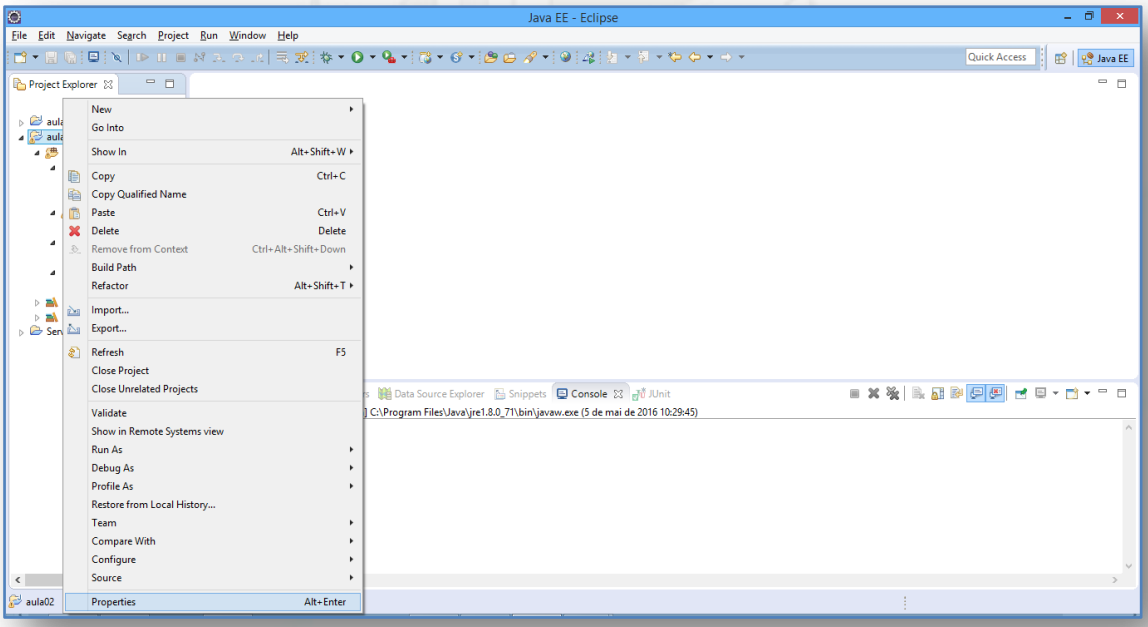
```
package control;

import entity.Cliente;

public interface IValidacaoCliente {

	Boolean validaId(Cliente c);
	Boolean validaIdIsNull(Cliente c);
	Boolean validaNome(Cliente c);
	Boolean validaEmail(Cliente c);
}
```

```
package control;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import entity.Cliente;

public class ValidacaoCliente
implements IValidacaoCliente {

                    //VO (Value Object)
	public Boolean validaId(Cliente c){
		if (c.getIdCliente()<=0){
			return false;
		}
		return true;
	}

//Programar para não travar no error
	public Boolean validaIdIsNull(Cliente c){
		if (c.getIdCliente() == null){
		 return true;
		}
		return false;
	}

	public Boolean validaNome(Cliente c){
```

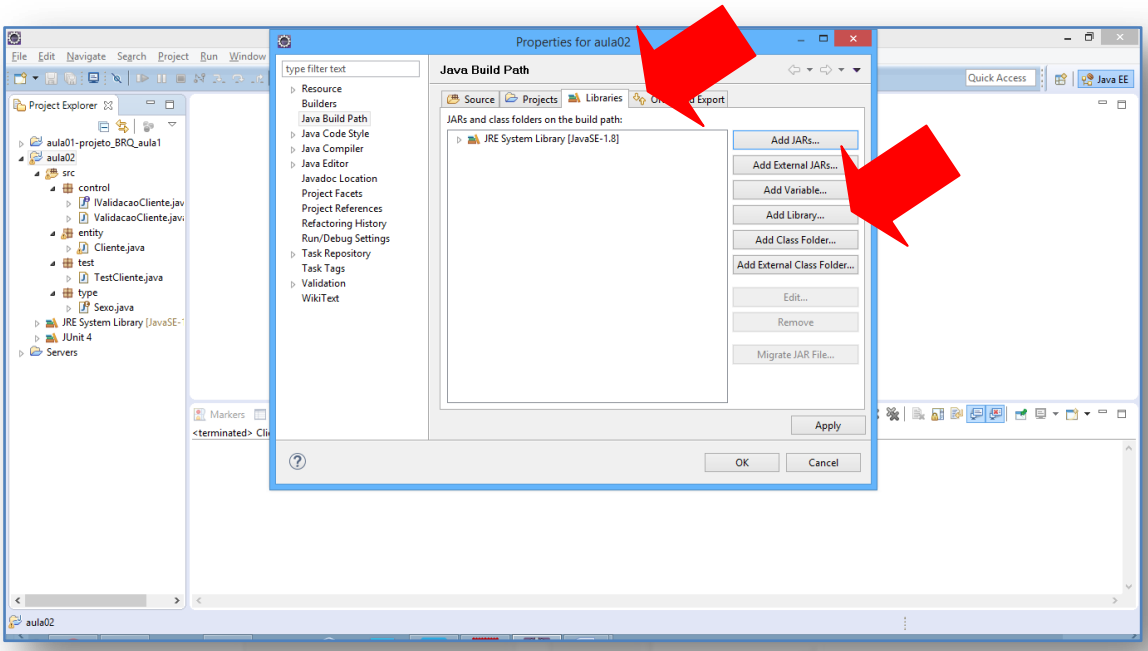
```
            Pattern p = Pattern.compile("[A-Z a-z]{2,35}");
//validando letra minuscula,maiuscula, espaço
            Matcher m = p.matcher(c.getNome());
            return m.matches(); //true or false
      }

      public Boolean validaEmail(Cliente c){
            Pattern p = Pattern.compile(".+@.+\\.[a-z]+");
//validando letra minuscula,maiuscula, espaço
//.(a-z0-9_-) + quantidade
//@ tenho que colocar o @
//\\. (SOU OBRIGADO A PRESSIONAR .)
//[a-z]+ //vou repetir
            Matcher m = p.matcher(c.getEmail());
            return m.matches(); //true or false
      }
}
```

Para inserir o JUnit4..

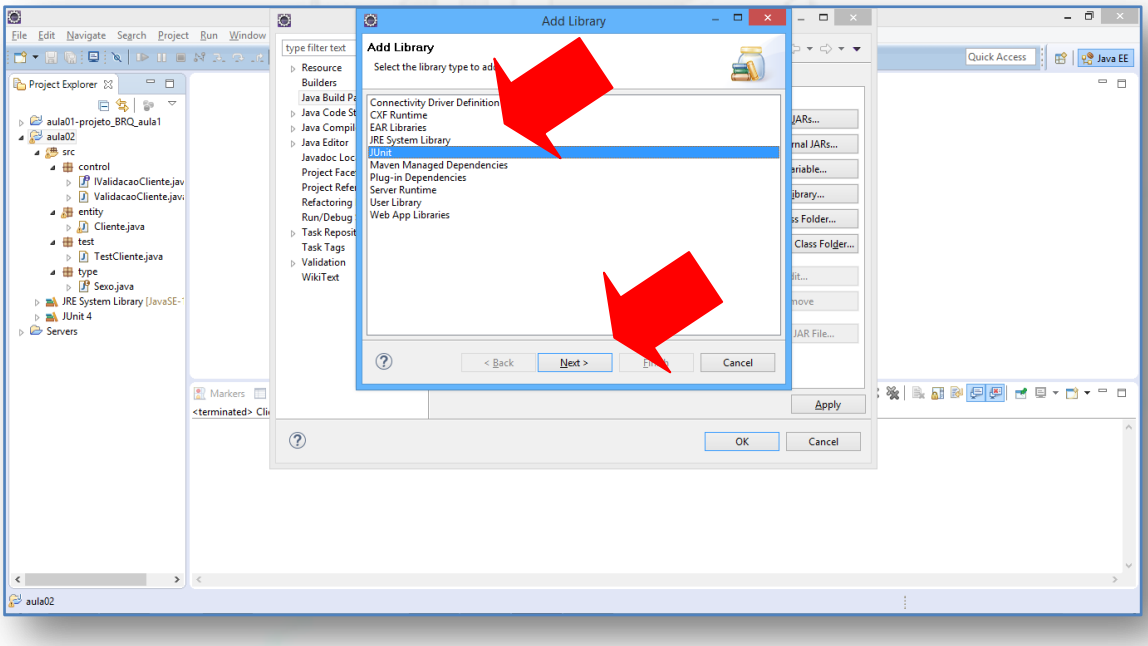
Clicar no projeto com botão direito -> Properties...



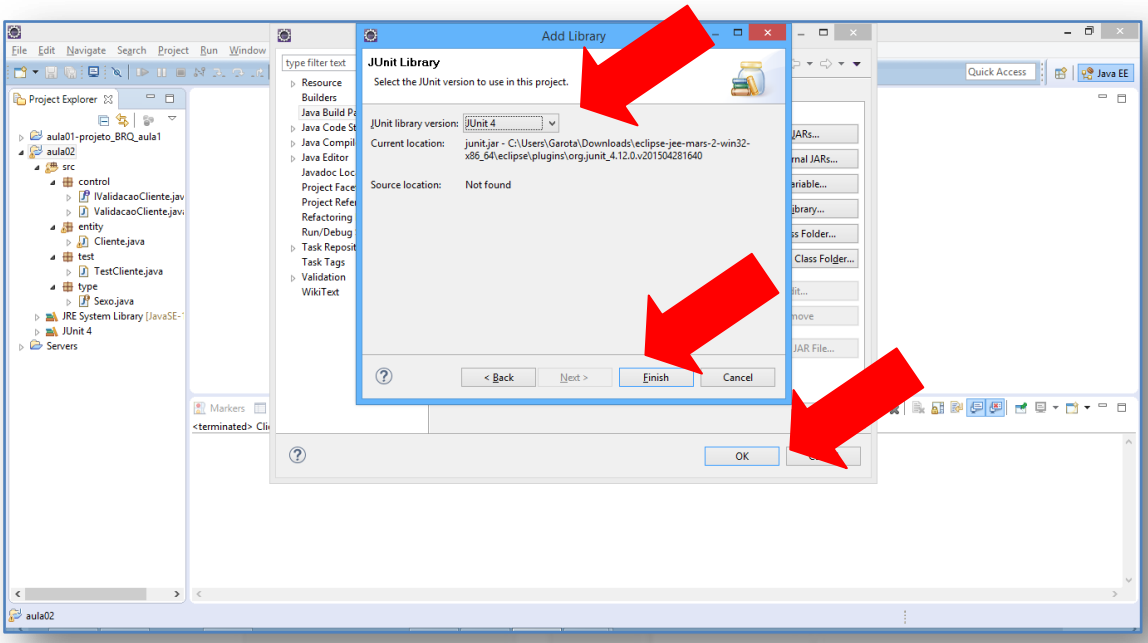
Clicar na aba Libraries -> Add Library...



Clicar em Junit -> Next...



JUnit4 -> Finish-> OK

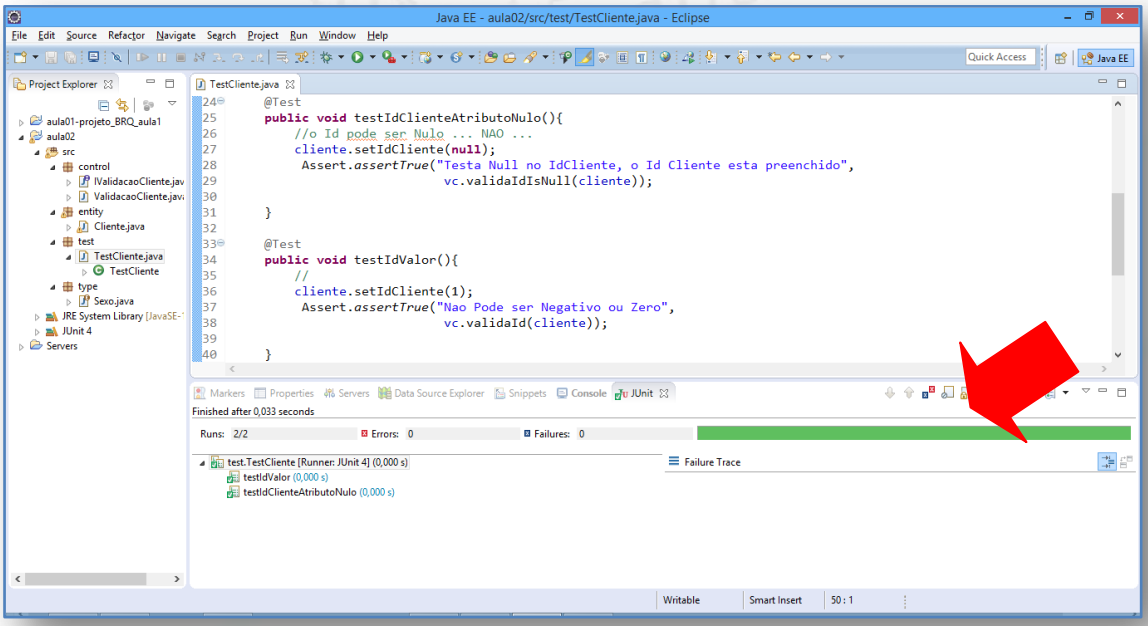



```
package test;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

import control.IValidacaoCliente;
import control.ValidacaoCliente;
import entity.Cliente;

public class TestCliente {

        private Cliente cliente; //entidade..
        IValidacaoCliente vc; //validação..

        @Before //executa antes dos testes
        public void init() {
           cliente = new Cliente();
           vc = new ValidacaoCliente();
        }

        @Test
        public void testIdClienteAtributoNulo(){
//o Id pode ser Nulo ... NAO ...
```

```
            cliente.setIdCliente(null);
           Assert.assertTrue("Testa Null no IdCliente, o Id
Cliente esta preenchido", vc.validaIdIsNull(cliente));

        }

        @Test
        public void testIdValor(){
             cliente.setIdCliente(1);
            Assert.assertTrue("Nao Pode ser Negativo ou Zero",
                              vc.validaId(cliente));
        }
      public Cliente getCliente() {
            return cliente;
      }
      public void setCliente(Cliente cliente) {
            this.cliente = cliente;
      }
}
```

Para testar a classe, rodar normalmente...

Testei cliente para nulo e "id" passei 1, tudo ok. Dá verde!



Passei valores errados. Passei "id" negativo (-1). Dá vermelho!
E imprime a mensagem de erro.