



Práticas de desenvolvimento aplicadas na automação de testes com Selenium

Robson Bittencourt
 **#gutsrs /@gutsrs**

Programação

- 19h15 às 19h45 Recepção, boas vindas e Coffee para integração
- 19h45 às 19h55 Abertura do evento, apresentação do GUTS-RS e expectativas do evento
- 19h55 às 20h45 Práticas de desenvolvimento aplicadas na automação de testes com Selenium (Robson Bittencourt)
- 20h45 às 21h15 Hands on com Selenium

Sobre o GUTS-RS

- **GUTS-RS:** Grupo de Usuários de Testes de Software do RS
- **Criado em:** agosto/2008
- **Objetivo:** compartilhar o uso de métodos, processos e ferramentas de Teste de Software e promover discussões sobre a aplicação das melhores práticas de teste e qualidade utilizadas no mercado
- **Público Alvo:** Gerentes, Analistas de Testes, Testadores, Desenvolvedores e demais profissionais e estudantes interessados na área
- **Coordenação:** Diraci Júnior, Eduardo Oliveira e Moisés Ramírez

Canais de Comunicação



<http://guts-rs.blogspot.com.br/>



Grupo de Usuários de Testes de Software do RS



@gutsrs



Guts RS



GUTS-RS



<http://guts-rs.eventbrite.com/>



<http://pt.slideshare.net/GUTS-RS>



guts-rs-sucesu@googlegroups.com

Comunicados

- **Submissão de Palestras 2016**
 - DOJO
 - Fishbowl
 - Palestra
 - TCC
 - Testing Games
 - Workshop
 - Outros
- **Assinar a lista de presença**
- **Preencher a Ficha do Evento**
- **Certificado de Participação**

Próximos Eventos

- GUTS Talks (julho)



Ferramentas de Automação de Testes

Sobre o palestrante

Robson é graduado em Sistemas de Informação, pela Facensa de Gravataí. Busca estar sempre a par das boas práticas de construção de software, como testes, Clean Code e Design Patterns. Gosta de estudar coisas novas, principalmente assuntos relacionados a Engenharia de Software e Métodos Ágeis. Gosta de repassar as coisas que aprende e tem feito isso através de palestras e do seu [blog](#).

Contatos



robson.luizv@gmail.com



[@r Luizv](https://twitter.com/r Luizv)



github.com/robsonbittencourt



rbittencourt.com



CWI SOFTWARE





Práticas de desenvolvimento aplicadas na automação de testes com Selenium

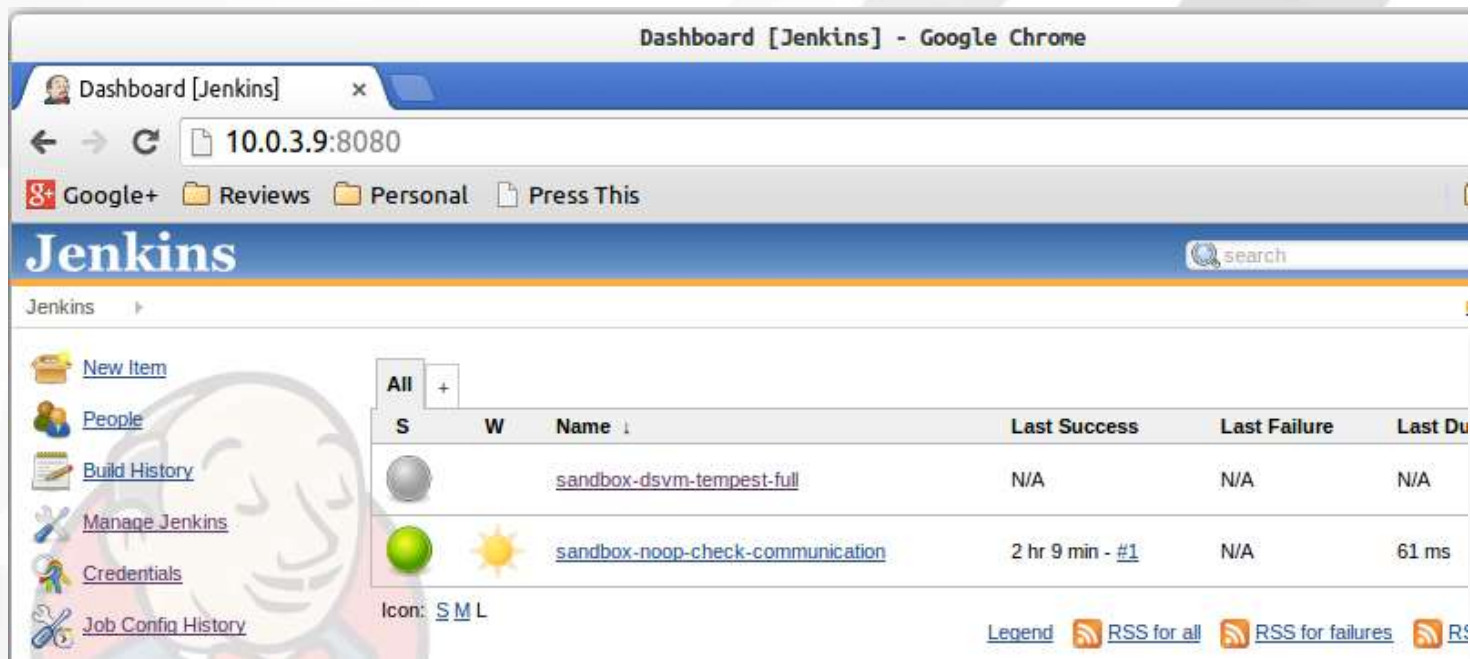
- Importância dos testes automatizados
- Linguagens Orientadas a Objetos
- Abstrações
- SOLID
- DRY
- Design Patterns
- Clean Code
- Organização
- Indo além

Importância dos testes automatizados



Importância dos testes automatizados

Feedback contínuo



The screenshot shows the Jenkins Dashboard in a Google Chrome browser window. The browser's address bar displays '10.0.3.9:8080'. The Jenkins interface includes a sidebar with navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', and 'Job Config History'. The main content area features a table of build jobs. The table has columns for 'S' (Status), 'W' (Workspace), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. Two jobs are listed: 'sandbox-dsvm-tempest-full' and 'sandbox-noop-check-communication'. The first job is in a grey 'Failed' state, and the second is in a green 'Success' state with a sun icon. At the bottom right, there are links for 'Legend', 'RSS for all', 'RSS for failures', and 'RSS for successes'.

S	W	Name	Last Success	Last Failure	Last Duration
Failed		sandbox-dsvm-tempest-full	N/A	N/A	N/A
Success	Sun	sandbox-noop-check-communication	2 hr 9 min - #1	N/A	61 ms

Importância dos testes automatizados

Nos dão segurança



Importância dos testes automatizados

Liberam tempo para testes e tarefas mais complexas



Importância dos testes automatizados

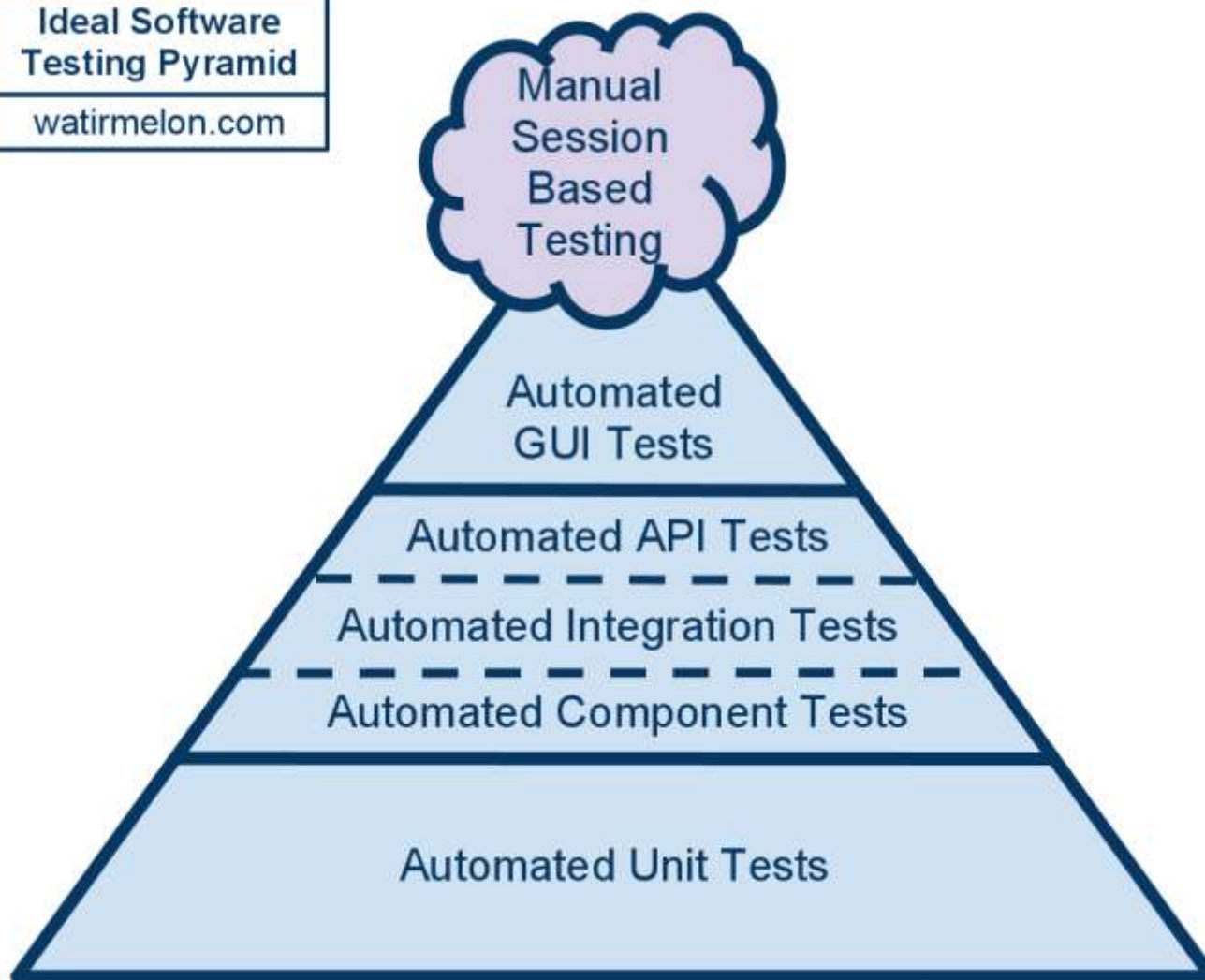
Uma vez escritos se tornam guardiões



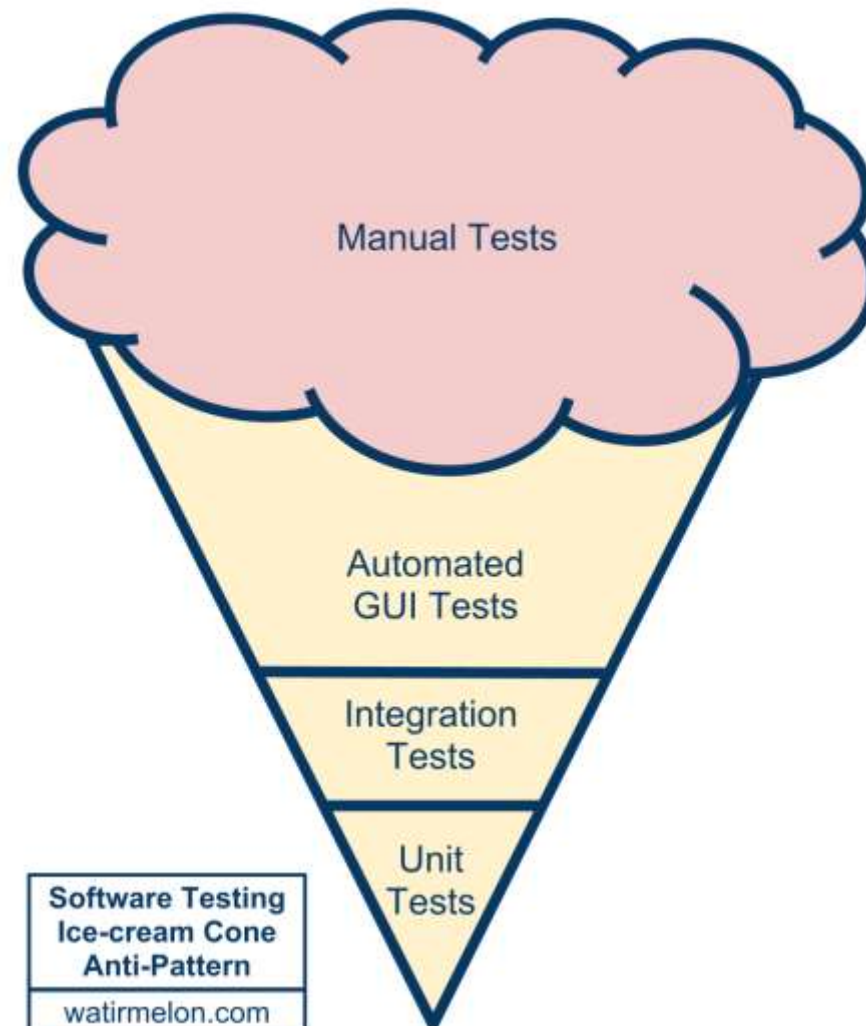
Importância dos testes automatizados

Ideal Software
Testing Pyramid

watirmelon.com



Importância dos testes automatizados



Importância dos testes automatizados

Mas porquê?

Importância dos testes automatizados

Escrever testes funcionais é complexo

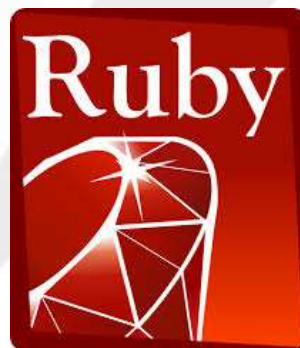


Importância dos testes automatizados

```
@Test
public void validUserCredential() {
    System.out.println("Starting test " + new Object(){}.getClass().getEnclosingMethod().getName());
    driver.get("http://www.store.demoqa.com");
    driver.findElement(By.xpath(".*[@id='account']/a")).click();
    driver.findElement(By.id("log")).sendKeys("testuser_3");
    driver.findElement(By.id("pwd")).sendKeys("Test@123");
    driver.findElement(By.id("login")).click();
    try {
        element = driver.findElement(By.xpath(".*[@id='account_logout']/a"));
    } catch (Exception e) {

    }
    Assert.assertNotNull(element);
    System.out.println("Endind test " + new Object(){}.getClass().getEnclosingMethod().getName());
}
```

Linguagens Orientadas a Objetos

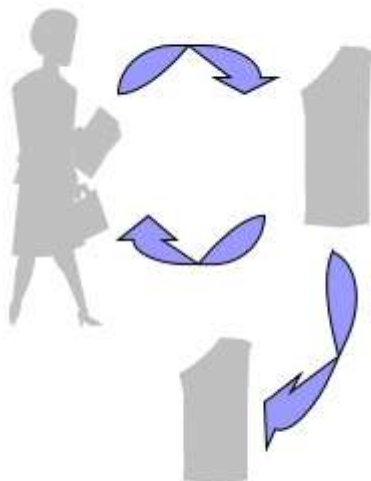


python

Linguagens Orientadas a Objetos

Diferença entre o paradigma procedural e a orientação a objetos

■ Procedural



Withdraw, deposit, transfer

■ Object Oriented



Customer, money, account

ABSTRAÇÕES

Carro



Metáfora do lenhador



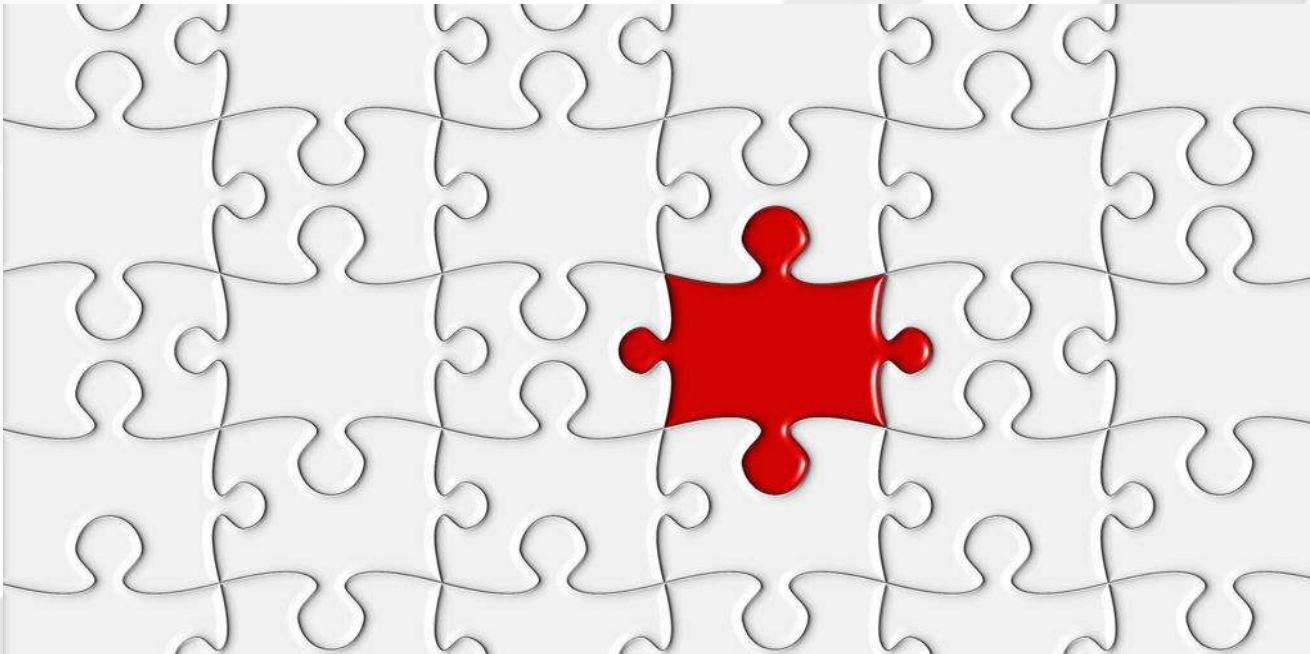
Criar uma caixa de ferramentas



Projeto de classes



Quebra-cabeças





SOLID

Software development is not a Jenga game.

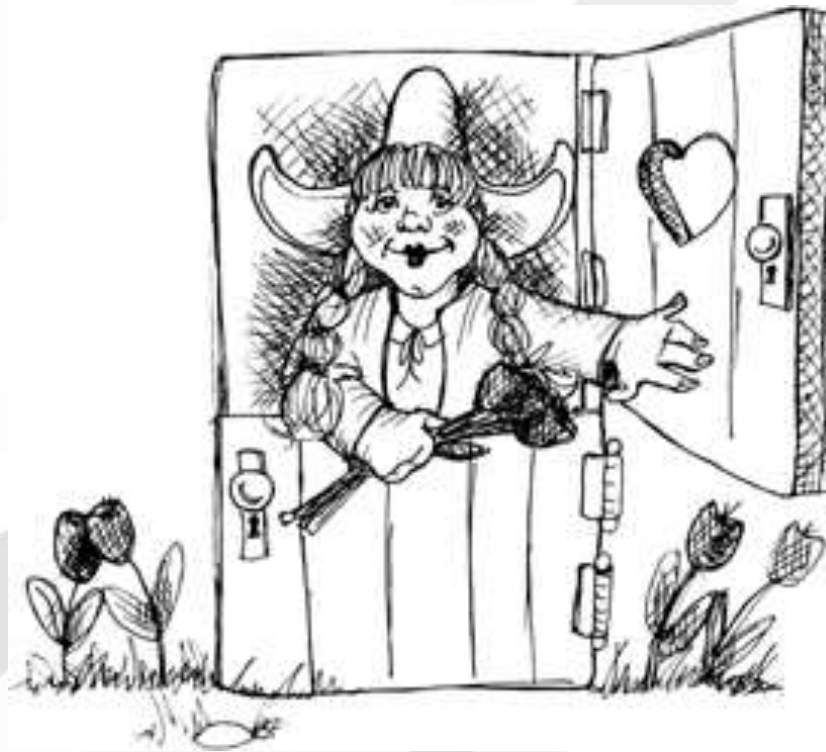
S - Single responsibility
O - Open / closed
L - Liskov substitution
I - Interface segregation
D - Dependency inversion



Single responsibility



Open / Closed principle



Open / Closed principle

```
public class CheckoutValueCalculator {  
  
    public double getCheckoutValue(List<NormalItem> itens) {  
        double total = 0;  
  
        for (NormalItem item : itens) {  
            total += item.getCost() + item.getProfit();  
        }  
  
        return total;  
    }  
}
```

Open / Closed principle

```
public class CheckoutValueCalculator {  
  
    double total = 0;  
  
    for (Object item : itens) {  
        if (item.getClass().isInstance(NormalItem.class)) {  
            total += item.getCost() + item.getProfit();  
        }  
  
        if (item.getClass().isInstance(EspecialItem.class)) {  
            total += item.getCost() + item.getProfit() - item.getDiscount();  
        }  
    }  
  
    return total;  
}
```

Open / Closed principle

```
public interface Item {  
    double getFinalPrice();  
}  
  
public class NormalItem implements Item {  
    @Override  
    public double getFinalPrice() {  
        return getCost() + getProfit();  
    }  
}  
  
public class EspecialItem implements Item {  
    @Override  
    public double getFinalPrice() {  
        return getCost() + getProfit() - getDiscount();  
    }  
}
```

Open / Closed principle

```
public class CheckoutValueCalculator {  
    public double getCheckoutValue(List<Item> itens) {  
        double total = 0;  
  
        for (Item item : itens) {  
            total += item.getFinalPrice();  
        }  
  
        return total;  
    }  
}
```


Don't repeat yourself

Não se repita

```
public UserList createNormalClient() {  
    NewUserPage newUserPage = new NewUserPage();  
    newUserPage.userName.set(getRandomString());  
    newUserPage.password.set(getRandomString());  
    newUserPage.repeatPassword.set(getRandomString());  
  
    // code to config normal client  
}  
  
public UserList createVipClient() {  
    NewUserPage newUserPage = new NewUserPage();  
    newUserPage.userName.set(getRandomString());  
    newUserPage.password.set(getRandomString());  
    newUserPage.repeatPassword.set(getRandomString());  
  
    // code to config vip client  
}
```

```
public UserList createNormalClient() {
    NewUserPage newUserPage = new NewUserPage();
    fillCredentials(newUserPage);

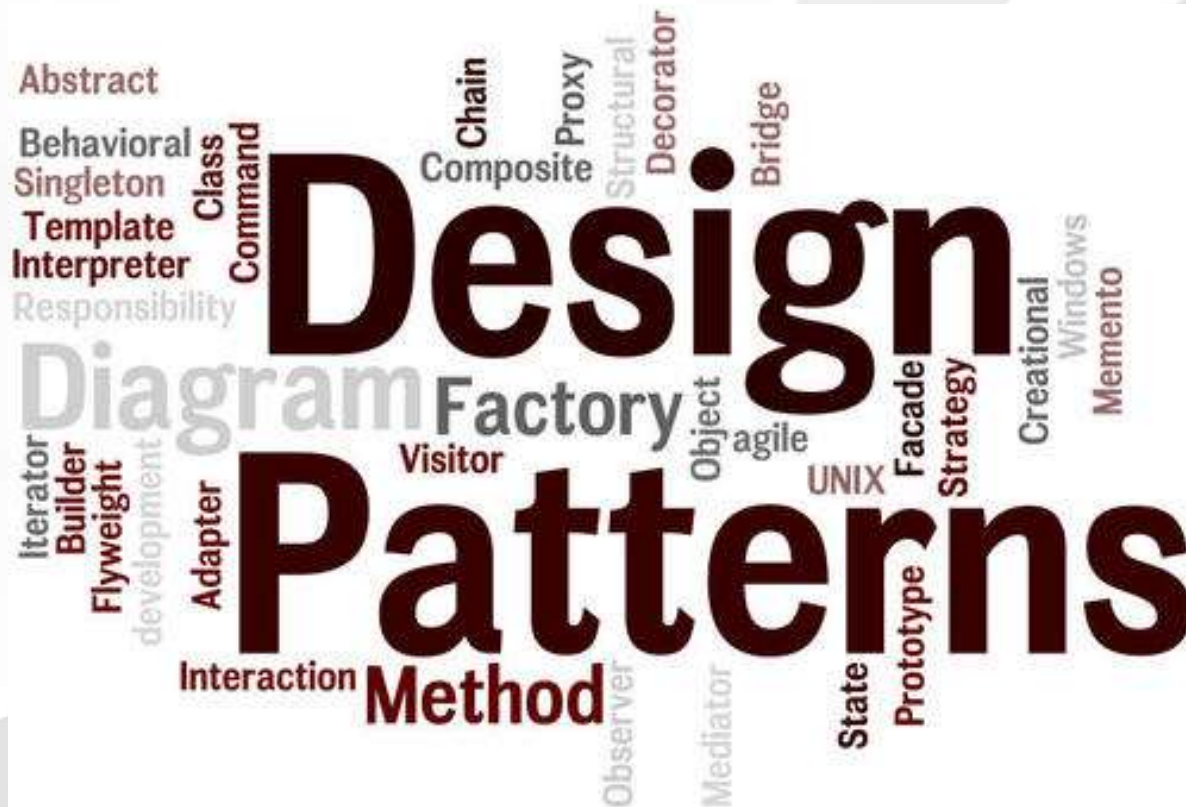
    // code to config normal client
}

public UserList createVipClient() {
    NewUserPage newUserPage = new NewUserPage();
    fillCredentials(newUserPage);

    // code to config vip client
}

private void fillCredentials(NewUserPage newUserPage) {
    newUserPage.userName.set(getRandomString());
    newUserPage.password.set(getRandomString());
    newUserPage.repeatPassword.set(getRandomString());
}
```

Design Patterns



Design Patterns

- Solução reproduzível de um problema
- Problemas clássicos geralmente possuem patterns
- Não trazem a solução final
- Instigam as boas práticas e princípios da orientação a objetos

Design Patterns

Page Object

Representa uma página

Reusáveis

Redução da manutenção

Linguagem de domínio



Page Object

```
public class StoreLoginPage {  
  
    public void login(String userName, String password) {  
        driver.findElement(By.xpath(".*[@id='account']/a")).click();  
        driver.findElement(By.id("log")).sendKeys(userName);  
        driver.findElement(By.id("pwd")).sendKeys(password);  
        driver.findElement(By.id("login")).click();  
    }  
  
    public void logout() {  
        driver.findElement(By.xpath(".*[@id='account_logout']/a")).click();  
    }  
  
}
```


Page Object

```
@Test
public void validUserCredential() {
    StoreLoginPage storeLoginPage = new StoreLoginPage(driver);
    storeLoginPage.goTo();

    storeLoginPage.login("testuser_3", "Test@123");
    storeLoginPage.logout();

    assertTrue(storeLoginPage.isLoginPage());
}
```

Abstrações de elementos

Representam um elemento ou componente

Encapsulam código do WebDriver



Abstrações de elementos

```
public class Button {  
  
    private WebElement element;  
  
    public Button(WebDriver driver, String selector) {  
        element = driver.findElement(By.id(selector));  
    }  
  
    public void click() {  
        element.click();  
    }  
  
    public String getText() {  
        return element.getText();  
    }  
  
}
```

Abstrações de elementos

```
public class TextField {  
  
    private WebElement element;  
  
    public TextField(WebDriver driver, String selector) {  
        element = driver.findElement(By.id(selector));  
    }  
  
    public void set(String text) {  
        element.sendKeys(text);  
    }  
  
    public String get() {  
        return element.getText();  
    }  
  
    public void clear() {  
        element.clear();  
    }  
  
}
```

Abstrações de elementos

```
public class StoreLoginPage {  
  
    private WebDriver driver;  
  
    public void login(String userName, String password) {  
        driver.findElement(By.xpath(".*[@id='account']/a")).click();  
        driver.findElement(By.id("log")).sendKeys("testuser_3");  
        driver.findElement(By.id("pwd")).sendKeys("Test@123");  
        driver.findElement(By.id("login")).click();  
    }  
  
    public void logout() {  
        driver.findElement(By.xpath(".*[@id='account_']/a")).click();  
    }  
}
```

Abstrações de elementos

```
public class StoreLoginPage {  
  
    private WebDriver driver;  
  
    private Button accountButton = new Button(driver, "account");  
    private Button loginButton = new Button(driver, "login");  
    private Button logoutButton = new Button(driver, "account_logout");  
    private TextField userNameField = new TextField(driver, "log");  
    private TextField passwordField = new TextField(driver, "pwd");  
  
    public void login(String userName, String password) {  
        accountButton.click();  
  
        userNameField.set(userName);  
        passwordField.set(password);  
  
        loginButton.click();  
    }  
  
    public void logout() {  
        logoutButton.click();  
    }  
}
```

Abstrações de elementos

```
public void login(String userName, String password) {  
    driver.findElement(By.xpath(".*[@id='account']/a")).click();  
    driver.findElement(By.id("log")).sendKeys("testuser_3");  
    driver.findElement(By.id("pwd")).sendKeys("Test@123");  
    driver.findElement(By.id("login")).click();  
}
```

```
public void login(String userName, String password) {  
    accountButton.click();  
  
    userNameField.set(userName);  
    passwordField.set(password);  
  
    loginButton.click();  
}
```


Design Patterns

Fixtures / Builders

Auxiliam na criação do setup de teste

Reusáveis

Foco no caso de teste



Fixtures / Builders

```
public class UserFixture {  
  
    public UserLoginPage createUser(String userName, String password) {  
        HomePage homePage = new HomePage();  
        homePage.goTo();  
        homePage.createUser();  
  
        NewLoginPage newUserPage = new NewLoginPage();  
        newUserPage.fillInformation(userName, password);  
        newUserPage.save();  
  
        return new UserLoginPage();  
    }  
}
```

Fixtures / Builders

```
@Test
public void shouldDeleteAnUser() {
    String userName = getRandomString();
    String password = getRandomString();

    HomePage homePage = new HomePage();
    homePage.goTo();
    homePage.createUser();

    NewUserPage newUserPage = new NewUserPage();
    newUserPage.fillInformation(userName, password);
    newUserPage.save();
    UserListPage userListPage = new UserListPage();

    UserPage userPage = userListPage.search(userName);
    userListPage = userPage.delete();

    assertFalse(userListPage.isExistingUser(userName));
}
```

Fixtures / Builders

```
@Test
public void shouldDeleteAnUser() {
    String userName = getRandomString();
    String password = getRandomString();

    UserListPage userListPage = new UserFixture().createUser(userName, password);

    UserPage userPage = userListPage.search(userName);
    userListPage = userPage.delete();

    assertFalse(userListPage.isExistingUser(userName));
}
```

Design Patterns

Fixtures / Builders

```
HomePage homePage = new HomePage();  
homePage.goTo();  
homePage.createUser();  
  
NewUserPage newUserPage = new NewUserPage();  
newUserPage.fillInformation(userName, password);  
newUserPage.save();  
UserListPage userListPage = new UserListPage();
```

```
UserListPage userListPage = new UserFixture().createUser(userName, password);
```

Clean Code



Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

Martin Fowler, 2008.

Bons nomes



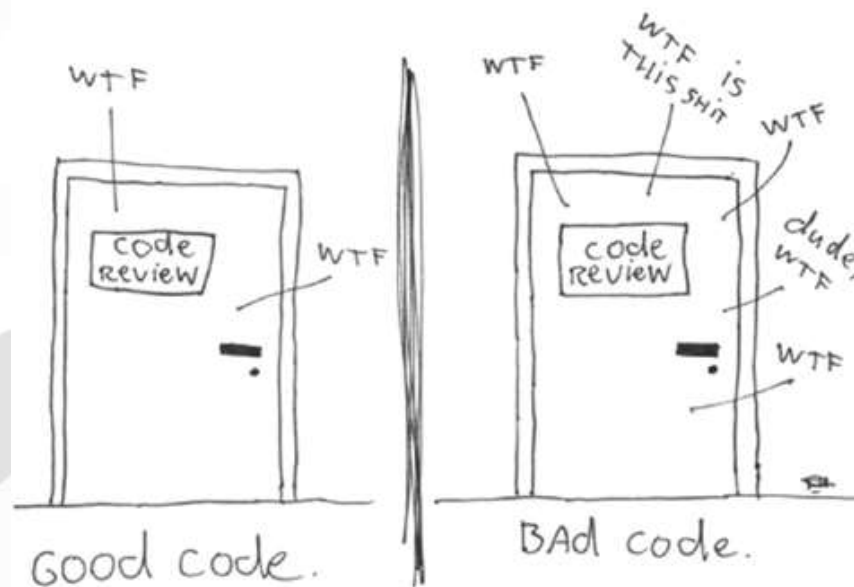
Bons nomes

```
@Test
public void deleteUser() {
    String usrNm = getRandomString();
    String pwd = getRandomString();
    UserListPage userListPage = new UserFixture().create(usrNm, pwd);
    //test something
}
```

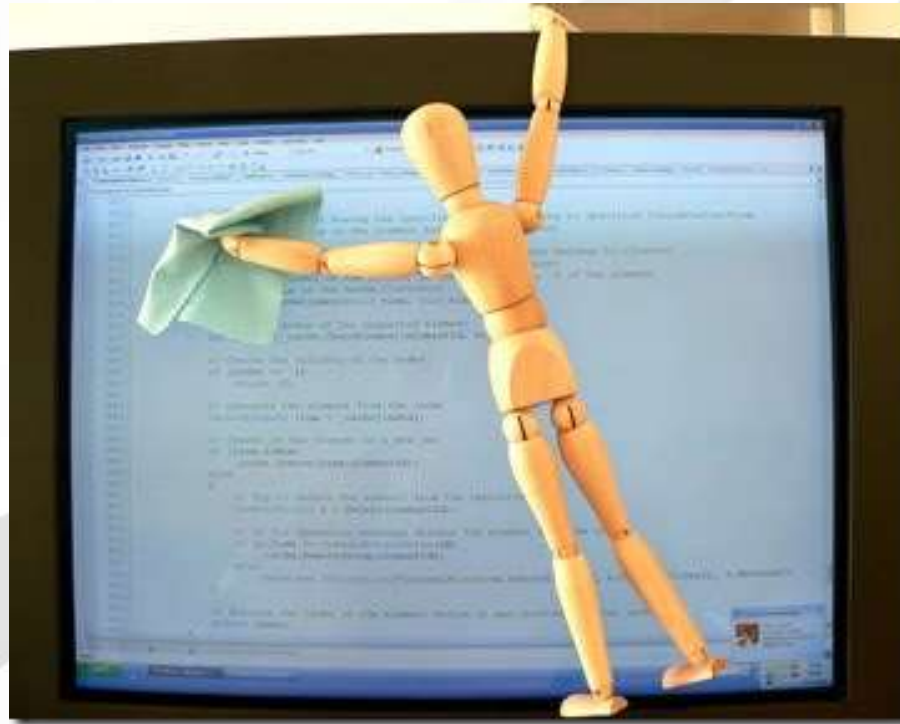
```
@Test
public void shouldDeleteAnUser() {
    String userName = getRandomString();
    String password = getRandomString();
    UserListPage userListPage = new UserFixture().create(userName, password);
    //test something
}
```

Métodos Pequenos

The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



Comentários



Organização



Fluxo de Páginas

```
public class ItensCartTest {  
  
    @Test  
    public void shouldAddItensOnCart() {  
        LoginPage loginPage = new LoginPage();  
  
        HomePage homePage = loginPage.login();  
  
        ItemPage itemPage = homePage.selectItem();  
  
        CartPage cartPage = itemPage.addToCart();  
  
        assertEquals(1, cartPage.getItensOnCart());  
    }  
}
```

Testes independentes

A classe de Teste deve criar seus próprios dados

Testes independentes

Uma classe de teste por funcionalidade
testada

Testes independentes

Dependências dentro da mesma classe são
aceitáveis

given-when-then

```
public class DeleUserTest {  
  
    @Test  
    public void shouldDeleteAnUser() {  
        String userName = getRandomString();  
        String password = getRandomString();  
        UserListPage userListPage = new UserFixture().create(userName, password);  
  
        UserPage userPage = userListPage.search(userName);  
        userListPage = userPage.delete();  
  
        assertFalse(userListPage.isExistingUser(userName));  
    }  
}
```



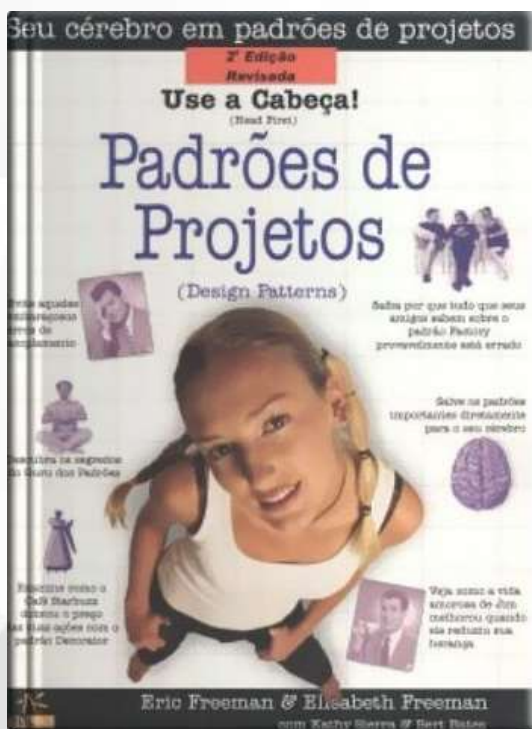
"That's all Folks!"



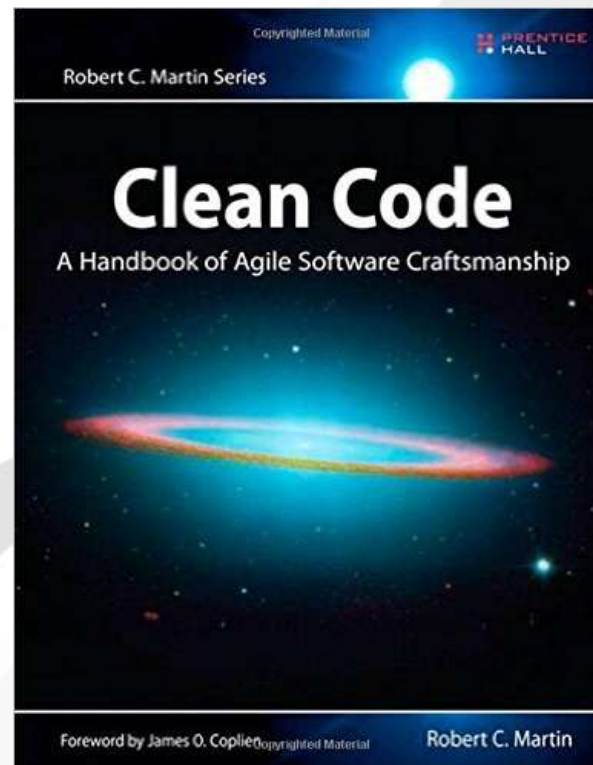
Livros



Livros



Livros



Mokona



github.com/robsonbittencourt/mokona

ABSTRAÇÕES



Dúvidas?

robson.luizv@gmail.com

