

Programação Orientada a Objetos

Introdução ao Testes de Software com JUnit

Prof. Dirson Santos de Campos
dirson_campos@ufg.br

04/12/2023

Programação Orientada a Objetos

- Introdução ao Teste de Software
- Ferramentas para Teste de Software, exemplo, em Java, JUNIT

Programação Orientada a Objetos

- **Processo de Software**
- Um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software.

Programação Orientada a Objetos

- **Processo de Software**
 - **Modelo Básico de um Processo de Software:**
 - Planejamento
 - Análise e Especificação de Requisitos
 - Projeto
 - Implementação
 - **Testes**
 - Entrega e Implantação
 - Operação
 - Manutenção

Programação Orientada a Objetos

- **Processo de Software**

- Fatores que influenciam na definição do Processo:
 - Tipo do software
 - Paradigma (Orientado a Objetos)
 - Domínio da Aplicação
 - Tamanho
 - Complexidade
 - Características da Equipe

Programação Orientada a Objetos

- **Metodologia Ágeis de Desenvolvimento de Software**
- Origem histórica em 2001, um grupo de 17 renomados desenvolvedores de software, agruparam e aprimoraram os diversos conceitos de metodologias ágeis existentes e assinaram o “**Manifesto para o Desenvolvimento Ágil de Software**”, que passou então a ser muito difundido pelas comunidades de desenvolvimento.

Programação Orientada a Objetos

- **Metodologia Ágeis de Desenvolvimento de Software**



Programação Orientada a Objetos

- **Metodologia Ágeis de Desenvolvimento de Software.**
Os valores determinam que se deve priorizar:
 - Os indivíduos e as interações entre eles mais que os processos e as ferramentas;
 - O software funcionando mais do que uma documentação completa e abrangente;
 - A colaboração com e dos clientes mais do que as negociações de contratos e;
 - Respostas a mudanças mais do que seguir o plano inicial.

Importante: Os teste de software são aplicados a metodologia ágeis são os Testes ágeis.

Programação Orientada a Objetos

- Metodologia Ágeis de Desenvolvimento de Software
- Foco em agilidade de equipes e qualidade de projetos, apoiada em valores como simplicidade, comunicação, *feedback* e coragem.
- Máxima integração entre pessoas e, principalmente, estimulando uma participação maior do cliente

Programação Orientada a Objetos

- Exemplo de Metodologia Ágil:
XP – Extreme Programming
- Sugere um conjunto de boas práticas que melhoram o planejamento, execução, e gerenciamento do projeto de software.
- Melhoram a eficiência, diminuindo o retrabalho, garantindo dessa forma a qualidade do seu projeto.
- Baseada em testes ágeis que simplificadamente nada mais são do que uma prática de teste de software que segue as regras do manifesto ágil, tratando o desenvolvimento de software como o cliente de testes.

Programação Orientada a Objetos

- XP – Extreme Programming
- **Algumas boas práticas:**
 - Organizacionais : Planejamento, Pequenas versões, testes de aceitação, envolvimento do cliente;
 - Equipes: Padronização de código, Propriedade coletiva, Integração contínua, Ritmo sustentável;
 - Pares: Programação em pares, Design simples, **Testes Unitários**, Refatoração;

Programação Orientada a Objetos

- XP – Extreme Programming
- O objetivo principal do XP é levar ao extremo esse conjunto de práticas que são ditas como boas na engenharia de software.
- **Entre elas podemos citar o teste**, visto que procurar defeitos é perda de tempo, nós temos que constantemente testar, se perder tempo com código sujo é ruim, melhoraremos o nosso código sempre que uma nova mudança for feita.
- Portanto, como podemos notar todas as coisas práticas do XP são levadas ao extremo.

Programação Orientada a Objetos

- **Teste de Software**

- Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado.

Programação Orientada a Objetos

Testar \neq Depurar

- **Principal diferença**
- **Depurar** - o que se faz quando se sabe que o programa não funciona;
- **Teste** - tentativas sistemáticas de encontrar erros em programa que você “acha” que está funcionando.

“Testes podem mostrar a presença de erros, não a sua ausência (Dijkstra)”

Programação Orientada a Objetos

• Teste de Software



Tipos de teste (caixa-branca e caixa-preta)

- **“White box” (Teste caixa-branca)**
 - Possui acesso ao código fonte, conhecendo a estrutura interna do produto. Sendo analisados e possibilitando que sejam escolhidas partes específicas de um componente para ser avaliados, permitindo uma busca precisa do comportamento da estrutura.
 - Os níveis de teste caixa branca são os **Testes de Unidade e o Teste Estático.**
 - **Exemplo de Ferramenta para teste caixa-branca (JUnit)**

Tipos de teste (caixa-branca e caixa-preta)

- **“Black box” (Teste caixa-preta)**
 - Baseia-se nos requisitos básicos do software, sendo o foco nos requisitos da aplicação, ou seja, nas ações que deve desempenhar.
 - Os níveis de teste caixa preta são Integração, Sistema, Aceitação, Alfa e Beta. Possuem métodos e classes, comandos de repetição e condições.
 - Se resumem em **testes de entrada e saída**.
 - **Exemplo de Ferramenta para teste preta (ferramenta Online Judge – Sharif-Judge)**

Programação Orientada a Objetos

- Teste de Software

- Tipos de Teste mais utilizados no desenvolvimento de software
 - Testes de Unidade
 - Testes de Integração
 - Testes de Sistema
 - Testes de Aceitação

Programação Orientada a Objetos

- Teste de Software

- Testes de Unidade

- Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente.

- Testes de Integração

- visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.

Programação Orientada a Objetos

- Teste de Software
- Testes de Sistema
 - avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Verifica se o software satisfaz aos requisitos
- Testes de Aceitação
 - são realizados geralmente por um restrito grupo de usuários finais do sistema para verificar se o comportamento está de acordo com o que foi solicitado.

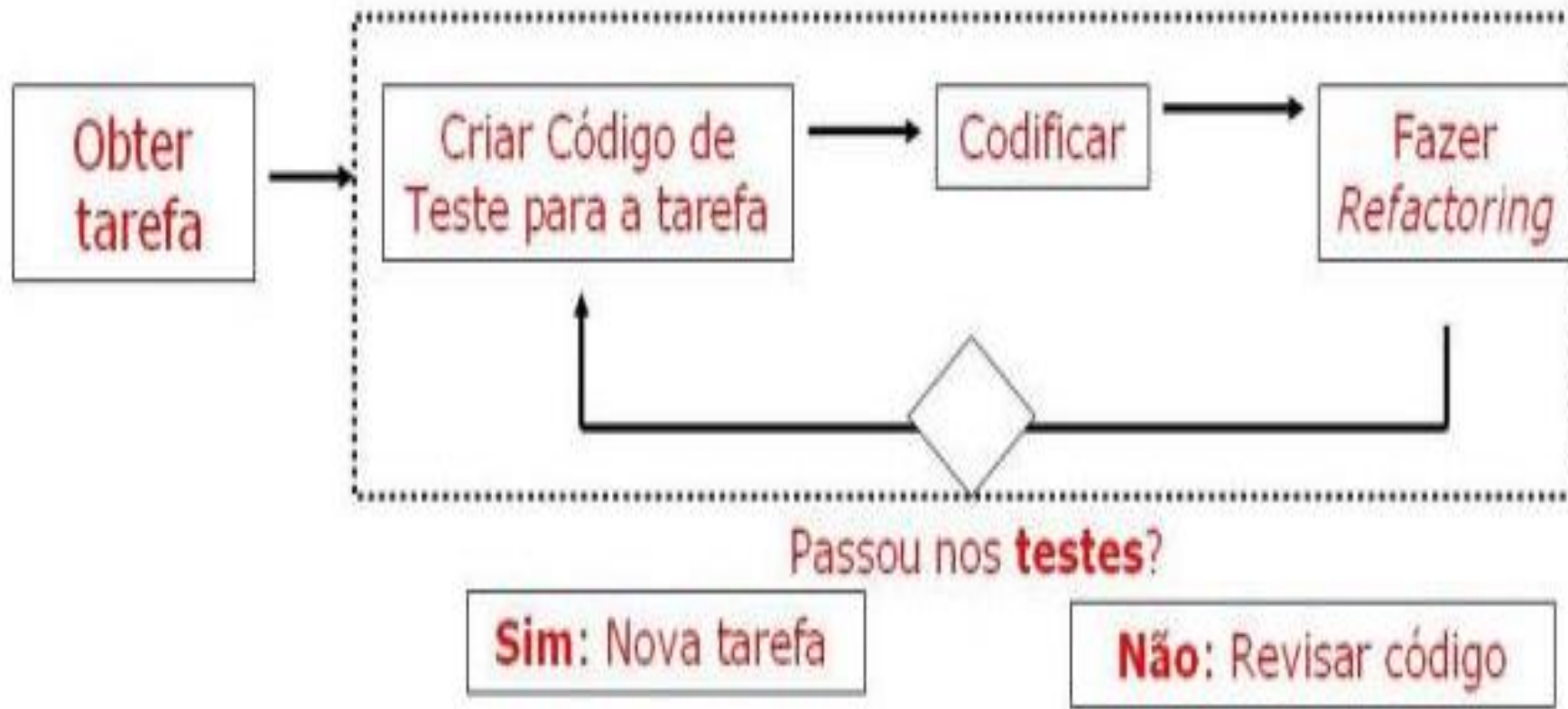
Programação Orientada a Objetos

- **TDD – *Test Driven Development***

- Desenvolvimento orientado a testes é uma técnica de desenvolvimento de software que baseia em um ciclo curto de repetições:
 - Primeiramente o desenvolvedor escreve um teste automatizado que define uma melhoria desejada ou uma nova funcionalidade.
 - Então, é produzido código que possa ser validado pelo teste
 - Posteriormente o código é refatorado para um código sob padrões aceitáveis.

Programação Orientada a Objetos

- TDD – *Test Driven Development***




Programação Orientada a Objetos


- **Ferramenta para Testes no IDE Eclipse**




 JUnit 4


The 5th major version of the programmer-friendly testing framework for Java and the JVM

 User Guide

 Javadoc

 Code & Issues

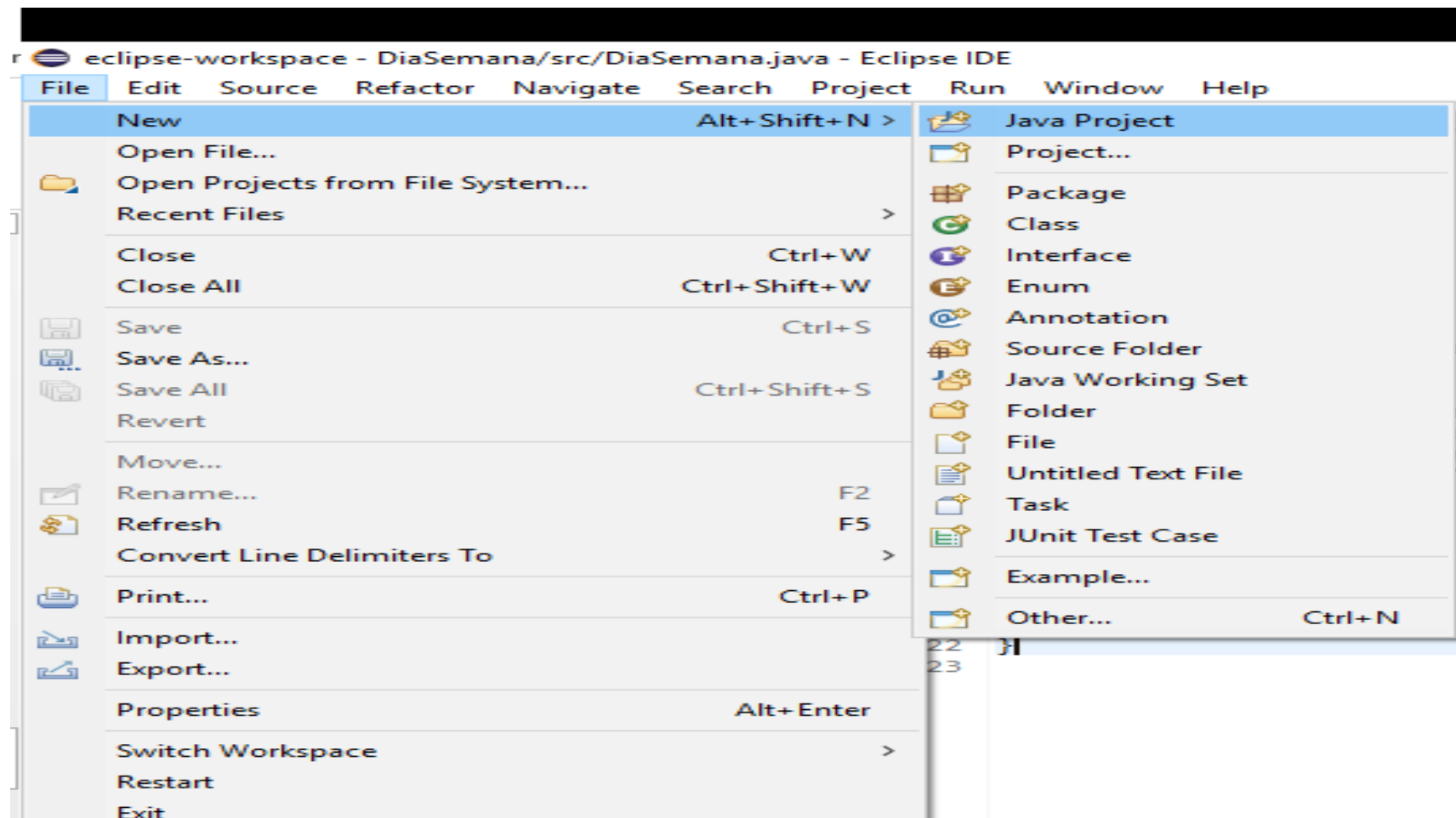
 Q & A

 Support JUnit

- **Fonte: <https://junit.org/junit5/>**

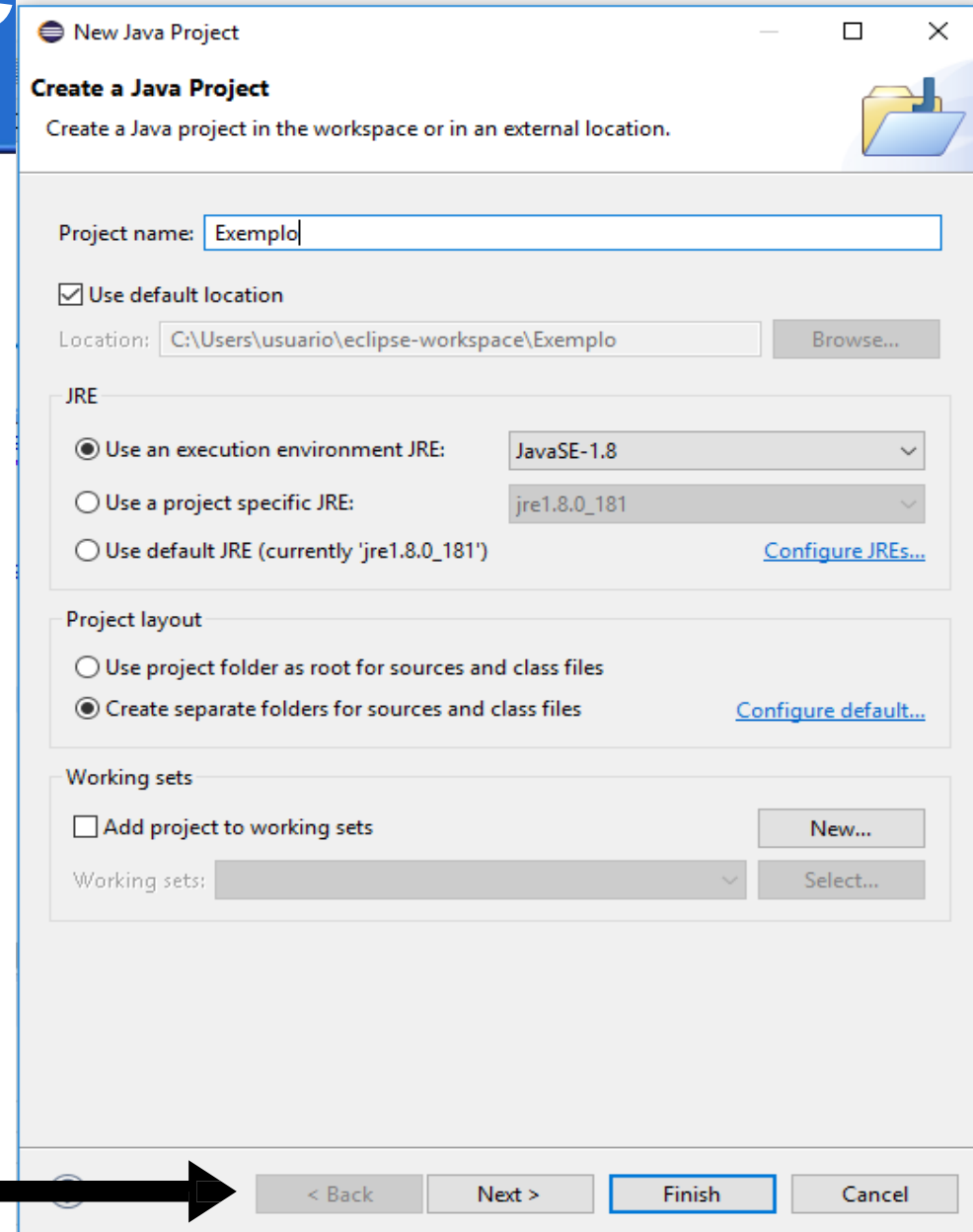
JUnit no Eclipse

- Crie um novo projeto na IDE Eclipse



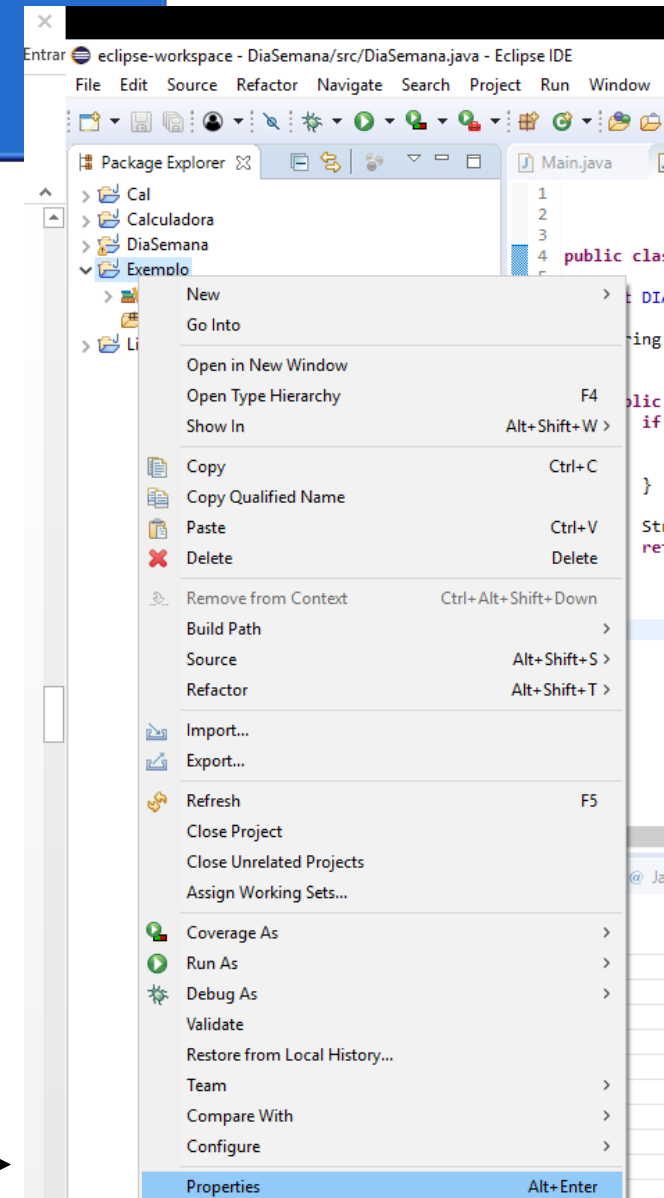
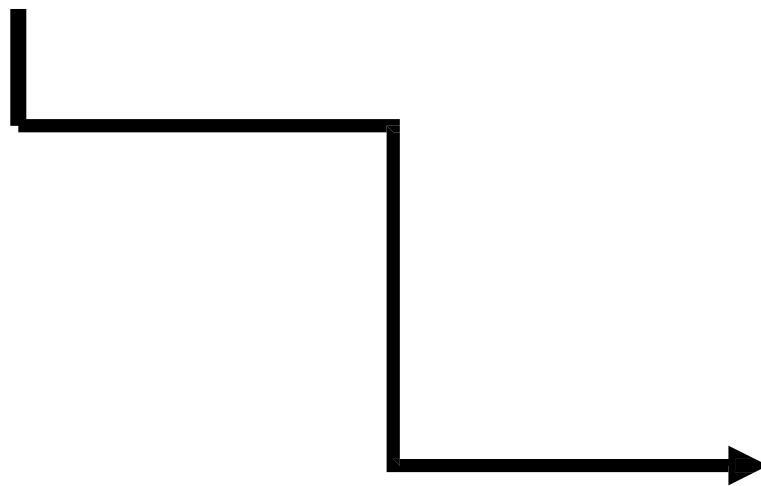
JUnit no Eclipse

- Adicione o nome ao projeto e clique em Finalizar



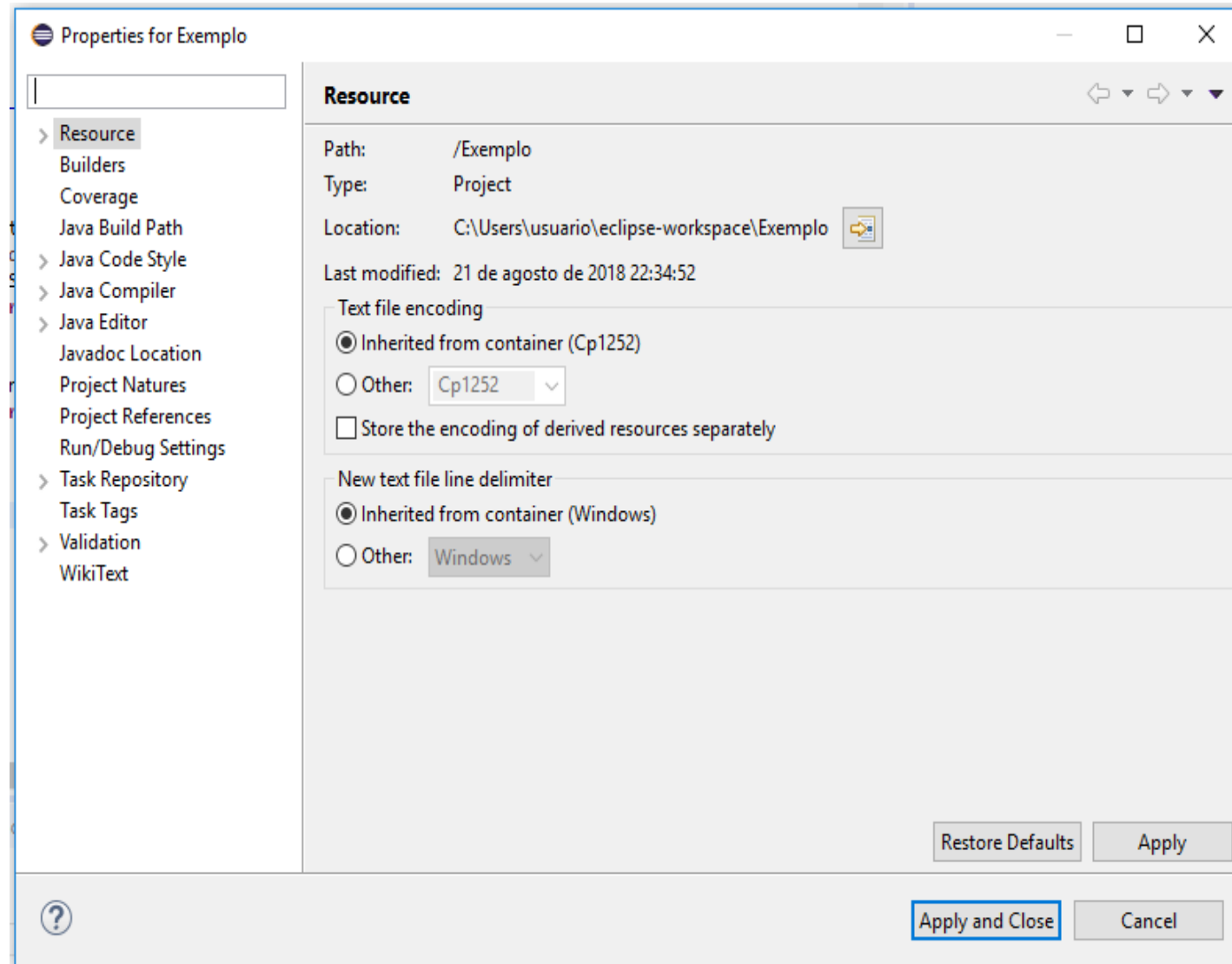
JUnit no Eclipse

- Após a criação:
 - Vá até a lista de pacotes
 - Selecione o projeto;
 - Clique com o botão direito;
 - No menu referente ao projeto, escolha propriedades



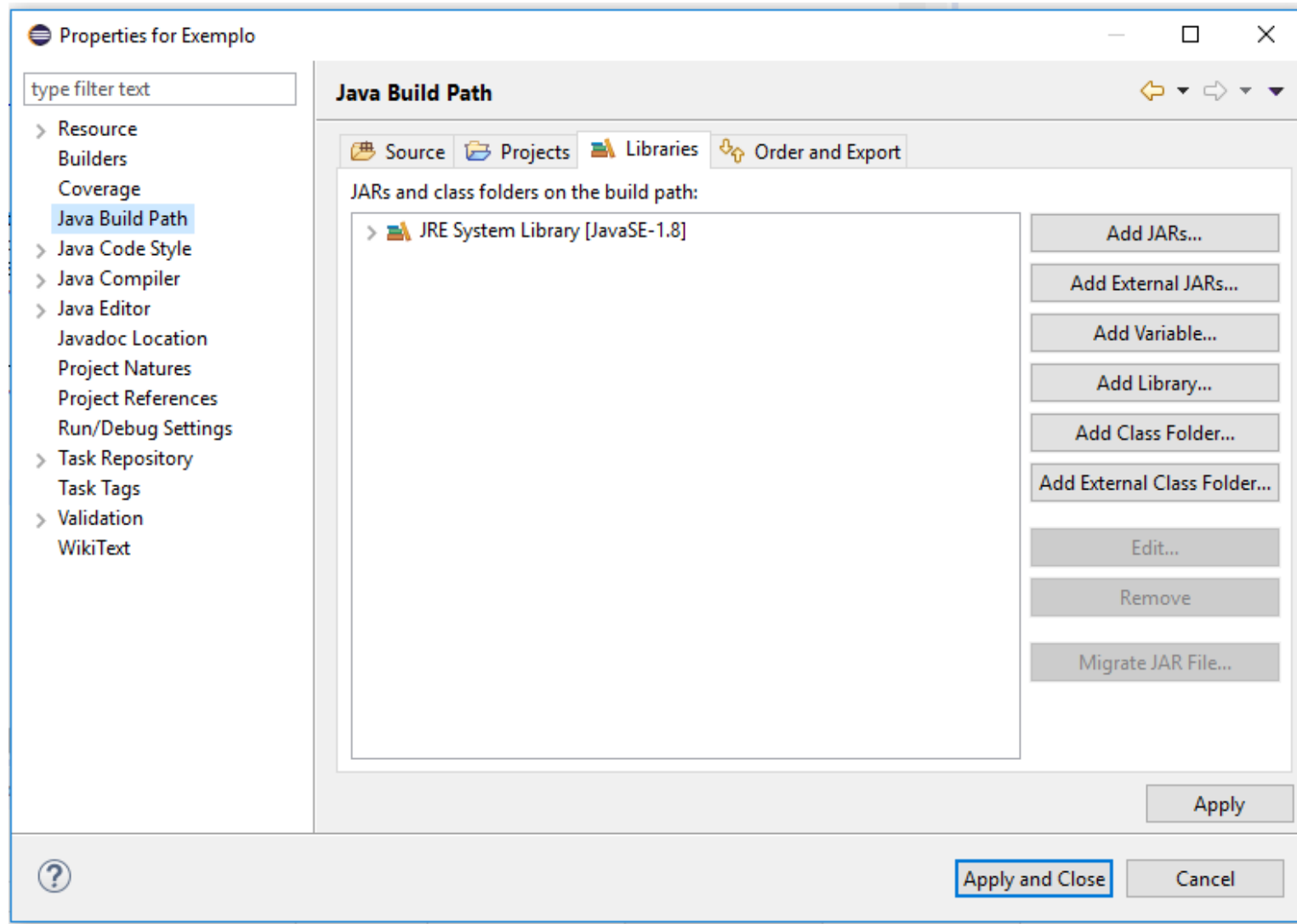
JUnit no Eclipse

- Na janela que será aberta, escolha a opção “Java Build Path”.



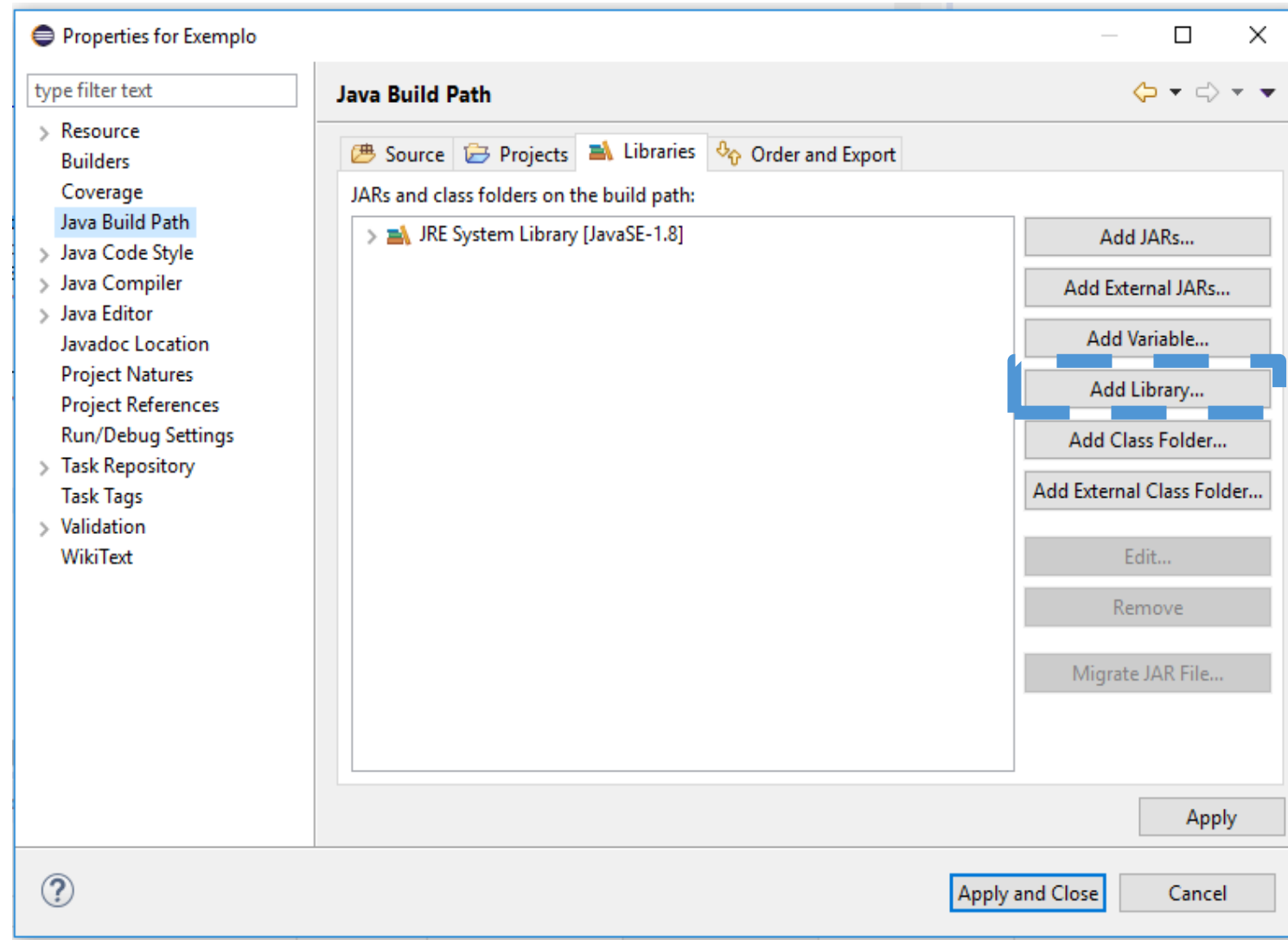
JUnit no Eclipse

- Observe as bibliotecas disponíveis para o projeto



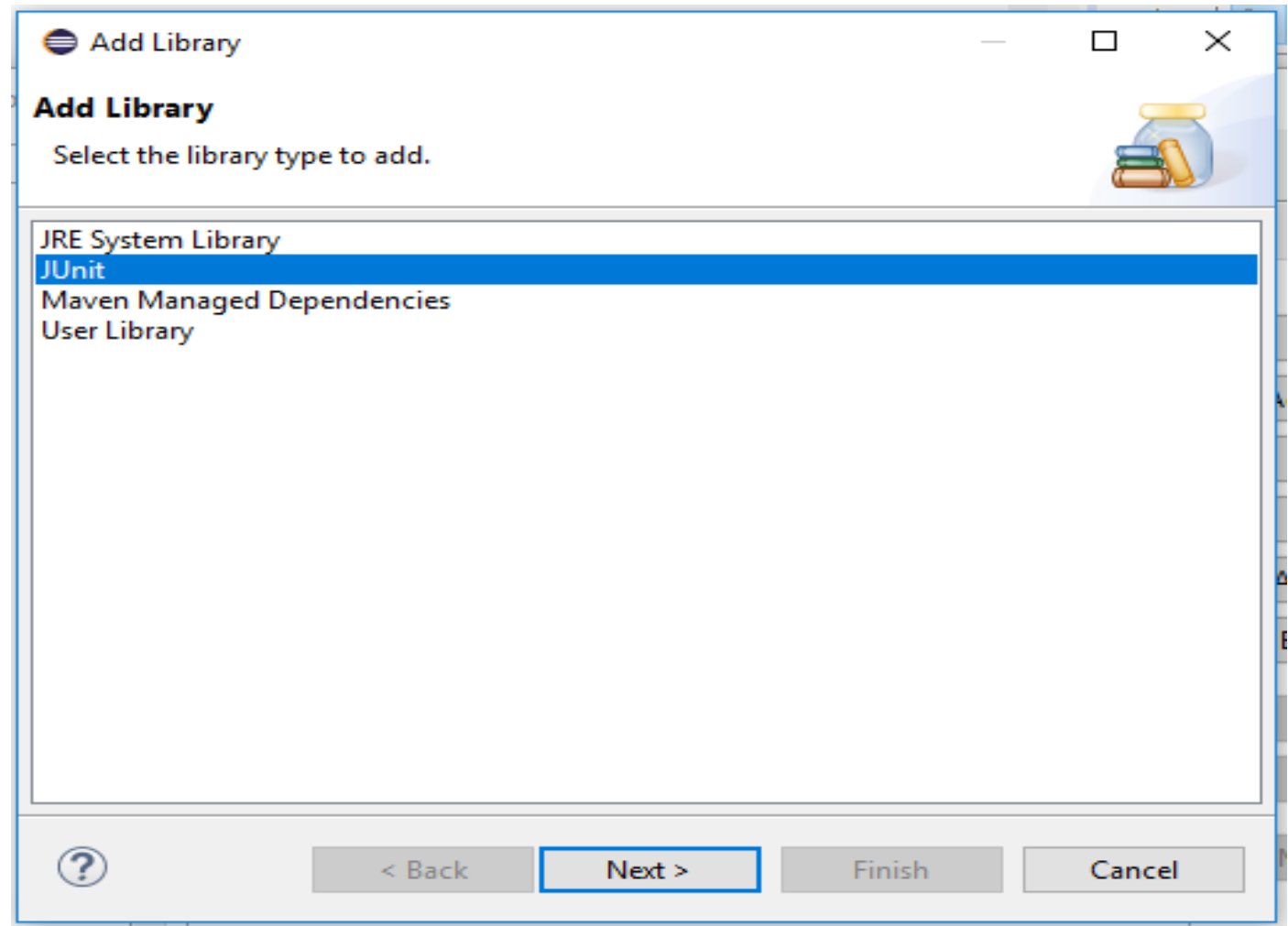
JUnit no Eclipse

- Clique no botão adicionar biblioteca



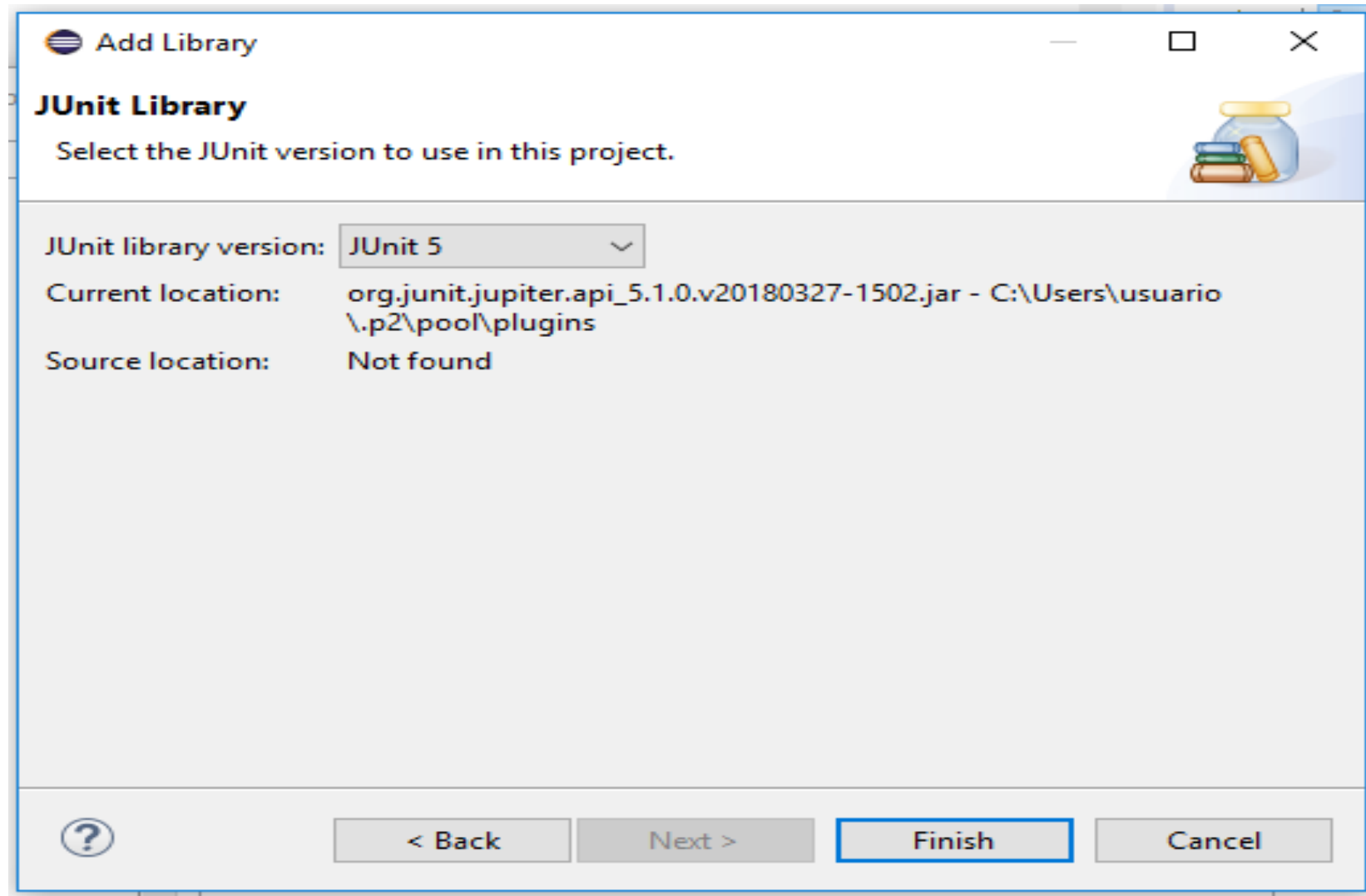
JUnit no Eclipse

- Na nova janela, selecione JUnit e clique em Próximo



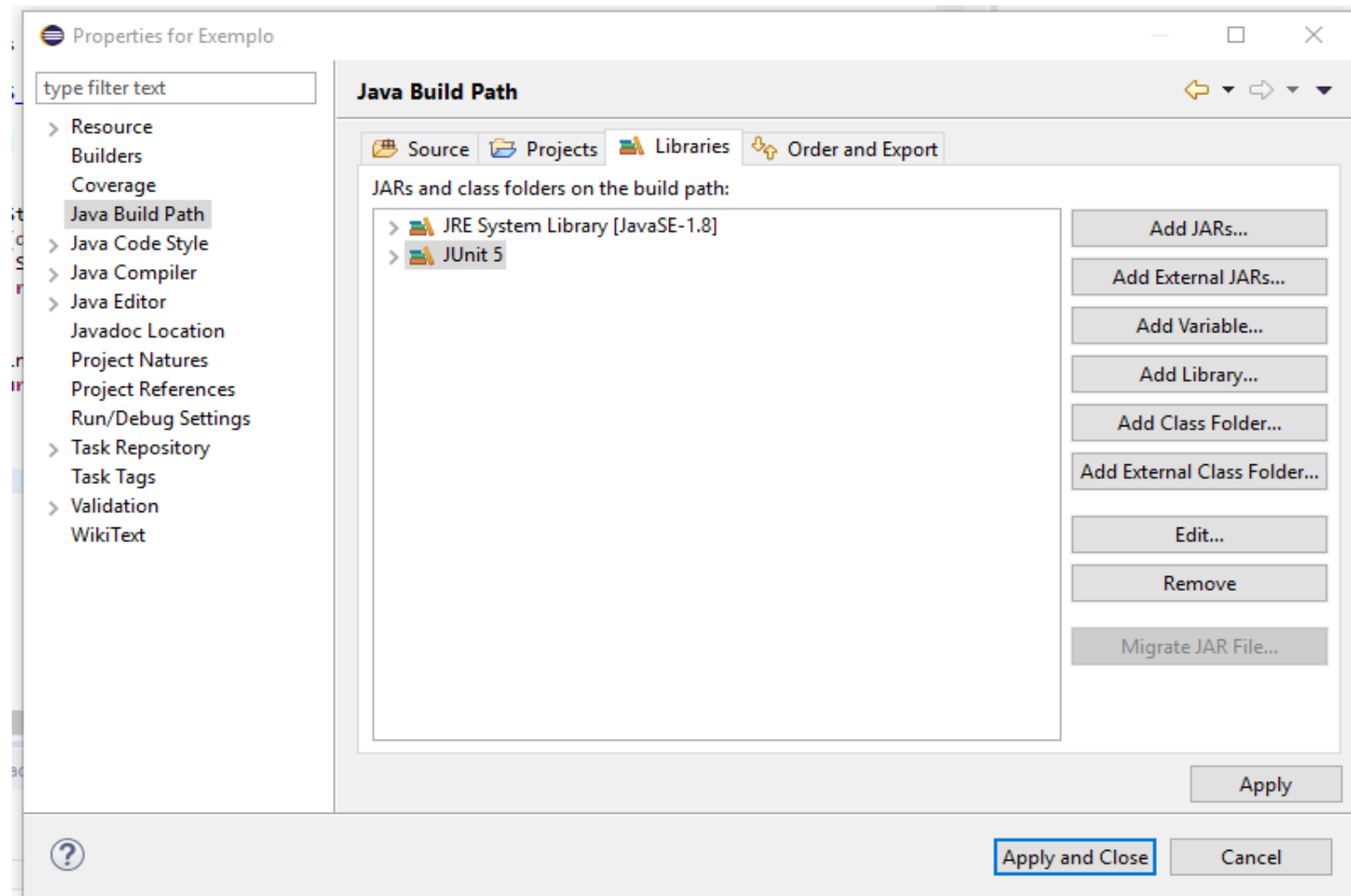
JUnit no Eclipse

- Na nova janela, selecione a versão do JUnit e clique em finalizar.



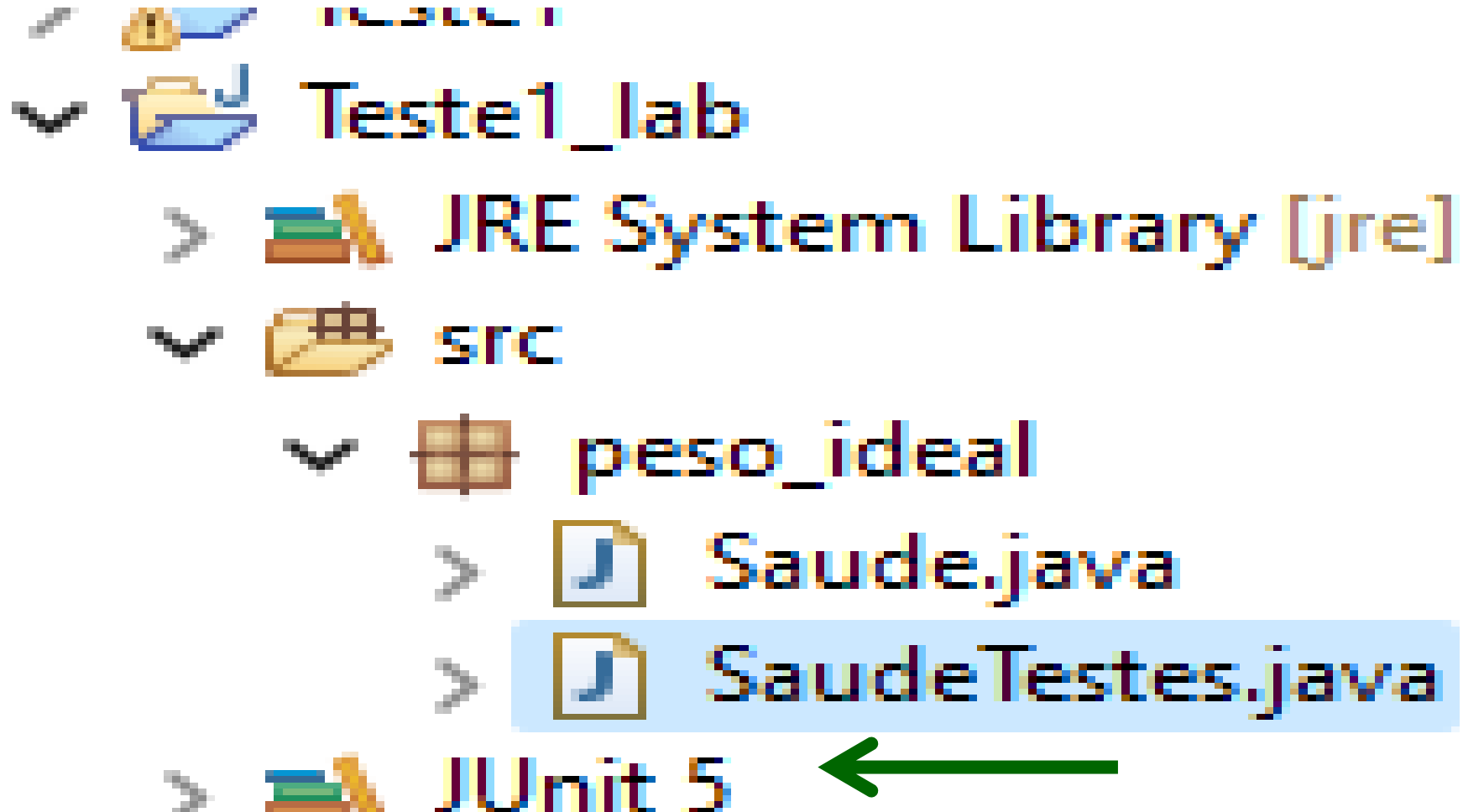
JUnit no Eclipse

- Por fim, clique em Aplicar e Fechar



JUnit no Eclipse

- *A ferramenta Junit deve aparecer no IDE Eclipse do Projeto*



Programação Orientada a Objetos

. *Ferramenta para Testes no IDE VS Code*

The screenshot displays the Visual Studio Code interface with the 'Extension Pack for Java' installed. The left sidebar shows the 'EXTENSÕES: MARKETPLACE' view with a search for 'java'. The main area shows the details of the 'Extension Pack for Java' (v0.25.11) by Microsoft, which includes 19,997,812 downloads and a 4-star rating (59 reviews). Below the main extension, a 'Pacote de Extensões (6)' section lists the included components: IntelliCode (AI-assisted development), Language Support for Java(TM) by Red Hat (Java Linting, Intellisense, formatting, refactor...), and three other extensions (Debugger for Java, Maven for Java, and Test Runner for Java) which are also listed in the left sidebar. The right sidebar shows the 'Categorias' (Categories) section with options like Programming Languages, Snippets, Linters, Debuggers, Formatters, and Extension Pack.

EXTENSÕES: MARKETPLACE...

Extension Pack for Java v0.25.11 [Visualizar](#)
Microsoft [microsoft.com](#) | 19.997.812 | ★★★★★ (59)
Popular extensions for Java development that provides Java IntelliSense, de

[Instalar](#) [Configurar](#)

DETALHES CONTRIBUIÇÕES DE RECURSO LOG DE MUDANÇAS

Pacote de Extensões (6)

- IntelliCode**
AI-assisted development
Microsoft [Instalar](#)
- Language Support for Java(TM) by Red Hat**
Java Linting, Intellisense, formatting, refactor...
Red Hat [Instalar](#)
- Debugger for Java**
A lightweight Java debugge...
Microsoft [Instalar](#)
- Maven for Java**
Manage Maven projects, ex...
Microsoft [Instalar](#)
- Test Runner for Java**
Run and debug JUnit or Test...
Microsoft [Instalar](#)
- Project Manager for Java**
Manage Java projects in Vis...
Microsoft [Instalar](#)

Categorias

- Programming Languages
- Snippets
- Linters
- Debuggers
- Formatters
- Extension Pack

Programação Orientada a Objetos

. Ferramenta para Testes no IDE VS Code

The screenshot displays the Visual Studio Code interface with the 'Extension Pack for Java' installed. The top bar shows the extension name and version (v0.25.11). The left sidebar lists the installed extensions: 'Extension Pack for Java', 'Debugger for Java', 'Maven for Java', 'Test Runner for Java', and 'Project Manager for Java'. The main panel shows the details of the 'Extension Pack for Java', including its description, version, and a list of sub-extensions. A white arrow points to the 'Test Runner for Java' sub-extension in the 'Pacote de Extensões (6)' section.

Extensão: Extension Pack for Java - Django-3-by-Example - Visual Studio Code

EXTENSÕES: MARKETPLACE

java

Extension Pack for Java v0.25.11 Visualizar
Microsoft | 19.997.812 | ★★★★★ (59)
Popular extensions for Java development that provides Java IntelliSense.

Desabilitar | Desinstalar | Alternar para a Versão de Pré-Lançamento

Esta extensão foi habilitada globalmente.

DETALHES | CONTRIBUIÇÕES DE RECURSO | LOG DE MUDANÇAS

Pacote de Extensões (6)

- Manage Maven projects, execute goals, gen... | Microsoft
- Test Runner for Java** | Microsoft
Run and debug JUnit or TestNG test cases.

Categorias

- Programming Languages
- Snippets
- Linters
- Debuggers
- Formatters
- Extension Packs

Programação Orientada a Objetos

. *Ferramenta para Testes no IDE Vs Code*

Testing Java with Visual Studio Code

Edit

Testing Java in Visual Studio Code is enabled by the [Test Runner for Java](#) extension. It's a lightweight extension to run and debug Java test cases.

Overview

The extension supports the following test frameworks:

- [JUnit 4](#) (v4.8.0+)
- [JUnit 5](#) (v5.1.0+)
- [TestNG](#) (v6.9.13.3+)



The [Test Runner for Java](#) works with the [Language Support for Java™](#) by Red Hat and [Debugger for Java](#) extensions to provide the following features:

Programação Orientada a Objetos

- ***Ferramenta para Testes (JUnit)***
- O *JUnit* é um *framework* com suporte a testes automatizados para a linguagem de programação Java.
- Ela tem sido importante na popularização do TDD, em português, "Desenvolvimento Orientado a Testes".



Programação Orientada a Objetos

- ***JUnit* - Testes Unitários**

- 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 podem ser 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.

Programação Orientada a Objetos

- ***JUnit*** - Testes **Unitários**

- O Java fornece uma completa API (conjunto de classes) para construir os testes em aplicações gráficas e em modo console para executar os testes criados.

Programação Orientada a Objetos

- ***JUnit* - Testes Unitários**

- Os principais motivos que favorecem o uso desse *framework* são:
 - Verifica se cada unidade de código funciona da forma esperada.
 - Facilita a criação, execução automática de testes e a apresentação dos resultados.
 - É Orientado a Objeto
 - É free e pode ser baixado em: www.junit.org

Programação Orientada a Objetos

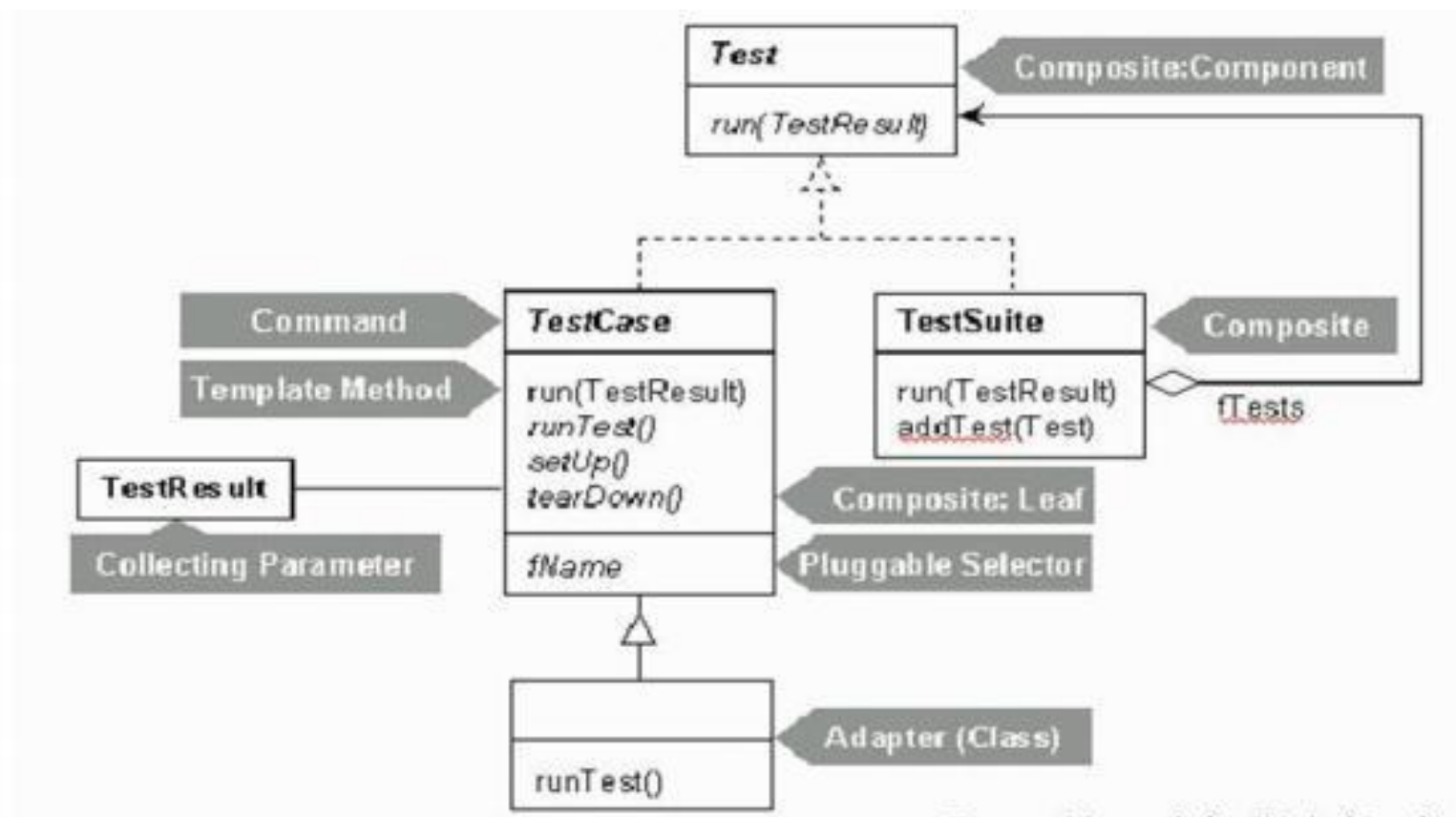
- ***JUnit* - Testes Unitários**

- Para usar o *Junit*:
- download do arquivo *junit.jar* em www.junit.org e inclusão no *classpath* para compilar e rodar os programas de teste.
- *JUnit* já vem configurado nas versões recentes de IDE's como Eclipse, VS Code, NetBeans, JBuilder, BlueJ e outros.

Programação Orientada a Objetos

- ***JUnit* - Testes Unitários**

- Classes da API



Programação Orientada a Objetos

- ***JUnit* - Testes Unitários**

- Verificação através de comandos Asserts
 - assertEquals(valorEsperado,valorAtual)
 - assertTrue(condicao)
 - assertFalse(condicao)
 - assertNull(objeto)
 - assertNotNull(objeto)
 - Existem vários comandos no Junit que podem ser vista no guia do usuário da ferramenta em:

<https://junit.org/junit5/docs/current/user-guide/>

Programação Orientada a Objetos

- ***JUnit*** - Testes Unitários

- O comando abaixo verifica se o retorno do método `metodoX`, executado com o valor de entrada (parâmetro) 1 é igual a 2.

```
assertEquals(2, metodoX(1));
```

Resultado
Esperado

Retorno da chamada do
método

Programação Orientada a Objetos

- **JUnit - Testes Unitários**

- **Como Implementar ?**

- Para cada classe a ser testada, crie uma classe que estenda junit.framework.TestCase

```
import junit.framework.*;
```

```
class MinhaClasseTest extends TestCase {  
    ...  
}
```

- Para cada método a ser testado defina um método public void test???() no test case MinhaClasse:

- MinhaClasseTest:

- public void testSoma()

```
class MinhaClasse{  
    ...  
}
```

```
class MinhaClasse {  
  
    public int Soma(Object o) {  
        ..  
    }  
}
```

Programação Orientada a Objetos

- ***JUnit* - Testes Unitários**

- Uma forma de Implementar – Junit 4.0 ou posterior

- Métodos de teste definidos com as **anotações** @Test

```
import junit.*;

class MinhaClasseTest {

    @Test
    public void somaTeste() {
        ...
    }
}
```

```
class MinhaClasse {

    public int Soma(Object o) {
        ..
    }
}
```

Programação Orientada a Objetos

- **Principais Anotações *JUnit***

- **@Test** – Identifica método que contem teste
- **@After** - Identifica método para ser executado após cada método de teste
- **@Before** - Identifica método para ser executado antes cada método de teste
- **@AfterClass** - Identifica método estático para ser executado após a execução de todos os métodos de teste da classe
- **@BeforeClass** - Identifica método estático para ser executado antes da execução de todos os métodos de teste da classe
- Existem várias anotações no Junit que podem ser vista no guia do usuário da ferramenta em:

<https://junit.org/junit5/docs/current/user-guide/>

Programação Orientada a Objetos

- ***JUnit*** - Testes Unitários

- Resultados Possíveis



- ***Sucesso***



- ***Falha***



- ***exceção***