

DISCIPLINA: PROJETO DE SISTEMAS APLICADO AS MELHORES PRÁTICAS EM QUALIDADE DE SOFTWARE E GOVERNANÇA DE TI

**AULA 19 – TESTE INTEGRADO DE SOFTWARE (TESTES DE INTEGRAÇÃO)
ESTUDO DE CASO**

**PROFESSOR:
RENATO JARDIM PARDUCCI**

PROFRENATO.PARDUCCI@FIAP.COM.BR

[Renato Parducci - YouTube](#)

ESTUDOS DE CASO





Faça o processo completo de criação de uma nova Workspace, criação de um projeto MAVEN, catalogação da biblioteca do MOCKITO no POM e desenvolvimento dos testes com mock para validar o método de gravar cliente da classe Cliente, sem acionar de verdade a DAO a seguir (classes neste e o no próximo slide):

Cliente.java ✕

```
1
2 public class Cliente {
3     String nome;
4
5     public void gravarNome(String nomeCliente){
6         this.nome = nomeCliente;
7     }
8 }
```

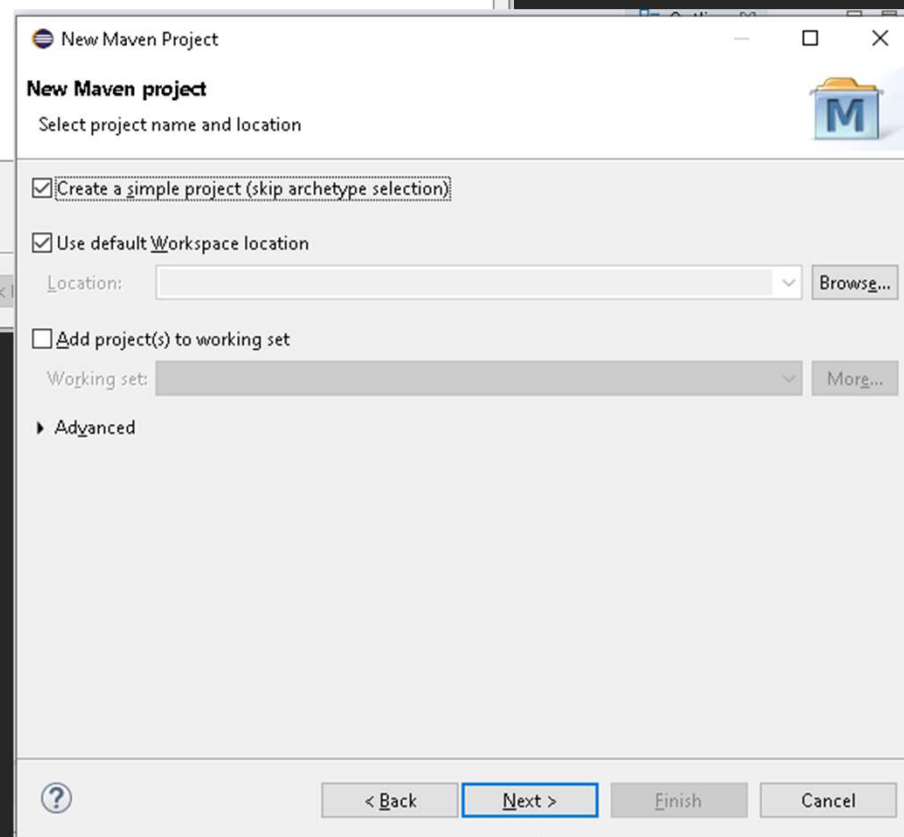
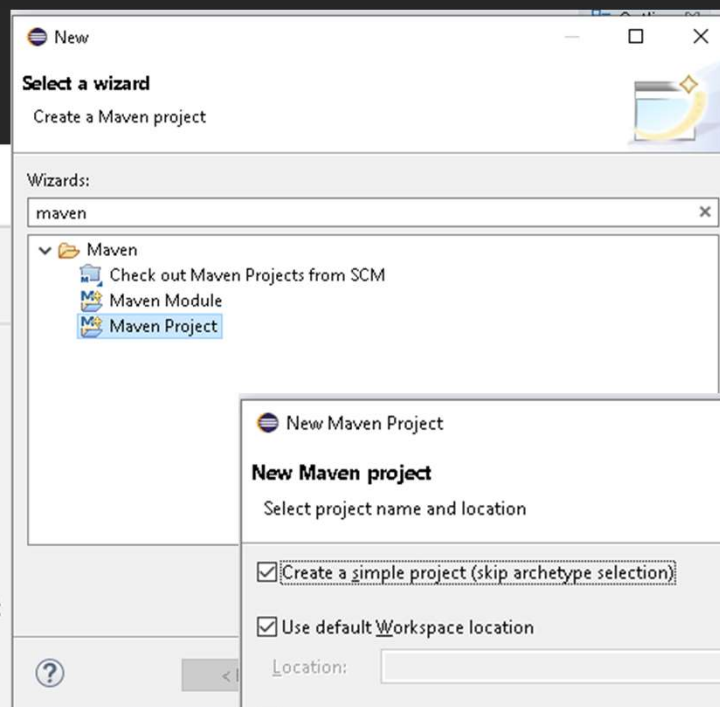
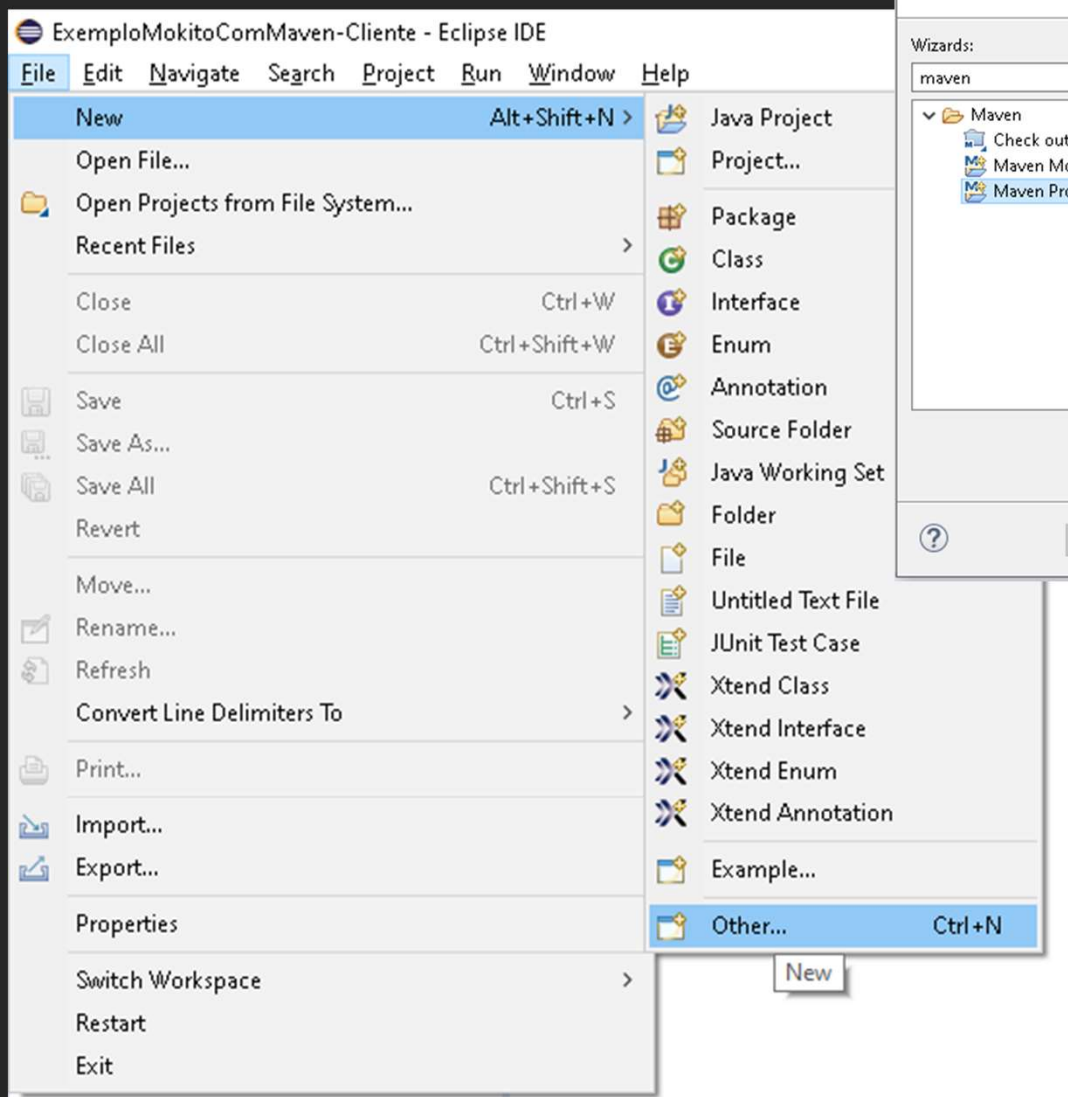


Classe a ser mockada:

```
ClienteDAO.java ✕  
1  
2 public class ClienteDAO {  
3     public boolean save(String nomeCli) {  
4         System.out.println("Não implementado");  
5         return true;  
6     }  
7 }  
8
```

SOLUÇÃO DO EXERCÍCIO DE AULA

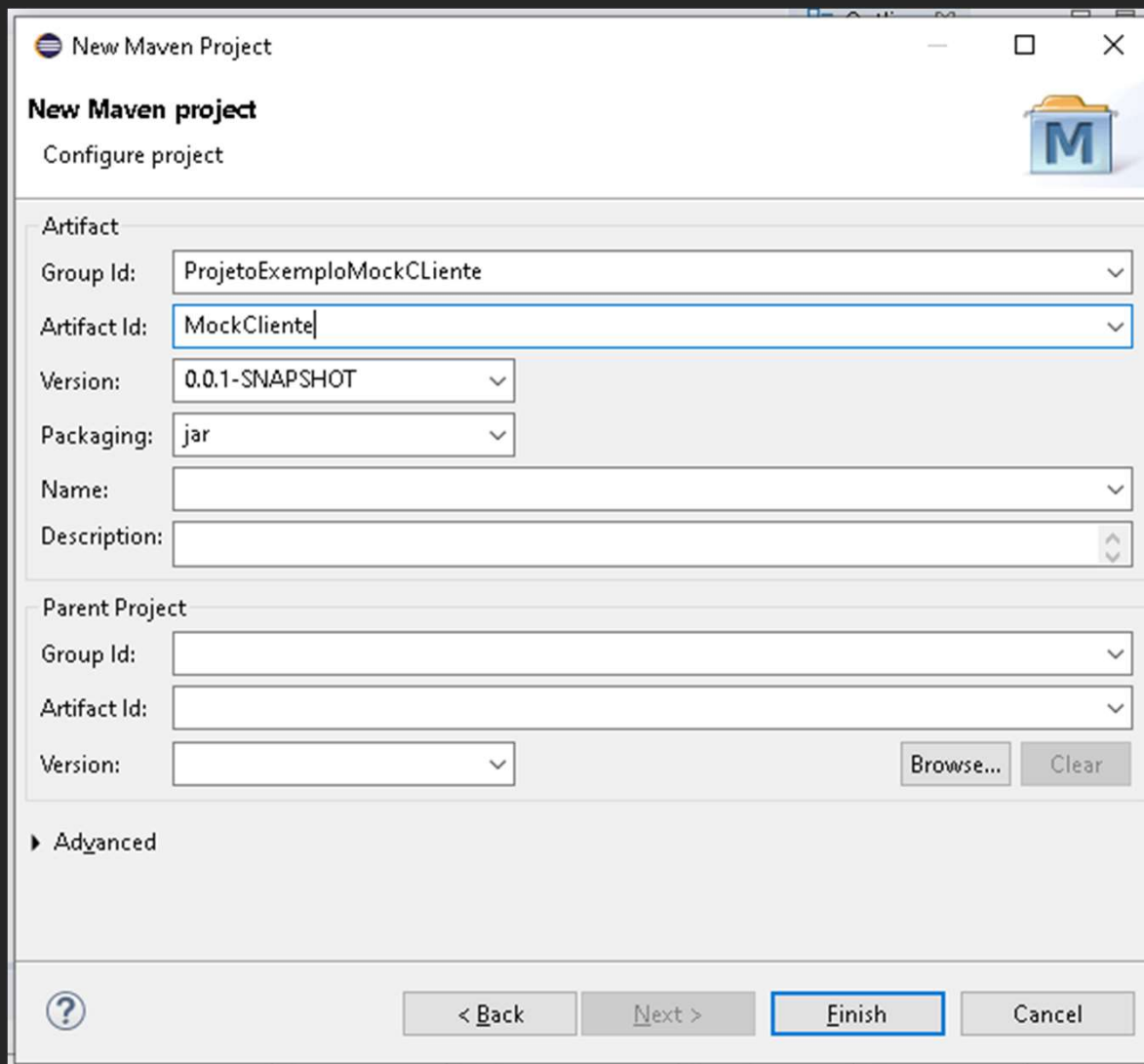
- 1º) Crie a pasta/workspace
- 2º) Crie um projeto simples, MAVEN:



SOLUÇÃO DO EXERCÍCIO DE AULA

1º) Crie a pasta/workspace

2º) Crie um projeto simples, MAVEN:



New Maven Project

New Maven project
Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

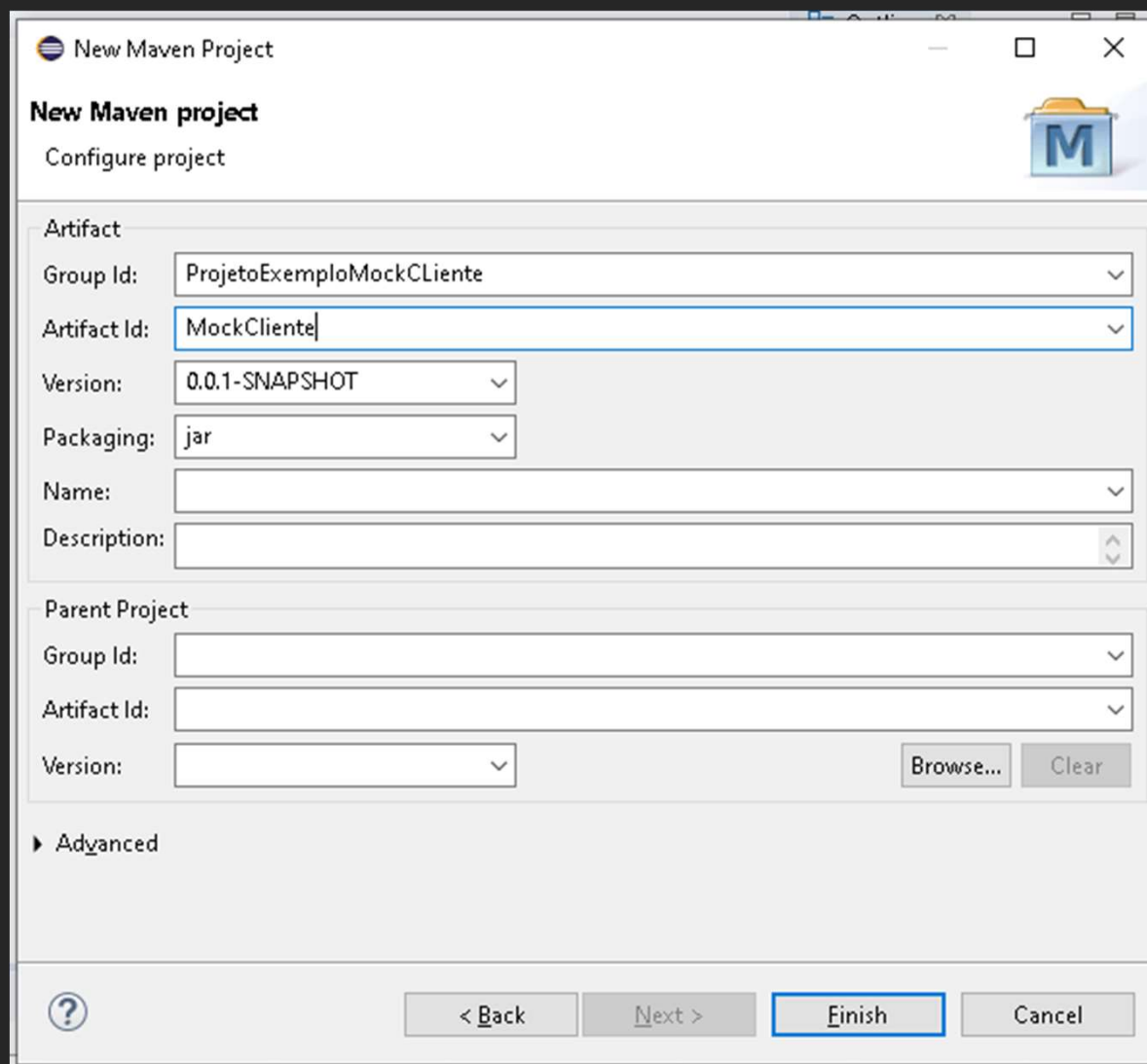
Artifact Id:

Version:

► **Advanced**

SOLUÇÃO DO EXERCÍCIO DE AULA

3º) Dê um nome ao artefato de projeto:



New Maven Project

Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

► **Advanced**

SOLUÇÃO DO EXERCÍCIO DE AULA

4º) Inclua as dependências no POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>ProjetoExemploMockCliente</groupId>
  <artifactId>MockCliente</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <java.version>13</java.version>
    <maven.compiler.source>13</maven.compiler.source>
    <maven.compiler.target>13</maven.compiler.target>
    <!-- JUnit 5 -->
    <junit.jupiter.version>5.8.1</junit.jupiter.version>

    <junit.plataform.version>1.7.2</junit.plataform.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>${junit.jupiter.version}</version>
      <scope>test</scope>
    </dependency>

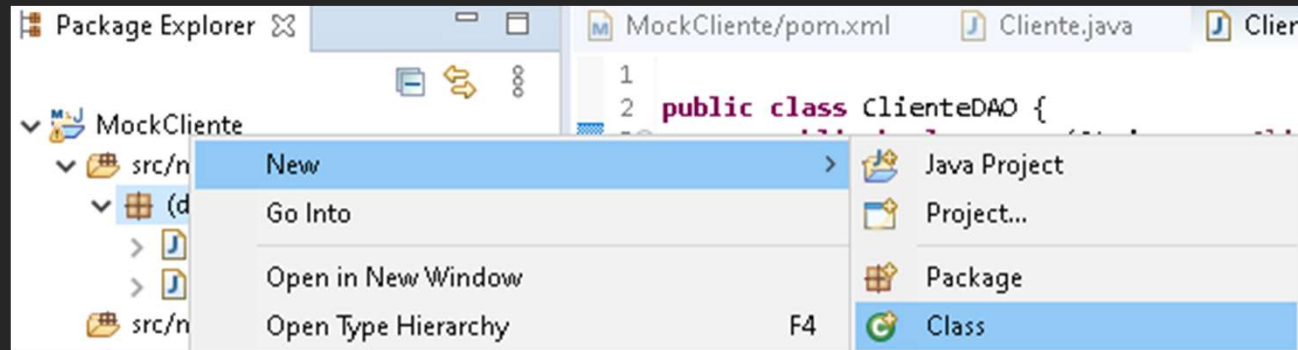
    <dependency>
      <groupId>org.json</groupId>
      <artifactId>json</artifactId>
      <version>20190722</version>
      <type>jar</type>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.mockito</groupId>
      <artifactId>mockito-junit-jupiter</artifactId>
      <version>4.0.0</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.mockito</groupId>
      <artifactId>mockito-core</artifactId>
      <version>3.12.4</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```


SOLUÇÃO DO EXERCÍCIO DE AULA

5º) Crie as classes de Entidade e DAO do projeto JAVA:



```

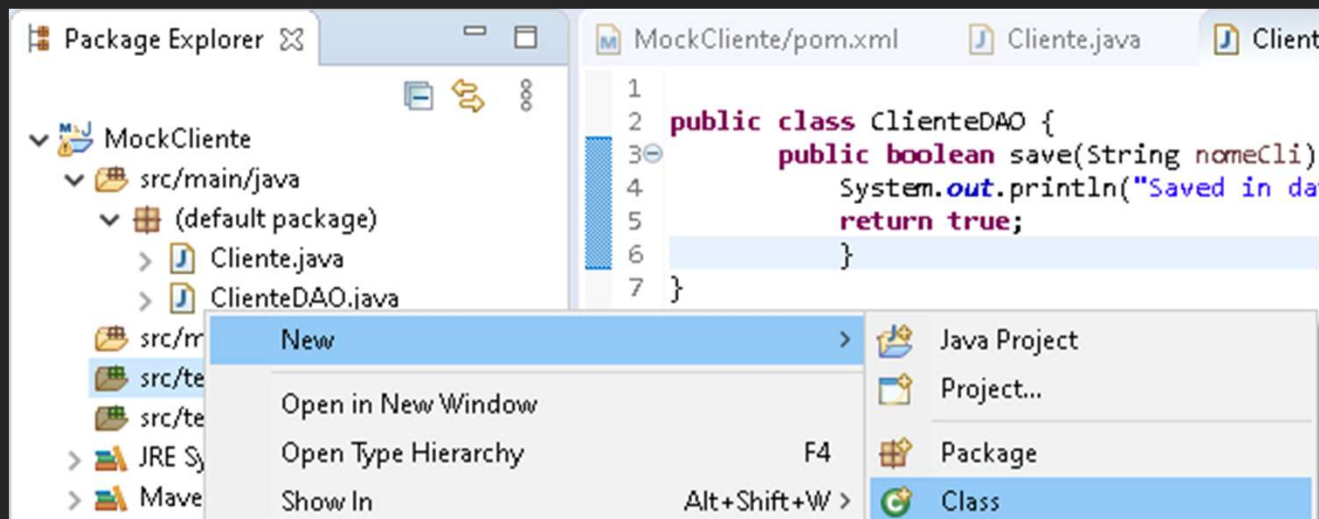
1
2 public class Cliente {
3     String nome;
4
5     public void gravarNome(String nomeCliente){
6         this.nome = nomeCliente;
7     }
8 }
    
```

```

1
2 public class ClienteDAO {
3     public boolean save(String nomeCli) {
4         System.out.println("Não implementado");
5         return true;
6     }
7 }
8
    
```

SOLUÇÃO DO EXERCÍCIO DE AULA

6º) Crie uma Classe de Teste no MAVEN:



```
1
2 public class TesteMockGravarCliente {
3
4 }
5
```

SOLUÇÃO DO EXERCÍCIO DE AULA

7º) Importe as bibliotecas para executar o teste, solucionando as quebras de código:

```
TesteMockGravarCliente.java ✕  
1 import static org.junit.Assert.assertEquals;  
2  
3 import org.junit.Test;  
4 import org.junit.jupiter.api.BeforeAll;  
5 import org.junit.runner.RunWith;  
6 import org.mockito.Mockito;  
7 import org.mockito.junit.MockitoJUnitRunner;  
8  
9 @RunWith(MockitoJUnitRunner.class)
```

SOLUÇÃO DO EXERCÍCIO DE AULA

8º) Defina seu teste, mocando a classe DAO (neste exemplo não está sendo usada a sentença @Before):

```

10
11 public class TesteMockGravarCliente {
12
13     static ClienteDAO clienteTeste;
14     Cliente clienteNovo = new Cliente();
15
16     // @BeforeAll
17     // public static void preparaTeste() {
18     //     clienteTeste = Mockito.mock(ClienteDAO.class);
19     //     Mockito.when(clienteTeste.save(Mockito.anyString())).thenReturn(true);
20     // }
21
22     @Test
23     public void insereCliente() {
24         clienteTeste = Mockito.mock(ClienteDAO.class);
25         Mockito.when(clienteTeste.save(Mockito.anyString())).thenReturn(true);
26
27         String nomeCliente = "Joaquina";
28
29         //Instancia o Objeto Cliente na Classe de Entidade
30         clienteNovo.gravarNome(nomeCliente);
31
32         //Usa o Objeto ClienteDAO Mockado para simular a gravação "fake" dos dados
33         boolean resultado = clienteTeste.save(nomeCliente);
34         if (resultado == true) {
35             System.out.println("Registro de cliente foi salvo!");
36         }
37         else {
38             System.out.println("Registro de cliente NÃO foi salvo!");
39         }
40         boolean retornoEsperado = true;
41         assertEquals(resultado, retornoEsperado);
42     }
43 }
44
45

```

SOLUÇÃO DO EXERCÍCIO DE AULA

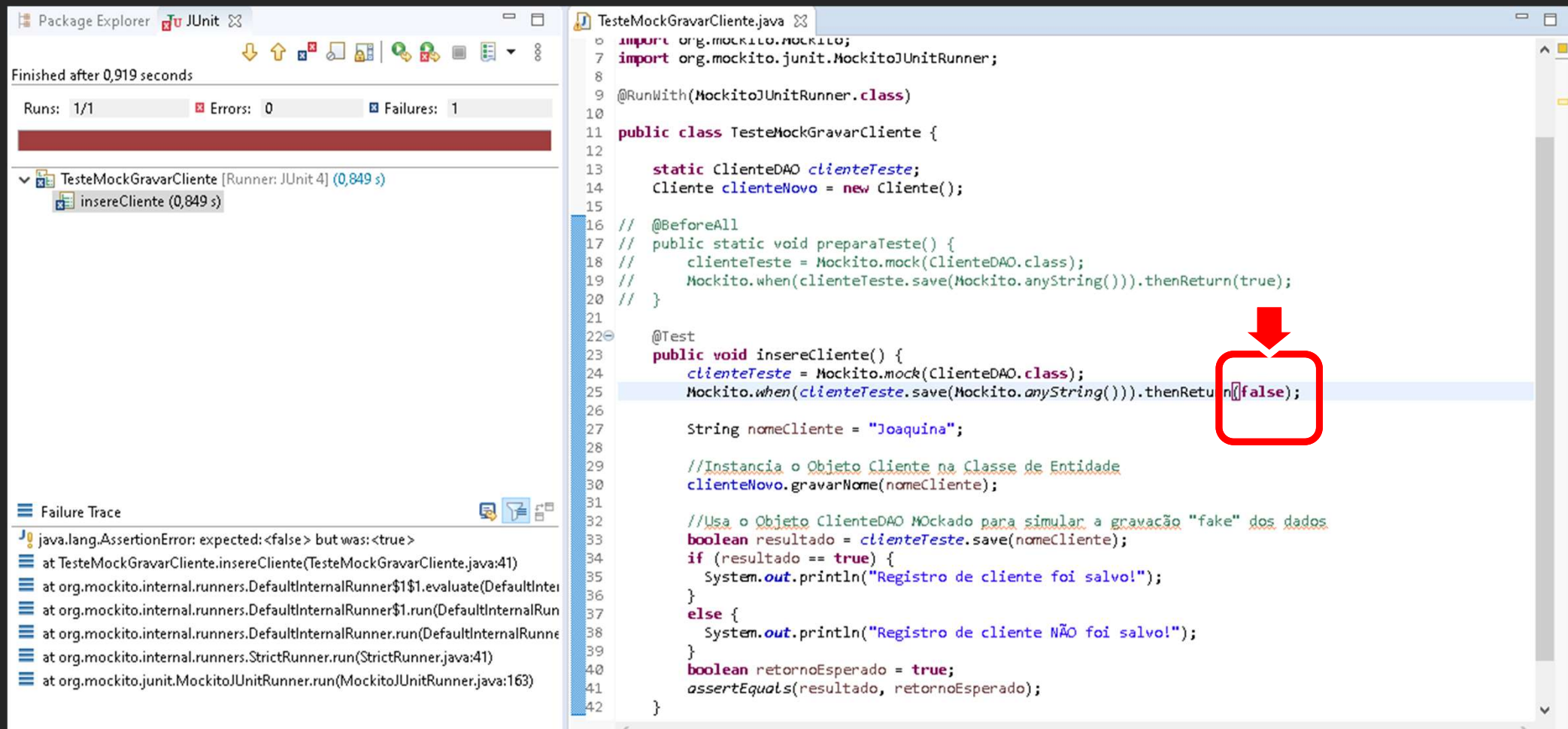
9º) Rode o teste mocado, forçando o retorno positivo da DAO:

The screenshot displays an IDE environment with the following components:

- Package Explorer:** Shows the project structure with a test class `TesteMockGravarCliente`.
- JUnit Runner:** Indicates the test finished after 0,876 seconds with 1/1 runs, 0 errors, and 0 failures.
- Test Results:** Shows the test `TesteMockGravarCliente [Runner: JUnit 4] (0,825 s)` passed, with a sub-test `insereCliente (0,825 s)`.
- Source Code:** The file `TesteMockGravarCliente.java` is open, showing:
 - Imports: `org.mockito.Mockito` and `org.mockito.junit.MockitoJUnitRunner`.
 - Annotations: `@RunWith(MockitoJUnitRunner.class)`.
 - Class: `TesteMockGravarCliente`.
 - Static fields: `clienteTeste` and `clienteNovo`.
 - Method: `insereCliente` which uses `Mockito.mock` and `Mockito.when` to simulate a successful save operation.
 - Assertion: `assertEquals(resultado, retornoEsperado);` where `retornoEsperado` is set to `true`.
- Context Menu:** A right-click context menu is visible on the left, showing options like 'Run As', 'Debug As', and 'Validate'. The 'Run As' option is highlighted.

SOLUÇÃO DO EXERCÍCIO DE AULA

10º) Tente mockar para forçar o erro da função da DAO:



Package Explorer JUnit

Finished after 0,919 seconds

Runs: 1/1 Errors: 0 Failures: 1

TesteMockGravarCliente [Runner: JUnit 4] (0,849 s)

insereCliente (0,849 s)

Failure Trace

```

java.lang.AssertionError: expected:<false> but was:<true>
    at TesteMockGravarCliente.insereCliente(TesteMockGravarCliente.java:41)
    at org.mockito.internal.runners.DefaultInternalRunner$1$1.evaluate(DefaultInternalRunner$1$1.java:54)
    at org.mockito.internal.runners.DefaultInternalRunner$1.run(DefaultInternalRunner$1.java:59)
    at org.mockito.internal.runners.DefaultInternalRunner.run(DefaultInternalRunner.java:64)
    at org.mockito.internal.runners.StrictRunner.run(StrictRunner.java:41)
    at org.mockito.junit.MockitoJUnitRunner.run(MockitoJUnitRunner.java:163)

```

```

TesteMockGravarCliente.java
import org.mockito.Mockito;
import org.mockito.junit.MockitoJUnitRunner;

@RunWith(MockitoJUnitRunner.class)

public class TesteMockGravarCliente {

    static ClienteDAO clienteTeste;
    Cliente clienteNovo = new Cliente();

    // @BeforeAll
    // public static void preparaTeste() {
    //     clienteTeste = Mockito.mock(ClienteDAO.class);
    //     Mockito.when(clienteTeste.save(Mockito.anyString())).thenReturn(true);
    // }

    @Test
    public void insereCliente() {
        clienteTeste = Mockito.mock(ClienteDAO.class);
        Mockito.when(clienteTeste.save(Mockito.anyString())).thenReturn(false);

        String nomeCliente = "Joaquina";

        //Instancia o Objeto Cliente na Classe de Entidade
        clienteNovo.gravarNome(nomeCliente);

        //Usa o Objeto ClienteDAO Mockado para simular a gravação "fake" dos dados
        boolean resultado = clienteTeste.save(nomeCliente);
        if (resultado == true) {
            System.out.println("Registro de cliente foi salvo!");
        }
        else {
            System.out.println("Registro de cliente NÃO foi salvo!");
        }
        boolean retornoEsperado = true;
        assertEquals(resultado, retornoEsperado);
    }
}

```



ESTUDOS DE CASO





DESAFIO

Tente criar um teste de integração para a API de consulta de playlist a seguir.

GET getAllPlaylists

```
https://us-central1-labenu-apis.cloudfunctions.net/labefy/playlists
```

DESCRIÇÃO:

Esta requisição serve para ver o `id` e o `name` de todas as suas playlists.

INPUT:

Headers

`Authorization`: token de autenticação da API

```
Authorization: "nome-sobrenome-turma"
```

OUTPUT:

Body

`quantity`: quantidade de playlists

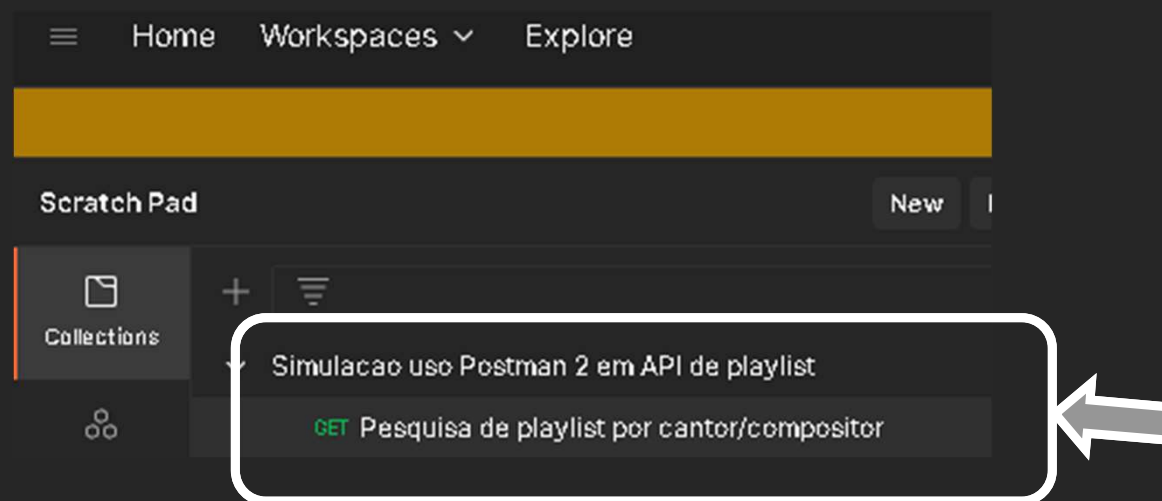
`list`: lista com as playlists

`id`: id de cada playlist

`name`: nome de cada playlist

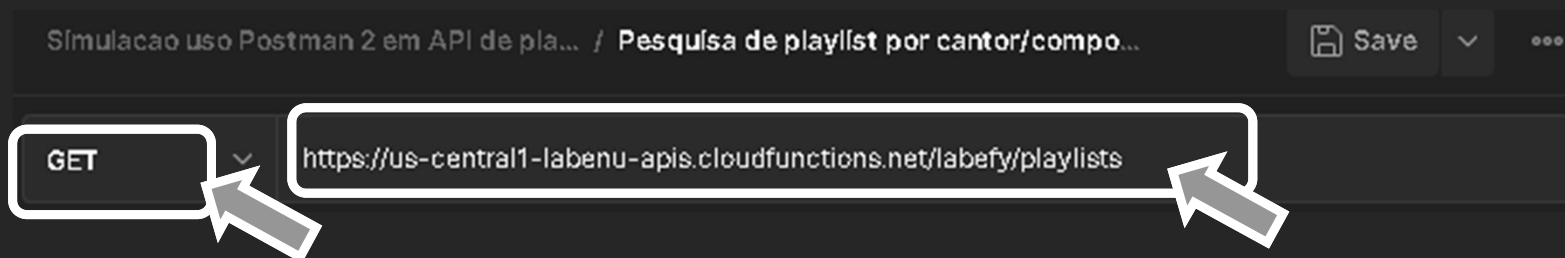
SOLUÇÃO DO EXERCÍCIO DE AULA

- 1º) Crie uma collection
- 2º) Crie a request na collection

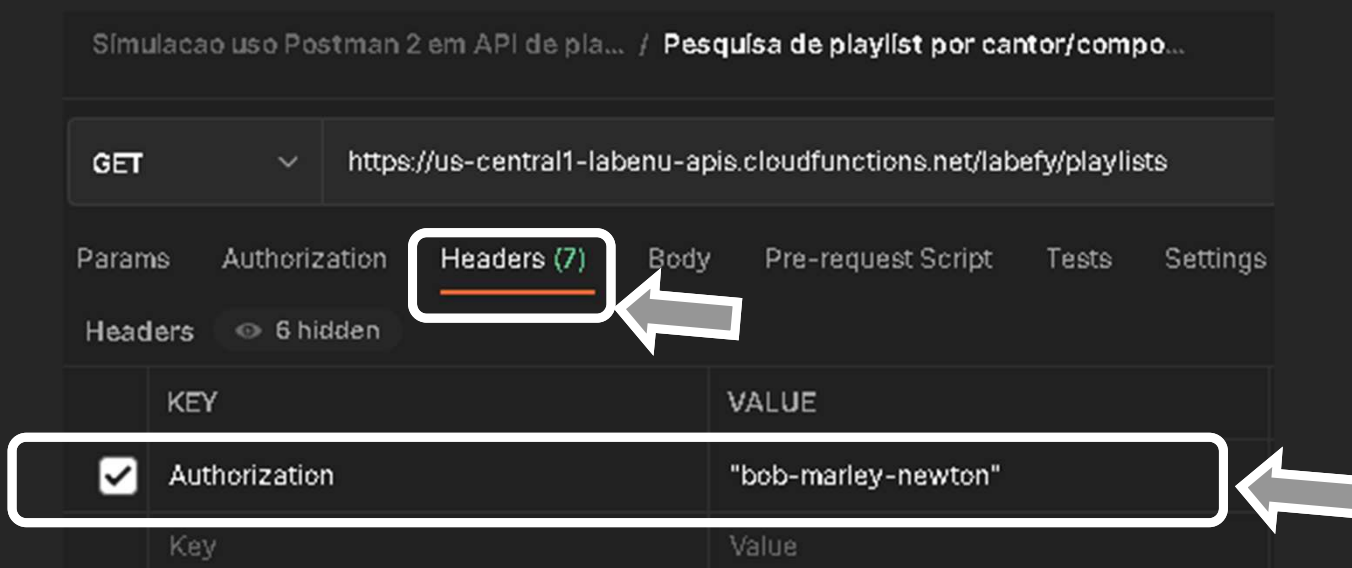


SOLUÇÃO DO EXERCÍCIO DE AULA

3º) Defina a requisição GET para a URL



4º) Ajuste o parâmetro de pesquisa (HEADER), conforme a documentação da API pede



SOLUÇÃO DO EXERCÍCIO DE AULA

5º) Execute e veja os resultados

Simulação uso Postman 2 em API de pla... / Pesquisa de playlist por cantor/compo...

GET <https://us-central1-labenu-apis.cloudfunctions.net/labefy/playlists> [Send](#)

Params Authorization **Headers (7)** Body Pre-request Script Tests Settings Cookies

Headers [6 hidden](#)

| | KEY | VALUE | DESCRIPTION | ... | Bulk Edit | Presets |
|-------------------------------------|---------------|---------------------|-------------|-----|-----------|---------|
| <input checked="" type="checkbox"/> | Authorization | "bob-marley-newton" | | | | |
| | Key | Value | Description | | | |

Body Cookies Headers (10) Test Results [200 OK](#) [2.02 s](#) [533 B](#) [Save Response](#)

Pretty Raw Preview Visualize JSON

```

1  {
2    "result": {
3      "quantity": 0,
4      "list": []
5    }
6  }
```

SOLUÇÃO DO EXERCÍCIO DE AULA

3º) Leia a documentação da API

GET getAllPlaylists

`https://us-central1-labenu-apis.cloudfunctions.net/labefy/playlists`

DESCRIÇÃO:

Esta requisição serve para ver o `id` e o `name` de todas as suas playlists.

INPUT:

Headers

`Authorization`: token de autenticação da API

`Authorization: "nome-sobrenome-turma"`

OUTPUT:

Body

`quantity`: quantidade de playlists

`list`: lista com as playlists

`id`: id de cada playlist

`name`: nome de cada playlist

ESTUDOS DE CASO



Teste de API por escolha do estudante