

AULA 1 – LABORATÓRIO – INICIANDO COM TESTES DE SOFTWARE (1,0)

TEMA: TESTE DE UNIDADE

RECURSOS: aplicação integracao.zip, ECLIPSE, JDK, MAVEN

ROTEIRO DA PRÁTICA

Básico de Aprendizado (0,15)

- 1) Verificar as variáveis de ambiente necessária (JDK) e (Maven)

```
julio-manoel@julio-manoel:~$ sudo update-alternatives --config java
Existem 2 escolhas para a alternativa java (disponibiliza /usr/bin/java).

   Seleccionar   Caminho                                     Prioridade Estado
-----
* 0              /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/java          369115136 modo automático
 1              /usr/lib/jvm/java-21-openjdk-amd64/bin/java      2111      modo manual
 2              /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/java          369115136 modo manual

Pressione <enter> para manter a escolha actual[*], ou digite o número da selecção: ^[[A^C
julio-manoel@julio-manoel:~$ sudo update-alternatives --config mvn
Há 1 escolha para a alternativa mvn (disponibiliza /usr/bin/mvn).

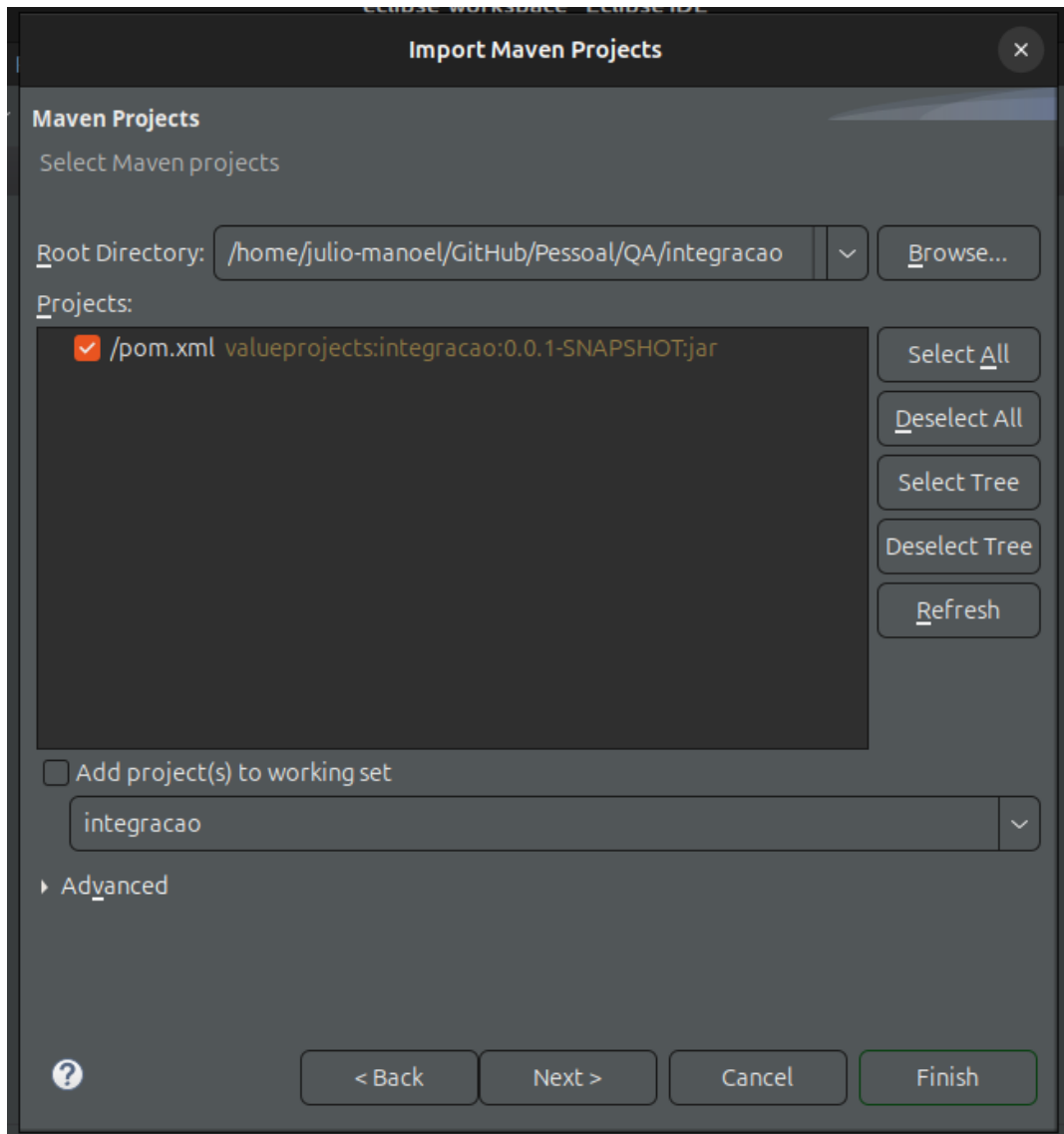
   Seleccionar   Caminho                                     Prioridade Estado
-----
* 0              /usr/share/maven/bin/mvn              300      modo automático
 1              /usr/share/maven/bin/mvn              300      modo manual

Pressione <enter> para manter a escolha actual[*], ou digite o número da selecção: █
```

- 2) Verificar a versão instalada em linhas de comando (Terminal, Shell)

```
julio-manoel@julio-manoel:~$ java --version
java 22.0.2 2024-07-16
Java(TM) SE Runtime Environment (build 22.0.2+9-70)
Java HotSpot(TM) 64-Bit Server VM (build 22.0.2+9-70, mixed mode, sharing)
julio-manoel@julio-manoel:~$ mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 22.0.2, vendor: Oracle Corporation, runtime: /usr/lib/jvm/jdk-22.0.2-oracle-x64
Default locale: pt_BR, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-41-generic", arch: "amd64", family: "unix"
```

- 3) Abra o projeto “Maven” no Eclipse:
 - a. Import Project Maven (botão direito do mouse / import e aponte para a pasta do projeto / Finish

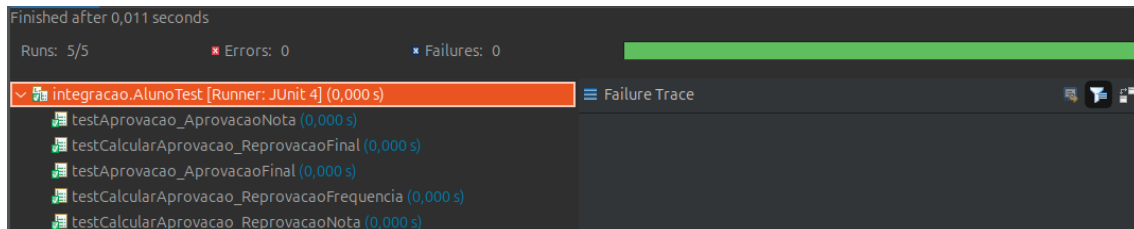


- 4) Arquivo.pom: “printar as dependências e plugins utilizados.

```
10 <dependencies>
11 <dependency>
12 <groupId>junit</groupId>
13 <artifactId>junit</artifactId>
14 <version>4.12</version>
15 <scope>test</scope>
16 </dependency>
17 </dependencies>
18
19 <plugins>
20 <plugin>
21 <artifactId>maven-compiler-plugin</artifactId>
22 <configuration>
23 <source>11</source>
24 <target>11</target>
25 </configuration>
26 </plugin>
27
28 <plugin>
29 <groupId>org.apache.maven.plugins</groupId>
30 <artifactId>maven-pmd-plugin</artifactId>
31 <version>3.13.0</version>
32 <executions>
33 <execution>
34 <phase>verify</phase>
35 <goals>
36 <goal>check</goal>
37 </goals>
38 </execution>
39 </executions>
40 </plugin>
41
42 <plugin>
43 <groupId>org.jacoco</groupId>
44 <artifactId>jacoco-maven-plugin</artifactId>
45 <version>0.8.5</version>
46 <executions>
47 <execution>
48 <goals>
49 <goal>prepare-agent</goal>
50 <goal>report</goal>
51 </goals>
52 </execution>
53 </executions>
54 </plugin>
55
56 <plugin>
57 <artifactId>maven-site-plugin</artifactId>
58 <version>3.7.1</version>
59 </plugin>
60 </plugins>
```

5) Testar a aplicação via Eclipse

1. Print os resultados com Junit



2. Print os resultados com Maven clean

```
[INFO] --- clean:3.2.0:clean (default-clean) @ integracao ---
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.3.4/maven-shared-ut
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.3.4/maven-shared-ut
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/34/maven-shared
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/34/maven-shared-
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/34/maven-parent-34.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/34/maven-parent-34.pom (43 kB at 1
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/commons-io/commons-io/2.6/commons-io-2.6.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/commons-io/commons-io/2.6/commons-io-2.6.pom (14 kB at 679 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-parent/42/commons-parent-42.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-parent/42/commons-parent-42.pom (68 k
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom (16 kB at 627 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.3.4/maven-shared-ut
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.3.4/maven-shared-ut
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/commons-io/commons-io/2.6/commons-io-2.6.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/commons-io/commons-io/2.6/commons-io-2.6.jar (215 kB at 6.1 MB/s)
[INFO] Deleting /home/julio-manoel/GitHub/Pessoal/QA/integracao-main/target
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.886 s
[INFO] Finished at: 2024-09-05T21:52:11-03:00
[INFO] -----
```

3. Print os resultados com Maven test

```
[INFO] -----  
[INFO] T E S T S  
[INFO] -----  
[INFO] Running integracao.AlunoTest  
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.035 s -- in integracao.AlunoTest  
[INFO] Results:  
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 7.375 s  
[INFO] Finished at: 2024-09-05T21:53:09-03:00  
[INFO] -----
```

4. Print os resultados do relatório gerado pelo Junit

```
1 -----  
2 Test set: integracao.AlunoTest  
3 -----  
4 Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.035 s -- in integracao.AlunoTest  
5 |
```

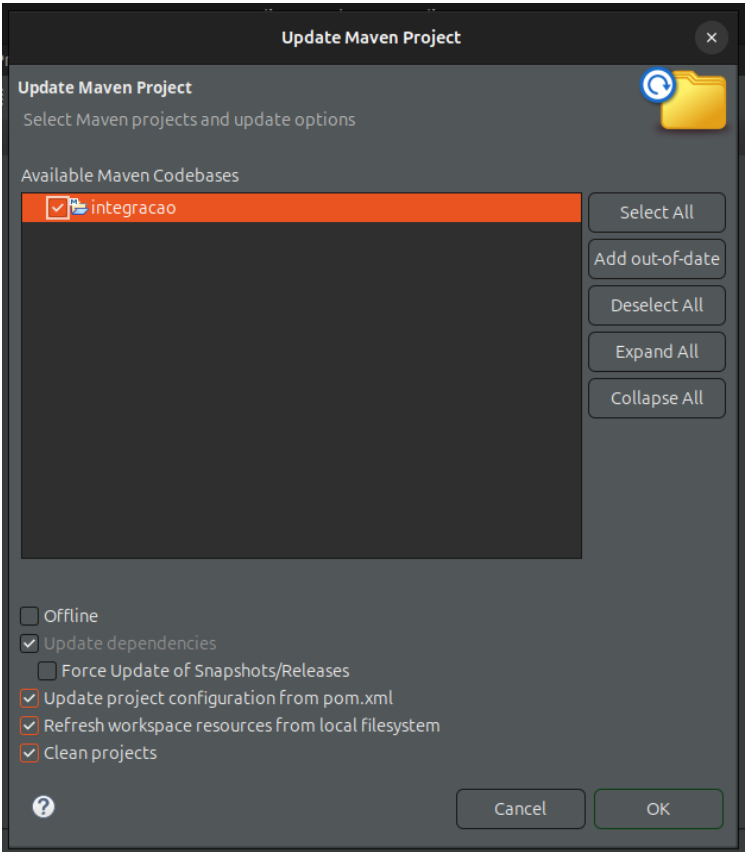
5. Print os resultados com Maven package

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 7.813 s  
[INFO] Finished at: 2024-09-05T21:57:10-03:00  
[INFO] -----  
julio-manoel@julio-manoel:~/GitHub/Pessoal/QA/integracao-main$ mvn package
```

6. Print os resultados com Maven install

```
[INFO] --- install:3.1.1:install (default-install) @ integracao ---  
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.5.0/plexus-utils-3.5.0.jar  
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.5.0/plexus-utils-3.5.0.jar (2  
[INFO] Installing /home/julio-manoel/GitHub/Pessoal/QA/integracao-main/pom.xml to /home/julio-manoel/.m2/repository/valueprojects/int  
[INFO] Installing /home/julio-manoel/GitHub/Pessoal/QA/integracao-main/target/integracao.jar to /home/julio-manoel/.m2/repository/val  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 6.469 s  
[INFO] Finished at: 2024-09-05T21:56:13-03:00  
[INFO] -----
```

7. Realize um Update Project. Qual a finalidade e quando recomendado?



A finalidade principal do Update Project, é atualizar as dependências do projeto, realizando a configuração é possível forçar a atualização dos pacotes, limpar a pasta target, atualizar o pom.xml e reatualizar todo o workspace dessa forma podendo solucionar alguns problemas de incompatibilidade e versões.

- 8. Print resultados com Jacoco, discutindo a complexidade ciclomática e cobertura dos testes.

Integracao > Integracao > Aluno

Sessions

Aluno

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
getFrequencia()	0%	n/a	1	1	1	1	1
getNota1()	0%	n/a	1	1	1	1	1
getNota2()	0%	n/a	1	1	1	1	1
calcularAprovacao()	100%	100%	0	5	0	10	0
setFrequencia(int)	100%	n/a	0	1	0	2	0
setNota1(float)	100%	n/a	0	1	0	2	0
setNota2(float)	100%	n/a	0	1	0	2	0
setNotaFinal(float)	100%	n/a	0	1	0	2	0
Aluno()	100%	n/a	0	1	0	1	0
Total	9 of 67	86%	0 of 8	100%	3	13	3

A complexidade ciclomática é uma métrica utilizada na área de desenvolvimento de softwares, que indica a complexidade do código com base em suas estruturas de controle (condições e loops). A cobertura de testes é uma métrica usada para avaliar a qualidade dos testes automatizados.

- 9. Print os resultados com PMD.

Resultados do PMD

O seguinte documento contém os resultados do [PMD](#) 6.21.0.

O PMD não encontrou problemas no seu código fonte.

Copyright © 2024. All rights reserved.

Análise DEVOPS (0,25)

1. Maven test deveria estar em uma esteira “DEV” ou “HOMOLOGACÃO”.

DEV

2. Maven install deveria ser um comando executado em ambiente “HOMOLOGACÃO”
OU PRODUÇÃO?

HOMOLOGACÃO


3. Qual a finalidade da pasta “Target” para DEVOPS?

No contexto de DevOps, a pasta target é fundamental, pois armazena os artefatos gerados durante os processos de build e teste, como arquivos compilados, pacotes finais, relatórios de testes, e logs.

Novos Casos de Testes (0,25)

Faça rodar estes dois novos testes (sugeridos pelo ChatGPT) a partir do código de teste:

java

 Copy code

```
@Test
public void testCalcularAprovacao_AprovacaoLimiteFrequencia(){
    aluno = new Aluno();
    aluno.setFrequencia(75);
    aluno.setNota1(70);
    aluno.setNota2(70);
    assertEquals(true, aluno.calcularAprovacao());
}

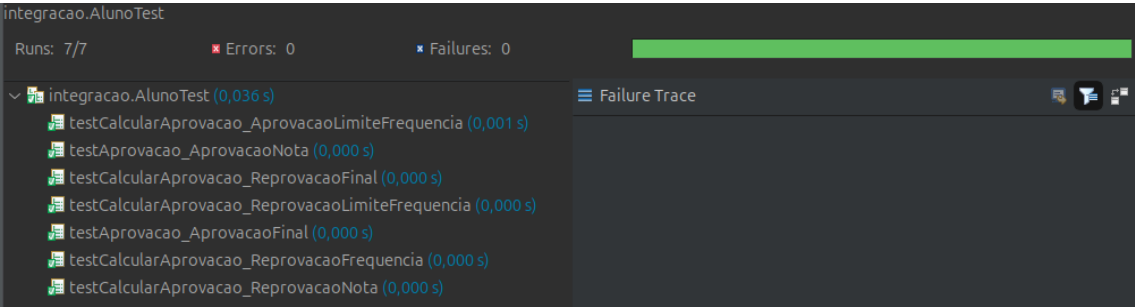
@Test
public void testCalcularAprovacao_ReprovacaoLimiteFrequencia(){
    aluno = new Aluno();
    aluno.setFrequencia(74.9);
    aluno.setNota1(70);
    aluno.setNota2(70);
    assertEquals(false, aluno.calcularAprovacao());
}
```

Obs: O metodo testCalcularAprovacao_ReprovacaoLimiteFrequencia da erro pois setFrequencia esperar um int e não float, por esse motivo foi utilizado 74 ao invés de 74.9

1. Melhorou a cobertura de testes?

Não pois já estava sendo testado isso anteriormente outros métodos sendo eles o testCalcularAprovacao_ReprovacaoFrequencia que já aplicava 74 na frequência limite para reprovação e as demais aplicavam 75 na frequência para aprovação assim testando os limites consecutivamente

2. Mostrar os resultados do teste rodando via JUNIT e Jacoco.

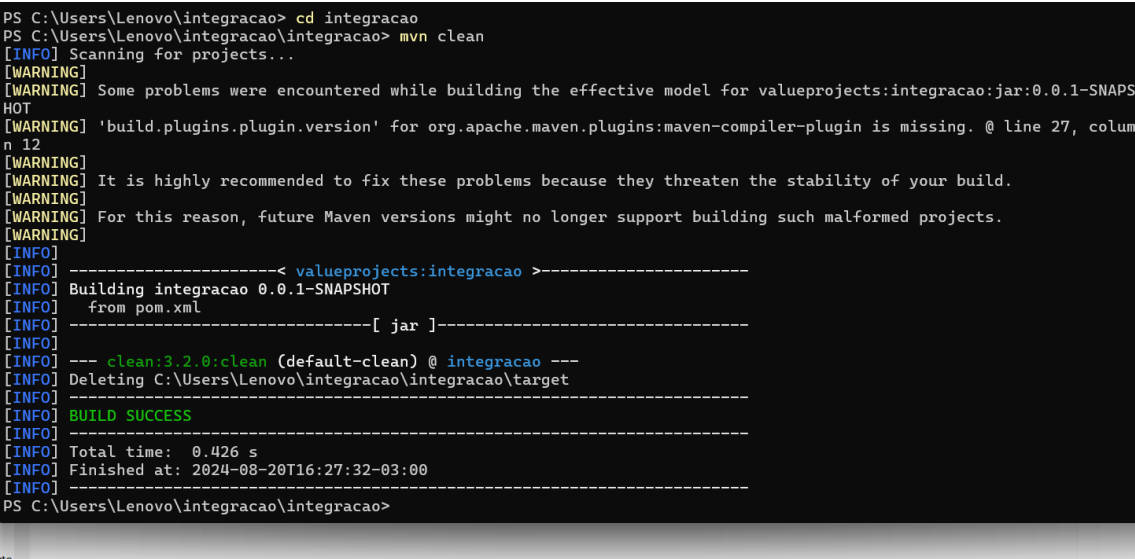


Aluno

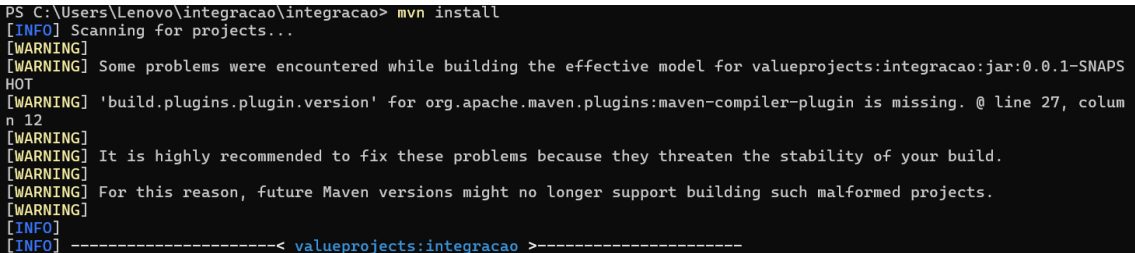
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getFrequencia()	1	0%	n/a	n/a	1	1	1	1	1	1
getNota1()	1	0%	n/a	n/a	1	1	1	1	1	1
getNota2()	1	0%	n/a	n/a	1	1	1	1	1	1
calcularAprovacao()	0	100%	0	100%	0	5	0	6	0	1
setFrequencia(int)	0	100%	n/a	n/a	0	1	0	2	0	1
setNota1(float)	0	100%	n/a	n/a	0	1	0	2	0	1
setNota2(float)	0	100%	n/a	n/a	0	1	0	2	0	1
setNotaFinal(float)	0	100%	n/a	n/a	0	1	0	2	0	1
Aluno()	0	100%	n/a	n/a	0	1	0	1	0	1
Total	9 of 67	86%	0 of 8	100%	3	13	3	18	3	9

Discuta os resultados em linha de comando (0,10)

mvn clean



mvn install



Verificar a pasta TARGET:


```
Diretório: C:\Users\Lenovo\integracao\integracao\target

ode                LastWriteTime                Length Name
----                -
----- 20/08/2024 16:28 classes
----- 20/08/2024 16:28 generated-sources
----- 20/08/2024 16:28 generated-test-sources
----- 20/08/2024 16:28 maven-archiver
----- 20/08/2024 16:28 maven-status
----- 20/08/2024 16:28 pmd
----- 20/08/2024 16:28 site
----- 20/08/2024 16:28 surefire-reports
----- 20/08/2024 16:28 test-classes
a----- 20/08/2024 16:28 2550 integracao.jar
a----- 20/08/2024 16:28 11853 jacoco.exec
a----- 20/08/2024 16:28 330 pmd.xml

PS C:\Users\Lenovo\integracao\integracao\target> |
```

1. Qual arquivo mostra os resultados dos testes com JUNIT?

Os resultados dos testes JUNIT fica dentro da pasta surefire-reports

2. Qual arquivo é o executável e pronto para instalação em Docker ou Cloud computing?

Os arquivos executáveis são os que possui a extensão .jar nesse caso seria integracao.jar

3. Qual arquivo mostra a gestão de código?

Seria o arquivo pmd.html que fica localizado dentro da pasta site

4. Qual arquivo mostra a métricas de qualidade de software a partir dos testes?

Seria o arquivo index.html do jacoco que fica localizado dentro da pasta site/jacoco

MÉTODO AAA – Criar casos de testes

Verificar a imagem:

```
public class AlunoTest {  
  
    Aluno aluno;  
  
    @Test  
    public void testCalcularAprovacao_ReprovacaoFrequencia() {  
  
        aluno = new Aluno();  
        aluno.setFrequencia(74);  
        assertEquals(false, aluno.calcularAprovacao());  
    }  
  
    @Test  
    public void testCalcularAprovacao_ReprovacaoNota() {  
        aluno = new Aluno();  
        aluno.setFrequencia(75);  
        aluno.setNota1 (29);  
        aluno.setNota2 (30);  
        assertEquals(false, aluno.calcularAprovacao());  
    }  
  
    @Test  
    public void testAprovacao_AprovacaoNota() {  
        aluno = new Aluno();  
        aluno.setFrequencia(75);  
        aluno.setNota1 (70);  
        aluno.setNota2 (70);  
        assertEquals(true, aluno.calcularAprovacao());  
    }  
}
```

1. Identifique as partes conforme o método AAA. Comente o código e print aqui como resposta. Crie você um novo caso de testes e mostre a cobertura dos testes e possíveis contribuições. Printar os resultados **(0,25)**

O método AAA é dividido em três partes, a preparo do teste que seria a inicialização da classe aluno, setFrequencia e setNota, assim configurando todo o ambiente de teste para poder realizar o teste. A segunda etapa é a realização do test no aluno.calcularAprovação(). Por fim verificamos a saída do que testamos para verificar se sua funcionalidade está sendo correta.

```
73 ● @Test  
74     public void testGetFrequencia() {  
75         aluno = new Aluno();  
76         aluno.setFrequencia(75);  
77         assertEquals(75, aluno.getFrequencia());  
78     }  
79  
80 ● @Test  
81     public void testGetNota1() {  
82         aluno = new Aluno();  
83         aluno.setNota1(75);  
84         assertEquals(75, aluno.getNota1(), 0.001);  
85     }  
86  
87 ● @Test  
88     public void testGetNota2() {  
89         aluno = new Aluno();  
90         aluno.setNota2(70);  
91         assertEquals(70, aluno.getNota2(), 0.001);  
92     }
```

Partir da análise realizado no relatório feito pelo Jacoco é possível compreender o que não estava sendo coberto pelos testes. Dessa forma foi feito os três métodos para que seja possível ter total cobertura dos testes.

Aluno

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● calcularAprovacao()	<div><div></div></div>	100%	<div><div></div></div>	100%	0	5	0	6	0	1
● setFrequencia(int)	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
● setNota1(float)	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
● setNota2(float)	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
● setNotaFinal(float)	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
● Aluno()	<div><div></div></div>	100%	n/a	n/a	0	1	0	1	0	1
● getFrequencia()	<div><div></div></div>	100%	n/a	n/a	0	1	0	1	0	1
● getNota1()	<div><div></div></div>	100%	n/a	n/a	0	1	0	1	0	1
● getNota2()	<div><div></div></div>	100%	n/a	n/a	0	1	0	1	0	1
Total	0 of 67	100%	0 of 8	100%	0	13	0	18	0	9