

# SIN5022 – Teste de Software

## Escola de Artes Ciências e Humanidades

### Evo Suite 1.6 – Geração automática de testes

---

Adilson Khouri

Nícolás Hamparsomian

# Evo Suite – O que é?

- Evo Suite é uma ferramenta responsável pela geração automatizada de testes de unidade para classes Java.
- A abordagem utilizada gera testes utilizando algoritmos genéticos para aleatoriamente construir casos de testes de modo que se busquem os que possuem a melhor cobertura do código-fonte.

# Evo Suite – Sobre

- Nasceu em 2010 como resultado de um projeto de pesquisa conduzido por Dr. Gordon Fraser e Dr. Andrea Arcuri.
- Projeto open-source disponibilizado sob a licença LGPL.
- O projeto é referência na geração automatizada de testes, sendo apoiado pela Google e pela Yourkit.

Como funciona a geração  
automática de testes  
da EvoSuite ?

# Algoritmos Genéticos



# O que é um Algoritmo Genético (GA)?

- São algoritmos de busca heurístico que podem ser aplicados em diversas áreas como: otimização restrita/irrestrita, programação não linear, programação estocástica entre outras.
- Possuem inspiração na área da biologia genética.

# Definições

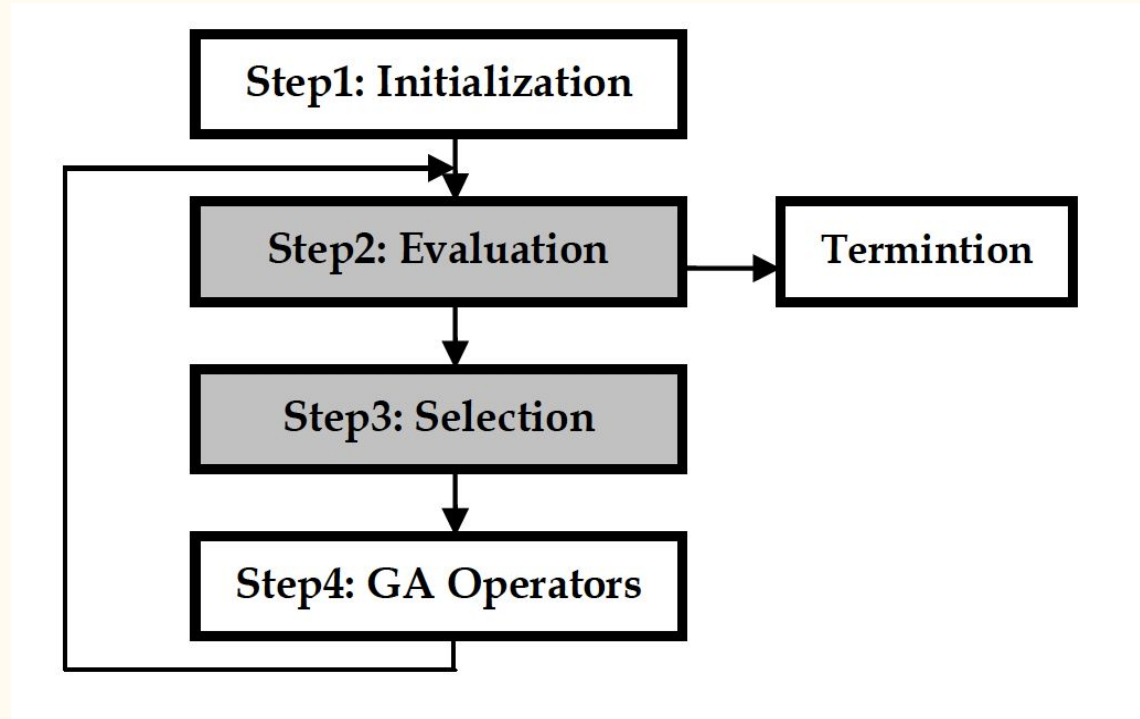
- Cromossomo: Representação da solução do problema.
- População: Conjunto de Cromossomos.
- Função de fitness: Função para escorar a população e definir os melhores cromossomos, que serão usados na próxima geração.
- Seleção: Escolha dos melhores cromossomos, o algoritmo clássico nessa etapa é denominado 'roleta' onde cada cromossomo recebe um escore (pela função fitness) que será sua probabilidade de ser selecionado para a nova população.

# Definições

- Reprodução: Escolha de dois cromossomos, em seguida pode-se aplicar: Cross over ou uma mutação.
- Cross over: formar um novo cromossomo usando partes de outros cromossomos.
- Mutação: Alterar o valor de um cromossomo com uma probabilidade baixa.



# Processso



# Exemplo de algoritmo genética na prática

- Um exemplo em vídeo por ser visualizado aqui: [Exemplo GA](#)
- A idéia desse GA é encontrar os parâmetros ótimos de dimensões do carrinho, como por exemplo, tamanho das rodas, posição das rodas, tamanho do 'corpo' do carrinho, quantidade de caixas sobre o carrinho, posição de caixas sobre o carrinho tal que possibilite chegar mais longe na pista de corrida.
- **O EvoSuite usa um GA para definir os testes!**

# Representação usada na EvoSuite



# Modelagem do problema

- Primeiro será explicado como é a representação do problema em termos de algoritmo genético.
- O Cromossomo é a *test suite* com todos os testes.
- Dentro da *test suite* temos os casos de teste individuais que são formados por conjuntos de comandos. Os autores separam em quatro tipos de comandos:
  - Primitivos: Valor numérico
  - Construtor: Geram novas instâncias de uma classe
  - Field: Acessam variáveis públicas de um objeto
  - Method: Invocação de métodos

# Modelagem do problema

- Função Fitness:

$$\text{Fitness} = |M| - |M_T| + \sum_{b_k \in B} d(b_k, T)$$

$$d(b, T) = \begin{cases} 0 & \text{if the branch has been covered} \\ v(d_{\min}(b, T)) & \text{if the predicate has been executed at least twice} \\ 1 & \text{otherwise} \end{cases}$$

- Onde  $|M|$  são os métodos a serem executados,  $|M_T|$  são os métodos já executados pelo teste, 'd' é a distância de branch e 'v' é uma função para normalizar 'd' entre  $[0,1]$

# Modelagem do problema

- **Cross Over:** Gera uma nova geração usando dois pais (P1 e P2), para tal, gera um número aleatório  $x[0,1]$ . Em seguida os  $x$  primeiros testes de P1 e  $1-x$  últimos testes de P2 serão usados para gerar o primeiro cromossomo da nova geração. Para gerar o segundo cromossomo são usados os últimos  $1-x$  testes de P1 e os  $x$  primeiros de P2. Os testes unitários sempre são independentes entre eles!

# Modelagem do problema

- **Mutação:** Quando um cromossomo sofre mutação, todos os testes de caso unitários dentro da suíte vão sofrer mutação com probabilidade  $1/|T|$ . Em outras palavras, aproximadamente 1 a cada sorteio.
- Quando uma suíte de teste sofre mutação podem ocorrer, com probabilidade  $1/3$  cada, as seguintes operações: i) remoção; ii) atualização; iii) inserção.
- **Remoção:** cada comando do teste unitário selecionado para ser removido, pode ser removido com probabilidade  $1/n$  (1 teste unitário é formado por diversos comandos), depois da remoção é adicionado outro comando com mesmo tipo para todas os comandos que apontavam para o valor removido.

# Modelagem do problema

- Alteração: um comando do teste unitário é selecionado para ser alterado, se for do tipo primitivo troca-se por um valor aleatório. Se não for do tipo primitivo então ocorre uma escolha aleatória de um comando do mesmo tipo para substituir
- Obs: A diferença entre Alteração e remoção é que na alteração há uma troca por algum valor aleatório na remoção não. Além disso, não são removidas/alteradas as referências do objeto alterado.
- Inserção: são acrescentados comandos aleatório com probabilidade  $x$ , em seguida  $x^2$ ,  $x^3$  até que não seja acrescentado mais nenhum
- Para avaliar o fitness temos que rodar todos os casos de teste da suíte e avaliar as informações de branch.



# Execução

- Os casos de teste são inicializados de forma aleatória para dar início ao processo de evolução.
- Cálculo do fitness da solução atual
- Seleção dos melhores candidatos para gerar uma nova população
- Mutação e Crossover para ter alteração genética...
  - volta para o cálculo do fitness e fica em loop por um determinado tempo (que pode ser definido via parâmetro de execução) ou até realizar a cobertura da classe sendo testada.

Como utilizar a EvoSuite para automaticamente gerar casos de testes que realizam a cobertura do meu código-fonte?

# Tutorial da EvoSuite

## Integração com o Maven

# Configuração do ambiente de desenvolvimento



# Tecnologias necessárias

Configurar previamente a sua máquina e instalar os seguintes programas:

- JDK 1.8 ou superior
- Maven 3.1 ou superior  
(Disponível por default na instalação do IntelliJ)
- IntelliJ IDEA (IDE)

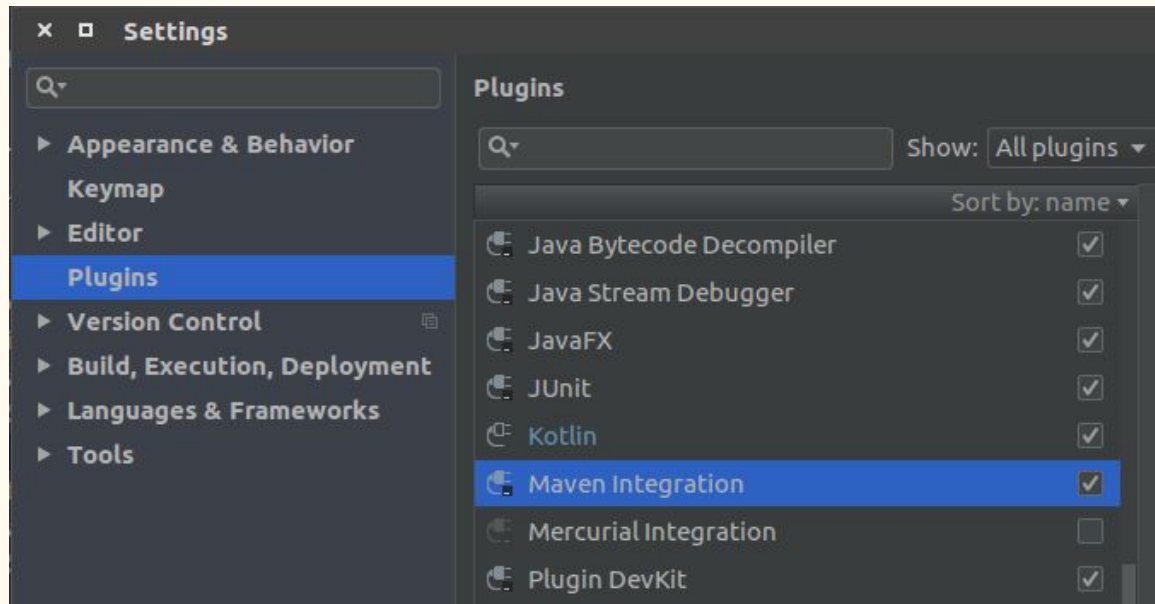
Obs: IDE não é obrigatória, iremos utilizar apenas para facilitar a geração da classe e execução dos testes

# Como criar um projeto Maven no IntelliJ.

—

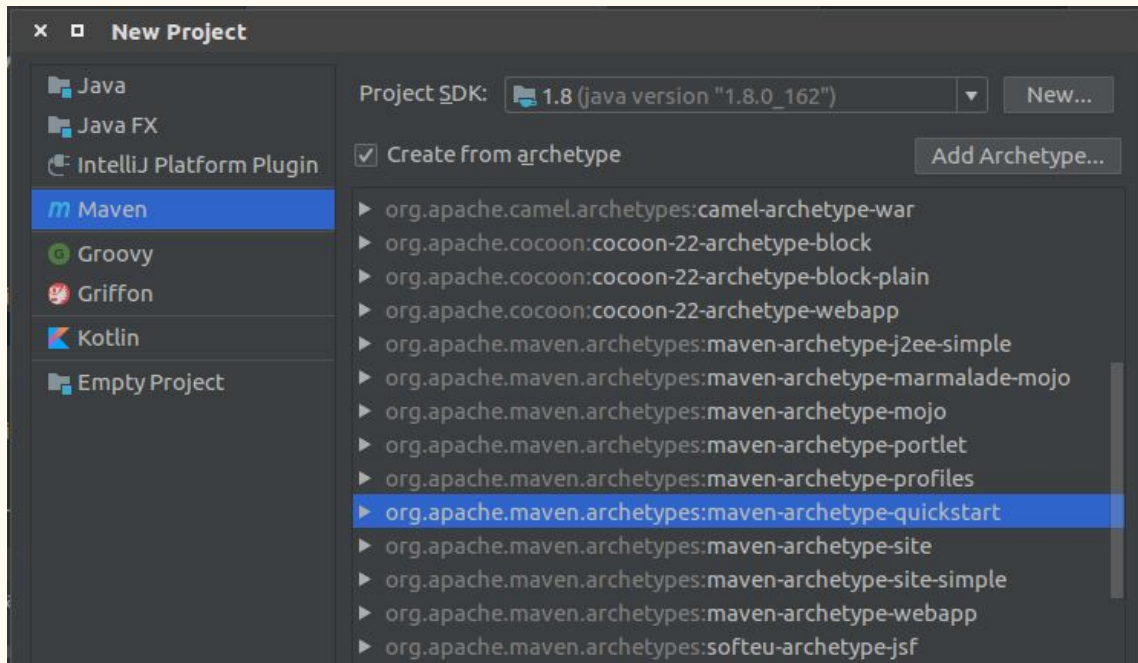
# IntelliJ - Verificar plugin com Maven

- File → Settings...
- Selecionar opção “Plugins”
- Verificar se a opção “Maven Integration” se encontra habilitada



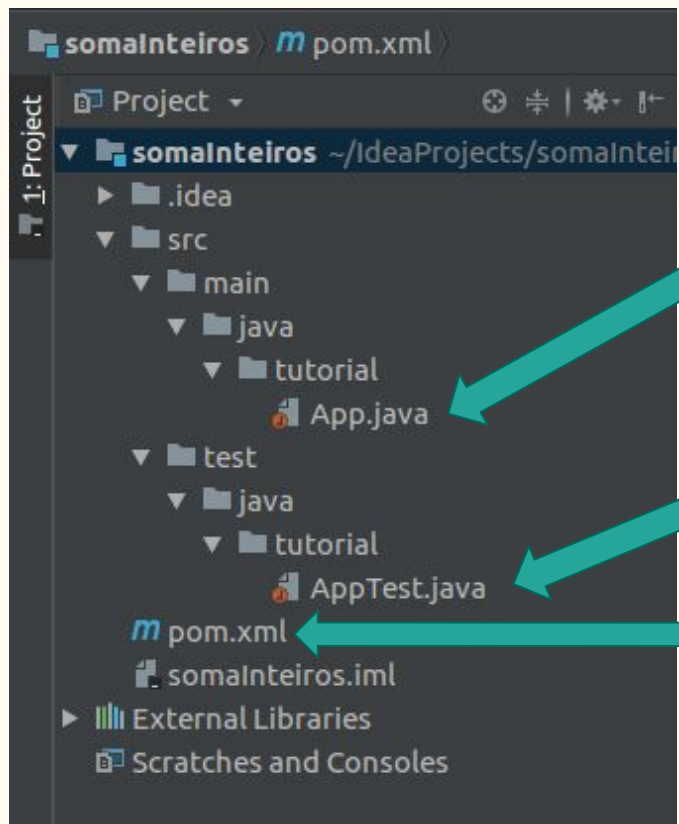
# IntelliJ - Novo projeto Maven

- File → New → Project...
- Selecionar opção “Maven” no menu de projetos
- Criar o projeto à partir de um protótipo. Selecionar o modelo “quickstart” na lista de protótipos disponíveis





# IntelliJ - Estrutura do projeto criado



Código-fonte

Testes de unidade

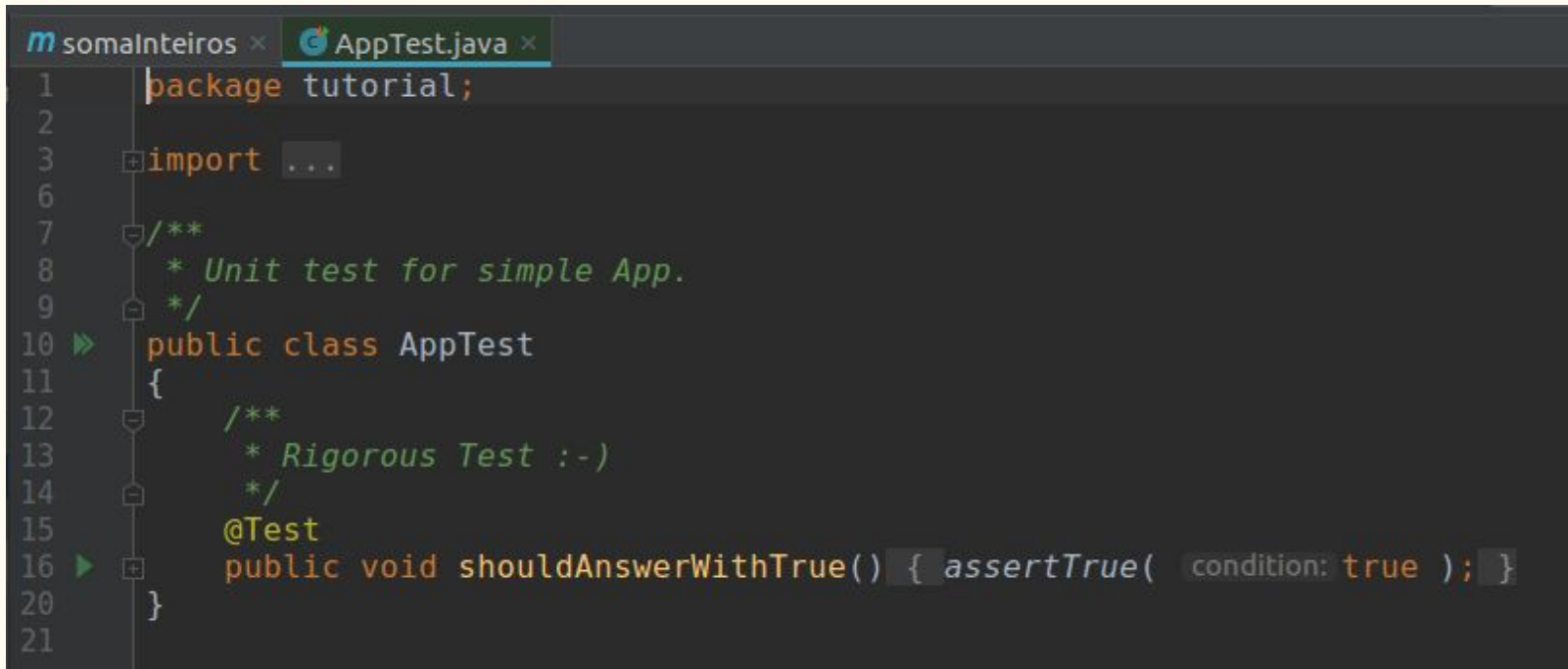
Arquivo de configuração de dependências do projeto

Explorando o projeto criado.

—

# IntelliJ - Classe de teste gerada

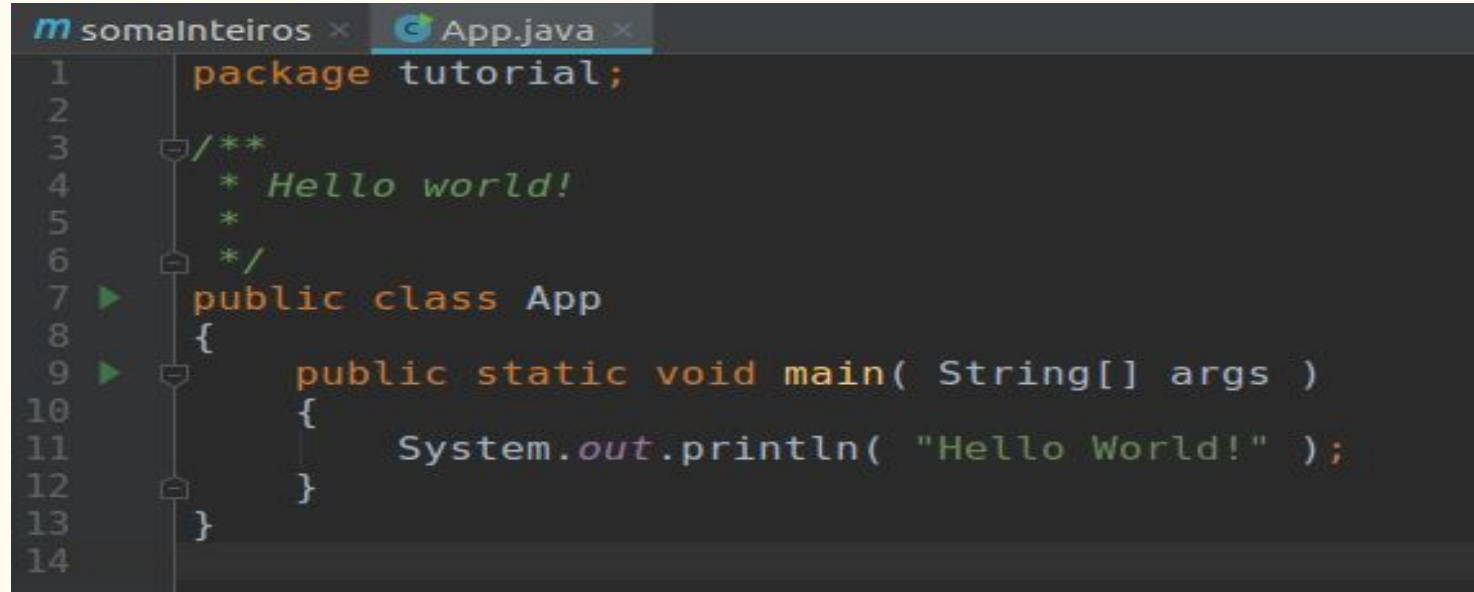
- Classe de teste auto gerada pelo Maven para indicar que a configuração deu certo:



```
m somaInteiros x AppTest.java x
1 package tutorial;
2
3 import ...
4
5
6
7 /**
8  * Unit test for simple App.
9  */
10 public class AppTest
11 {
12     /**
13      * Rigorous Test :-)
14      */
15     @Test
16     public void shouldAnswerWithTrue() { assertTrue( condition: true ); }
17
18
19
20 }
21
```

# Projeto Maven - Classe de aplicação gerada

- Classe de aplicação gerada automaticamente pelo Maven para indicar que a configuração ocorreu com sucesso:



```
m somaInteiros x App.java x
1 package tutorial;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hello World!" );
12    }
13 }
14
```

# Projeto Maven - Classe de aplicação gerada

- Neste exemplo, iremos implementar um método se soma na classe de aplicação auto gerada pelo Maven “App.java”.
- Ao implementarmos este método, a intenção é que os respectivos testes sejam gerados automaticamente pelo ferramenta EvoSuite que iremos configurar.

```
public static int soma( int numero1, int numero2 )  
{  
    return numero1 + numero2;  
}
```

# Projeto Maven - Arquivo POM.xml

- Por fim o arquivo mais importante de um projeto Maven, o arquivo pom.xml (Project Object Model)
- Um POM possui as informações básicas de um projeto, bem como as diretivas de como o artefato final deste projeto deve ser construído.

```
m somaInteiros x App.java x
1      <?xml version="1.0" encoding="UTF-8"?>
2
3      <project xmlns="http://maven.apache.org/POM/4.0.0" x
4          xsi:schemaLocation="http://maven.apache.org/POM/4.
5              <modelVersion>4.0.0</modelVersion>
6
7              <groupId>tutorial</groupId>
8              <artifactId>somaInteiros</artifactId>
9              <version>1.0-SNAPSHOT</version>
10
11              <name>somaInteiros</name>
12              <!-- FIXME change it to the project's website -->
13              <url>http://www.example.com</url>
14
```

# Configurando o arquivo POM.xml

—

# Configurando o arquivo POM.xml

- Definir versão do Java como 1.8
- Definir Versão do JUnit como 4.12

```
<properties>  
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
  <maven.compiler.source>1.8</maven.compiler.source>  
  <maven.compiler.target>1.8</maven.compiler.target>  
</properties>
```

```
<dependencies>  
  <dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>4.12</version>  
    <scope>test</scope>  
  </dependency>
```



# Configurando o arquivo POM.xml

- Adicionar o EvoSuite 1.6 como dependência do projeto dentro do escopo da Tag `<dependencies>`

```
<dependency>
  <groupId>org.evosuite</groupId>
  <artifactId>evosuite-standalone-runtime</artifactId>
  <version>1.0.6</version>
  <scope>test</scope>
</dependency>

</dependencies>
```

# Configurando o arquivo POM.xml

- Definir o EvoSuite como um plugin que faz parte do Build do Projeto dentro do escopo da Tag `<plugins>`

```
<plugin>
  <groupId>org.evosuite.plugins</groupId>
  <artifactId>evosuite-maven-plugin</artifactId>
  <version>1.0.6</version>
</plugin>

</plugins>
```

# Configurando o arquivo POM.xml

- Plugin do EvoSuite não está disponível na central de plugins do Maven. É necessário informar o endereço do repositório oficial da ferramenta.

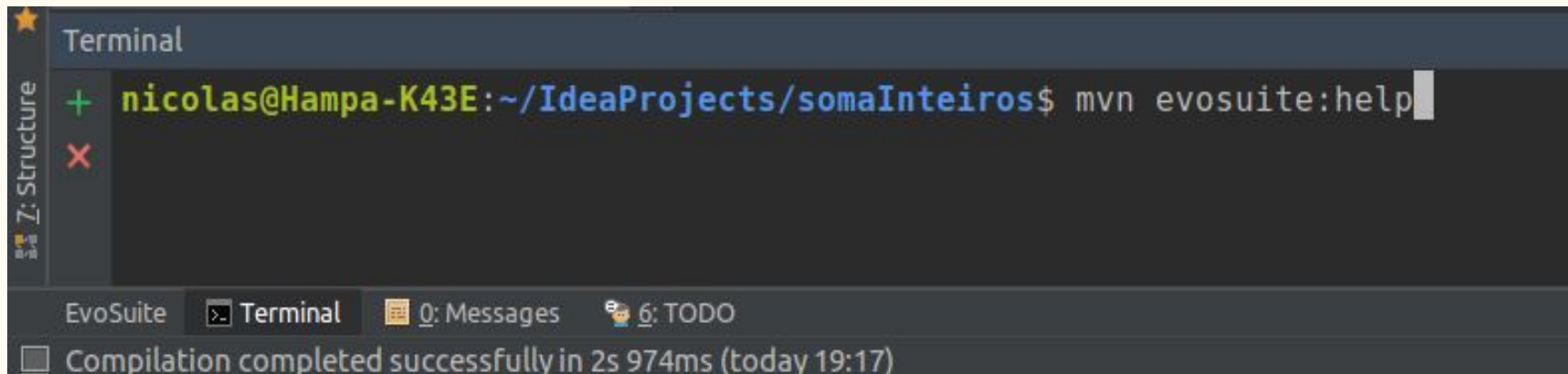
```
<pluginRepositories>
  <pluginRepository>
    <id>EvoSuite</id>
    <name>EvoSuite Repository</name>
    <url>http://www.evosuite.org/m2</url>
  </pluginRepository>
</pluginRepositories>
```

Geração de testes utilizando  
EvoSuite.



# Gerando testes de unidade para o projeto

- Uma vez que a configuração do projeto esteja realizada, é necessário que o EvoSuite e suas respectivas dependências sejam incorporadas ao projeto.
- Para tal, abrir o terminal da IDE IntelliJ e digitar algum comando Maven que possua o EvoSuite como goal. Neste caso usamos: “**mvn evosuite:help**”

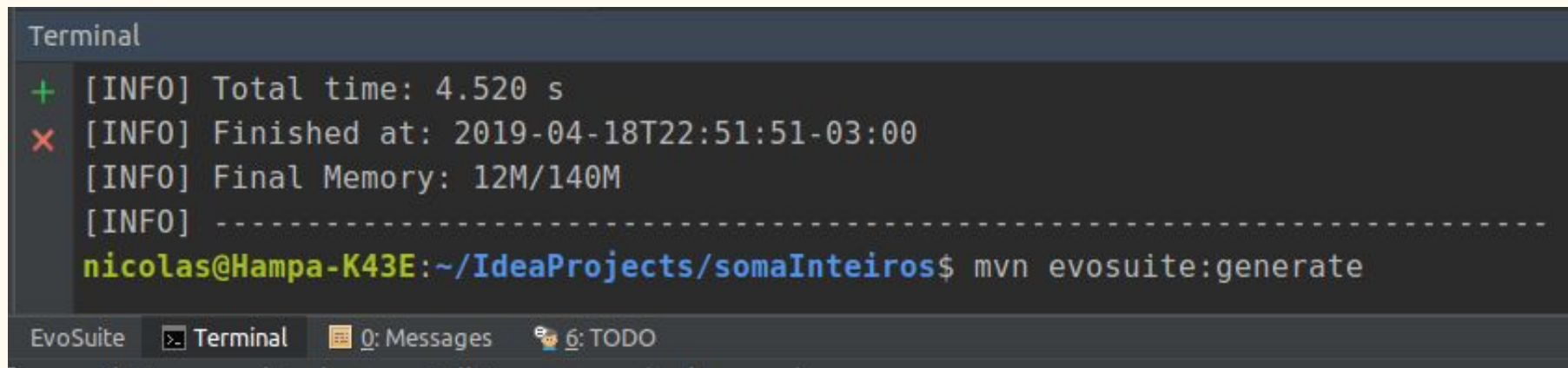


The screenshot shows the IntelliJ IDE interface. On the left, the 'Structure' tool window is visible with a star icon. The main terminal window is titled 'Terminal' and shows the command prompt 'nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros\$' followed by the command 'mvn evosuite:help'. The terminal output is empty. At the bottom of the IDE, the status bar shows 'EvoSuite' as the active tool, 'Terminal' as the active window, '0: Messages', and '6: TODO'. A notification bar at the very bottom states 'Compilation completed successfully in 2s 974ms (today 19:17)'.

```
Terminal
+ nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$ mvn evosuite:help
X
Z: Structure
EvoSuite  Terminal  0: Messages  6: TODO
Compilation completed successfully in 2s 974ms (today 19:17)
```

# Gerando testes de unidade para o projeto

- Agora estamos com tudo pronto para gerarmos os testes de unidade de nosso projeto. Para tal, basta digitar o comando Maven “**mvn evosuite:generate**”

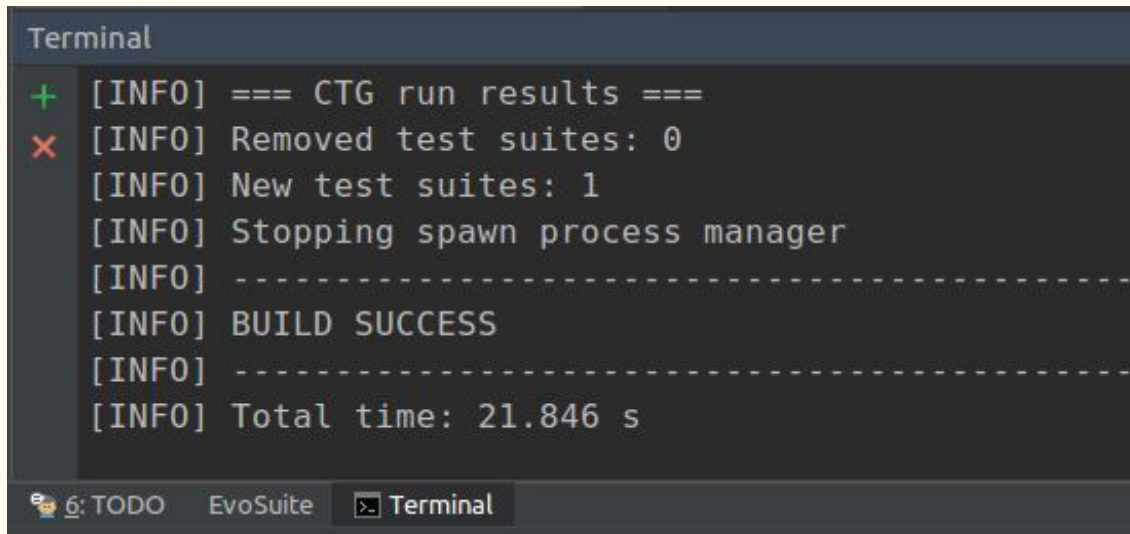


```
Terminal
+ [INFO] Total time: 4.520 s
x [INFO] Finished at: 2019-04-18T22:51:51-03:00
[INFO] Final Memory: 12M/140M
[INFO] -----
nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$ mvn evosuite:generate
```

EvoSuite   ▢ Terminal   0: Messages   6: TODO

# Gerando testes de unidade para o projeto

- Terminal irá exibir mensagem informando que os testes foram gerados com sucesso.
- Na mensagem será informado quantos conjuntos de testes foram criados automaticamente



The image shows a terminal window with a dark background and light-colored text. The title bar of the terminal is labeled "Terminal". The output text is as follows:

```
+ [INFO] === CTG run results ===  
x [INFO] Removed test suites: 0  
[INFO] New test suites: 1  
[INFO] Stopping spawn process manager  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 21.846 s
```

At the bottom of the terminal window, there is a status bar with three tabs: "6: TODO", "EvoSuite", and "Terminal". The "Terminal" tab is currently selected and highlighted.

# Gerando testes de unidade para o projeto

- Os testes criados, por padrão, **não** ficam disponíveis no diretório de testes. A ideia é que o desenvolvedor primeiro avalie a cobertura dos testes gerados.
- Os testes podem ser encontrados no diretório oculto “**.evosuite/best-tests**” localizado na raiz do projeto:

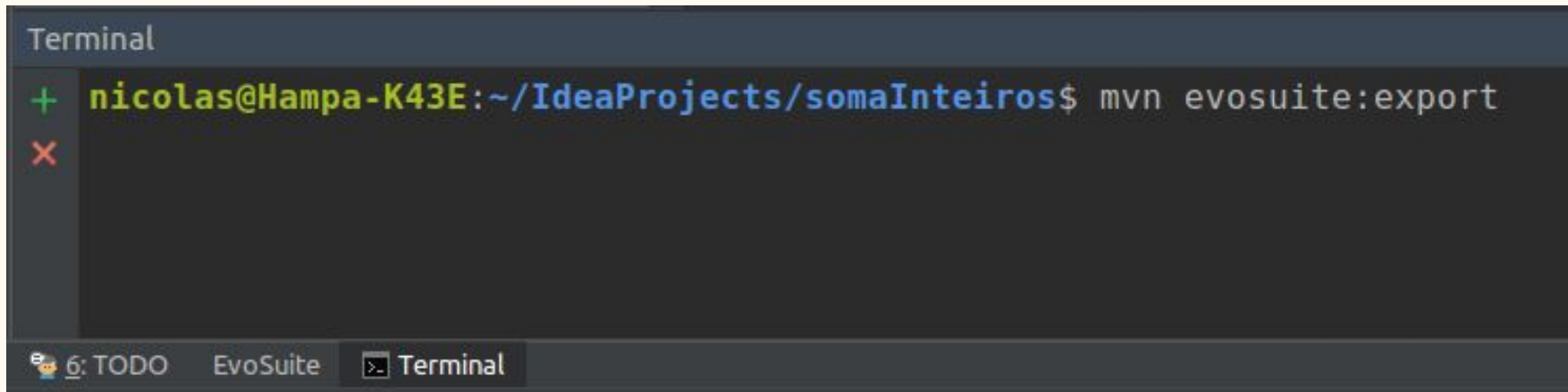
```
Terminal
+ nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$ ls -l .evosuite/best-tests/tutorial/
X total 12
-rw-rw-r-- 1 nicolas nicolas 1265 Abr 18 23:33 App_ESTest.java
-rw-rw-r-- 1 nicolas nicolas 4102 Abr 18 23:33 App_ESTest_scaffolding.java
nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$
```

6: TODO   EvoSuite   Terminal



# Gerando testes de unidade para o projeto

- É necessário adicionar os testes gerados ao diretório de testes do nosso projeto. Para tal, digitar o comando Maven “**mvn evosuite:export**”

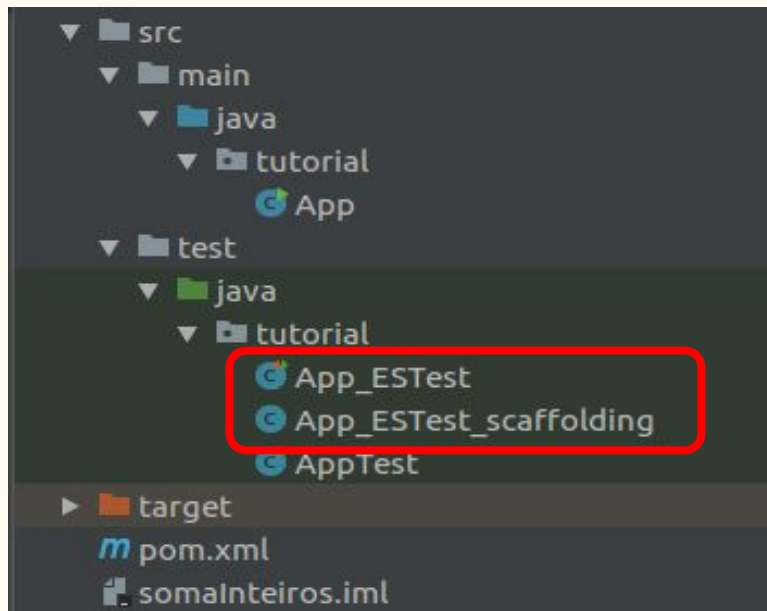


The image shows a terminal window with a dark background. The title bar at the top says "Terminal". The prompt is "nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros\$". The command "mvn evosuite:export" has been entered. To the left of the command line, there is a green plus sign and a red X icon. At the bottom of the window, there is a tab bar with three tabs: "6: TODO", "EvoSuite", and "Terminal". The "Terminal" tab is currently selected.

```
Terminal
+ nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$ mvn evosuite:export
X
6: TODO  EvoSuite  Terminal
```

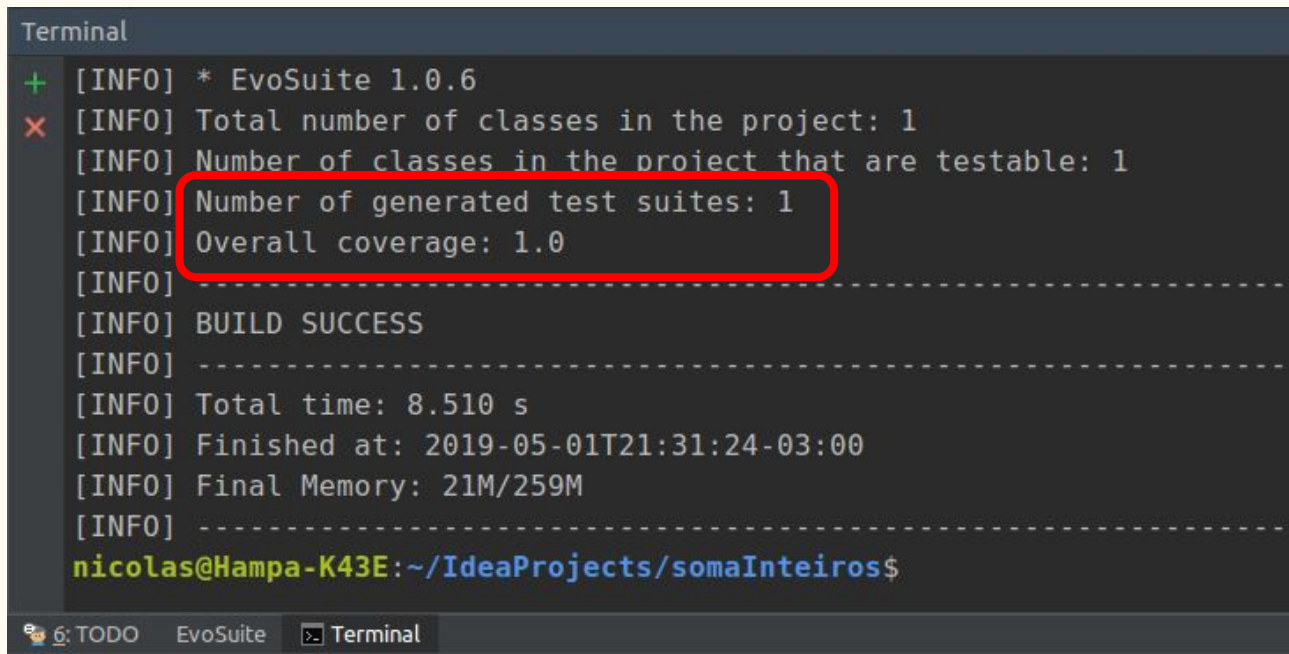
# Gerando testes de unidade para o projeto

- Será exibido no console uma mensagem de sucesso. Os testes agora estarão acessíveis à partir do diretório “test” do nosso projeto:



# Gerando testes de unidade para o projeto

- Para obter informações detalhadas sobre a cobertura do conjunto de testes recém gerado, digitar o comando “mvn evosuite:info”



```
Terminal
+ [INFO] * EvoSuite 1.0.6
x [INFO] Total number of classes in the project: 1
[INFO] Number of classes in the project that are testable: 1
[INFO] Number of generated test suites: 1
[INFO] Overall coverage: 1.0
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.510 s
[INFO] Finished at: 2019-05-01T21:31:24-03:00
[INFO] Final Memory: 21M/259M
[INFO] -----
nicolas@Hampa-K43E:~/IdeaProjects/somaInteiros$
```

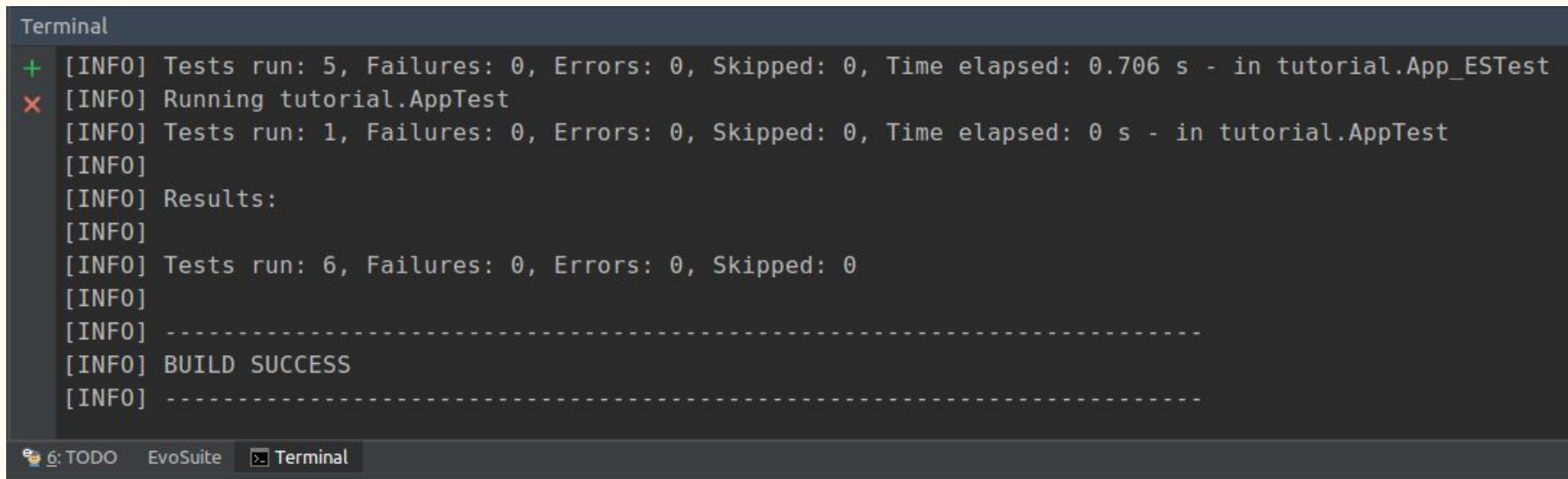
# Gerando testes de unidade para o projeto

- Ao abrir o arquivo “App\_ESTest”, é possível visualizar o conjunto de testes que foram criados automaticamente para a nossa classe:

```
@RunWith(EvoRunner.class) @EvoRunnerParameters(mockJVMNonDeterminism = true,  
public class App_ESTest extends App_ESTest_scaffolding {  
  
    @Test(timeout = 4000)  
    public void test0() throws Throwable {  
        int int0 = App.soma(0, 0);  
        assertEquals( expected: 0, int0);  
    }  
  
    @Test(timeout = 4000)  
    public void test1() throws Throwable {  
        int int0 = App.soma((-71), 197);  
        assertEquals( expected: 126, int0);  
    }  
}
```

# Gerando testes de unidade para o projeto

- Para finalizar, digitar o comando Maven “**mvn test**”. Os testes gerados automaticamente pelo EvoSuite serão executados e, ao final, será exibida uma mensagem de sucesso com a quantidade total de testes rodados:

A screenshot of a terminal window with a dark background. The title bar at the top says "Terminal". The terminal content shows the output of a Maven test command. It starts with a green plus icon and "[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.706 s - in tutorial.App\_ESTest". This is followed by a red cross icon and "[INFO] Running tutorial.AppTest". Then, another green plus icon and "[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in tutorial.AppTest". This is followed by "[INFO] Results:". Then, "[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0". Finally, "[INFO] BUILD SUCCESS" is shown, flanked by dashed lines. At the bottom of the terminal, there is a status bar with three tabs: "6: TODO", "EvoSuite", and "Terminal" (which is active and highlighted).

Personalizando a geração de testes.



# Personalizando Geração de Testes

- Abaixo um exemplo da utilização do goal “evosuite:generate” com exemplos de parâmetros para customizar a geração dos testes:

```
mvn -DmemoryInMB=2000 -Dcores=2 -Dsearch_budget=40  
evosuite:generate -Dcriterion=branch:line
```

- **-DmemoryInMB:** Quantidade máxima de memória que deverá ser alocada ao processo.
- **-Dcores:** Número de cores da CPU a ser utilizado.
- **-Dsearch\_budget:** Tempo limite em segundos para geração dos testes.

# Personalizando Geração de Testes

- Parâmetro **-Dcriterion** é utilizado para que possamos determinar quais os critérios de cobertura a serem utilizados durante a geração.
- Por default, **todos** os critérios são utilizados na geração dos testes
- No exemplo anterior definimos apenas os critérios “branch” (cobre as condições de verdadeiro/falso) e “line” (todas as linhas são os principais).



# Personalizando Geração de Testes

- Podemos usar outros valores para indicarmos quais são os nossos critérios de cobertura desejados:
  - **exception:** para cobrir todas as exceções
  - **weakmutation:** menor probabilidade de mutações
  - **method:** todos os métodos.

# Personalizando Geração de Testes

- Caso queira listar todas as possibilidades de parâmetros que podem ser utilizados para a geração automática de teste com a EvoSuite, digitar o seguinte comando no terminal:

```
mvn evosuite:help -Ddetail=true -Dgoal=generate
```