Apache NetBeans

Conhecendo um pouco da IDE

Professor: Edwar Saliba Júnior

Sumário

Apresentação:

1) Abra a IDE (*Integrated Development Environment*), ou seja, o Ambiente Integrado de Desenvolvimento conhecido como "Apache NetBeans". Você verá a IDE, como na imagem a seguir (Figura 1):

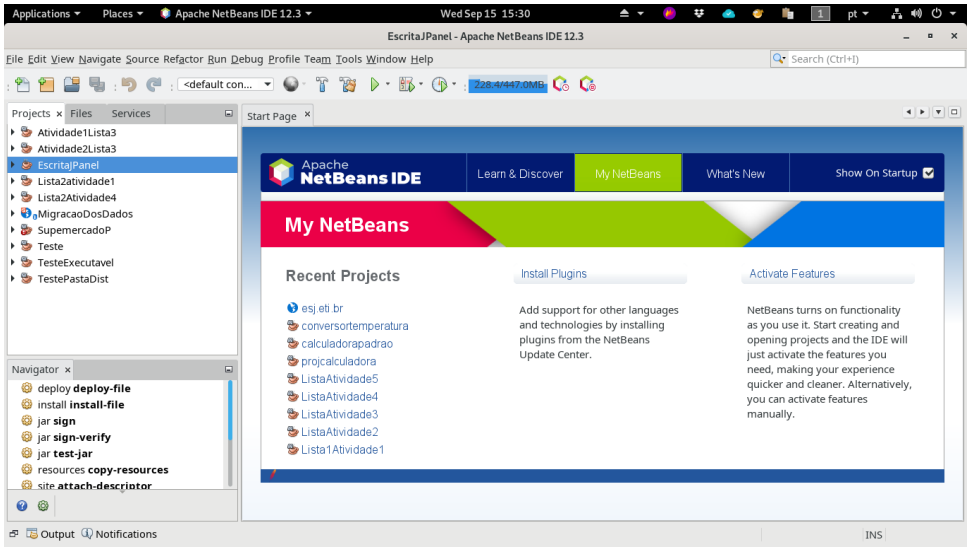

Figura 1: IDE Apache NetBeans

Criando Um Novo Projeto de Software:

2) vá em "*File* | *New Project*..." a seguinte tela aparecerá para você:

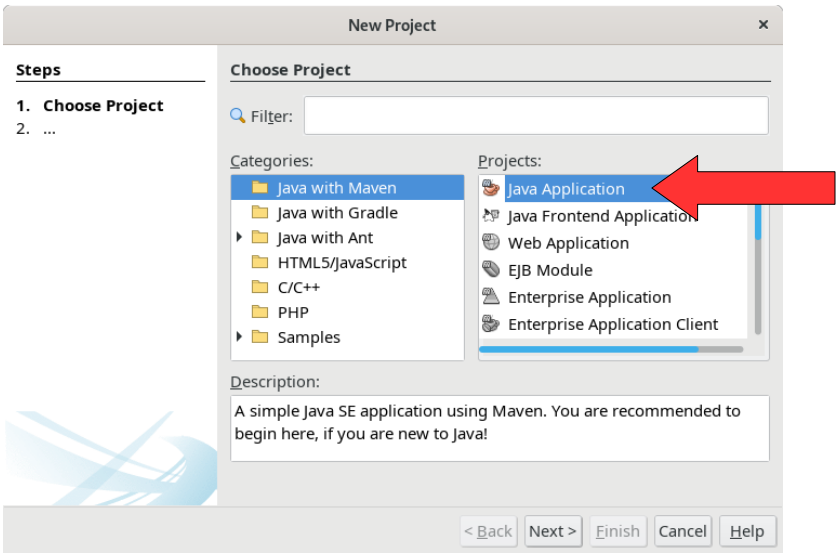

Figura 2: New Project

3) em *Categories* escolha a opção "*Java with Maven*" em *Projects* escolha a opção "*Java Application*". (Figura 2), e em seguida aperte o botão *Next*. A seguinte tela aparecerá para você (Figura 3):

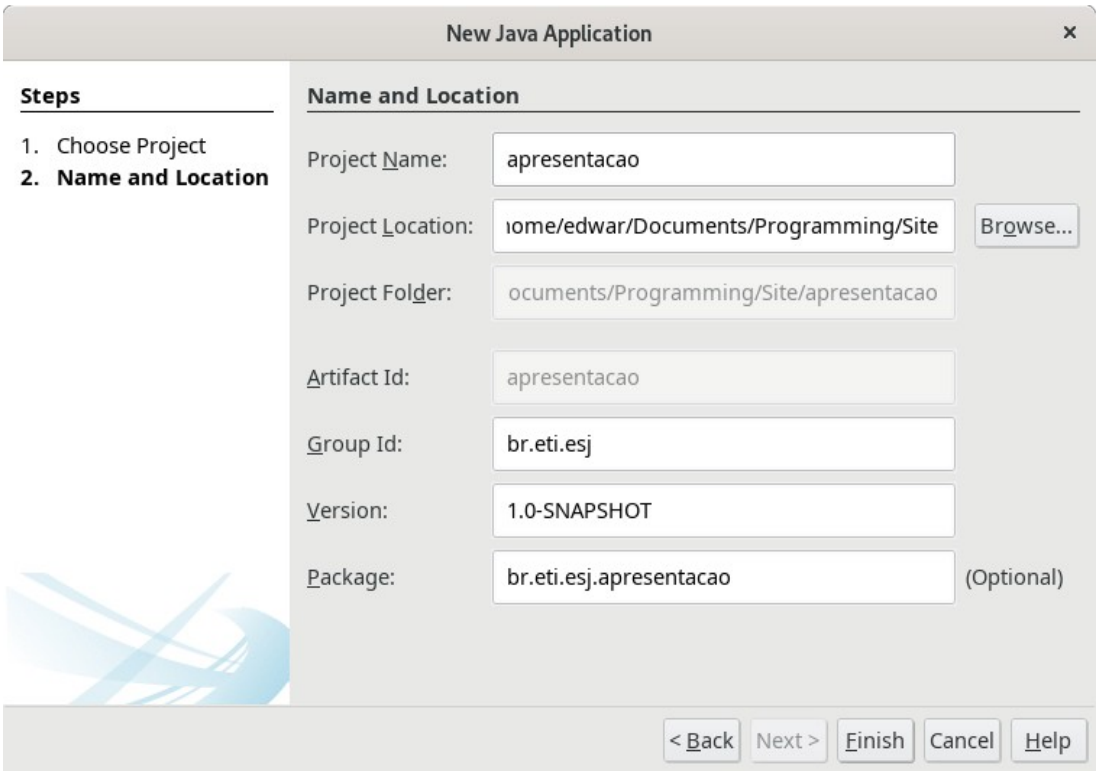

Figura 3: New Java Application

4) na tela apresentada na Figura 3, faça o seguinte:
   - dê um nome ao seu projeto, preenchendo o campo ***Project Name*** (para que você não tenha problemas com este tutorial mais adiante, sugere-se que você dê o nome deste primeiro projeto de: `apresentacao`). Lembre-se de não usar caracteres especiais e tampouco espaço em branco, caso opte por dar outro nome ao seu projeto,

- no campo ***Project Location***, você poderá escolher uma pasta (diretório), onde você deseja que seu projeto seja criado, para tanto use o botão "Browse…" (ao lado do campo),
- no campo ***Project Folder***, você verá o caminho completo que você escolheu, onde estará localizado seu novo projeto. Neste campo você poderá observar também, que será criada uma pasta com o nome que você deu ao seu projeto, e esta pasta será criada dentro da última pasta do cominho que você escolheu no campo *Project Location* (campo imediatamente acima do campo *Project Folder*),
- o campo ***Artefact Id*** é, da mesma forma que o campo *Project Folder*, é somente para leitura. Ou seja, você não poderá modificá-lo. E o valor nele descrito será usado pela IDE para controle interno,
- no campo ***Group Id*** costuma-se colocar a URL (*Uniform Resource Locator*) invertida da pessoa ou empresa. Caso você não tenha a sua, não se preocupe! Você poderá inventar um nome qualquer ou mesmo aceitar o que lhe for sugerido pela IDE, caso seja,
- o campo ***Version*** é utilizado para controle de versão, geralmente empresas produtoras de *software* o utilizam para controlar a versão de seus produtos e
- para finalizar temos o campo ***Package***, onde você poderá determinar o nome do primeiro "pacote" do projeto que você está criando;

5) tudo preenchido, então aperte o botão *Finish* (Figura 3). Será mostrado a você, uma tela com um novo projeto já iniciado (Figura 4);

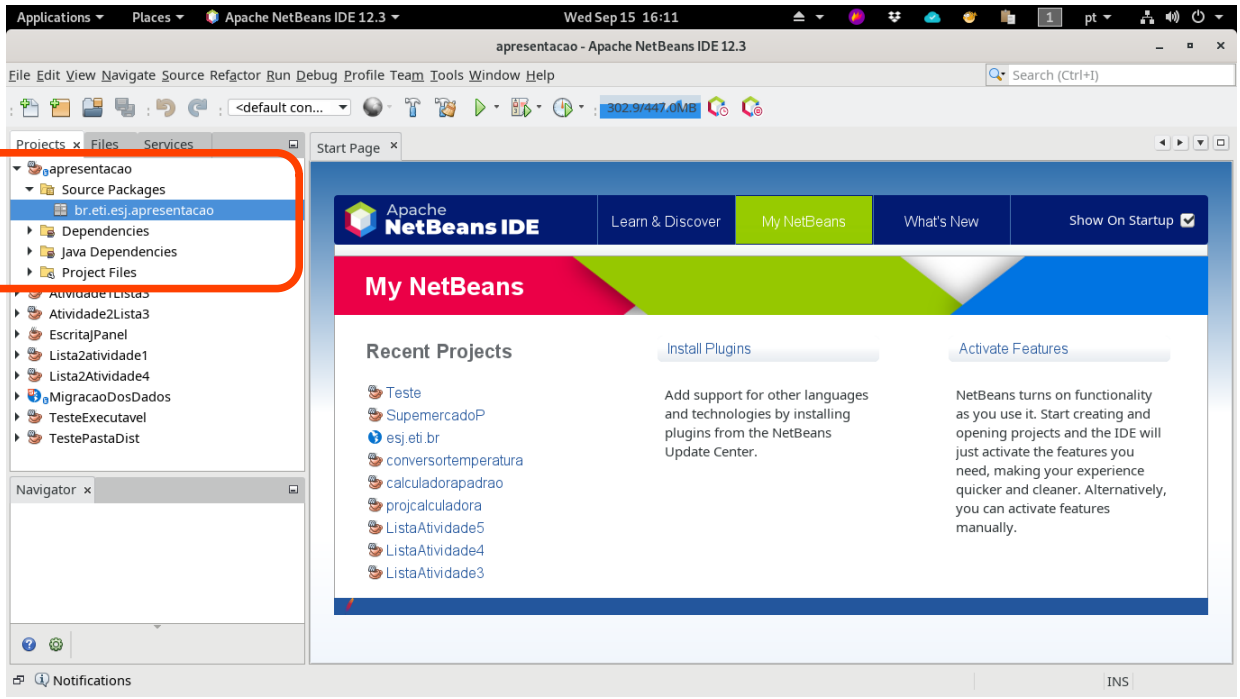

Figura 4: Projeto criado (circundado em vermelho).

6) você notará que a IDE, cria um projeto com o nome que você escolheu. Cria também um *package* com o nome que você escolheu para o projeto ou com o nome que você eventualmente tenha escrito no campo *Package* da Figura 3;
7) agora precisamos criar uma classe principal, que eu costumo dar o nome de Principal ou *Main*. Para tanto, clique com o botão direito do *mouse* no nome do *package* que foi criado (está marcado em azul, circundado por um retângulo vermelho, na Figura 4). No *menu pop-up* que aparecer, escolha a opção "*New* | *Java Class*…", então será mostrada a tela exposta na Figura

5. Escolha um nome pr'aquela que será a classe principal do seu *software*. Para este exemplo escolhemos o nome "Principal" e já o escrevemos no campo *Class Name* da tela na Figura 5.

8) ainda na tela apresentada na Figura 5 aperte o botão *Finish* para que seja criada a classe de nome "Principal". Observação: nomes de classes em Java, sempre começam com letra maiúscula (padrão *Camel Case*);

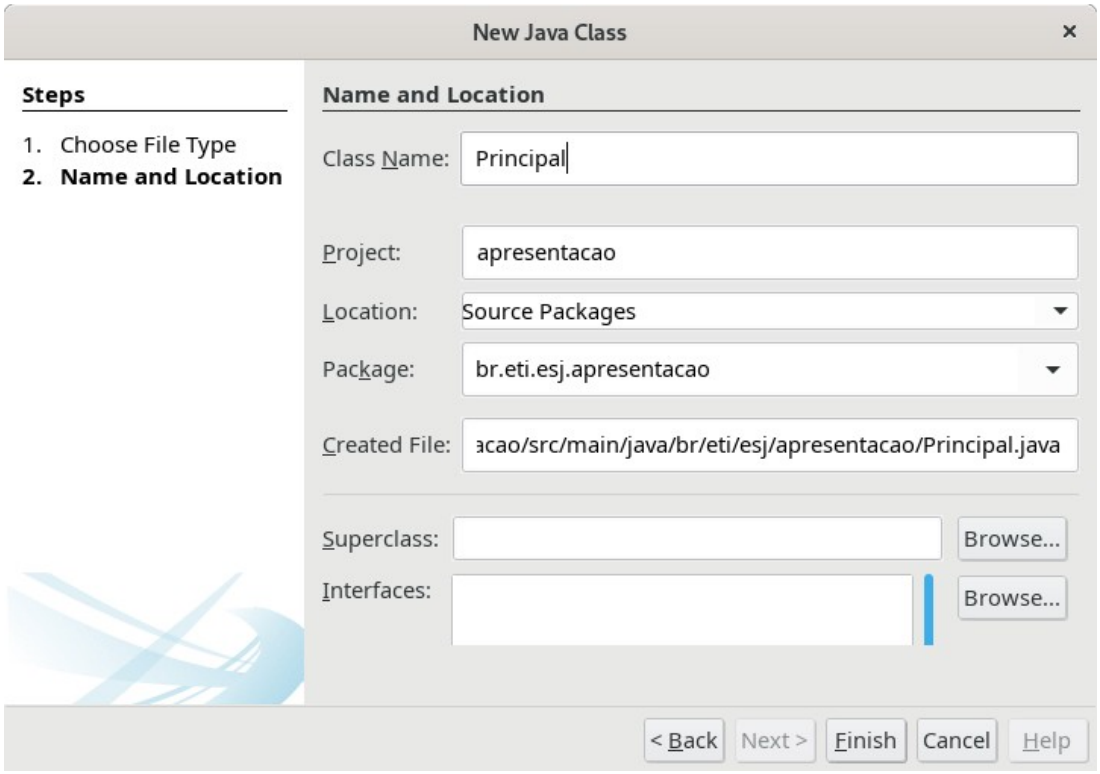

Figura 5: Criando a classe principal.

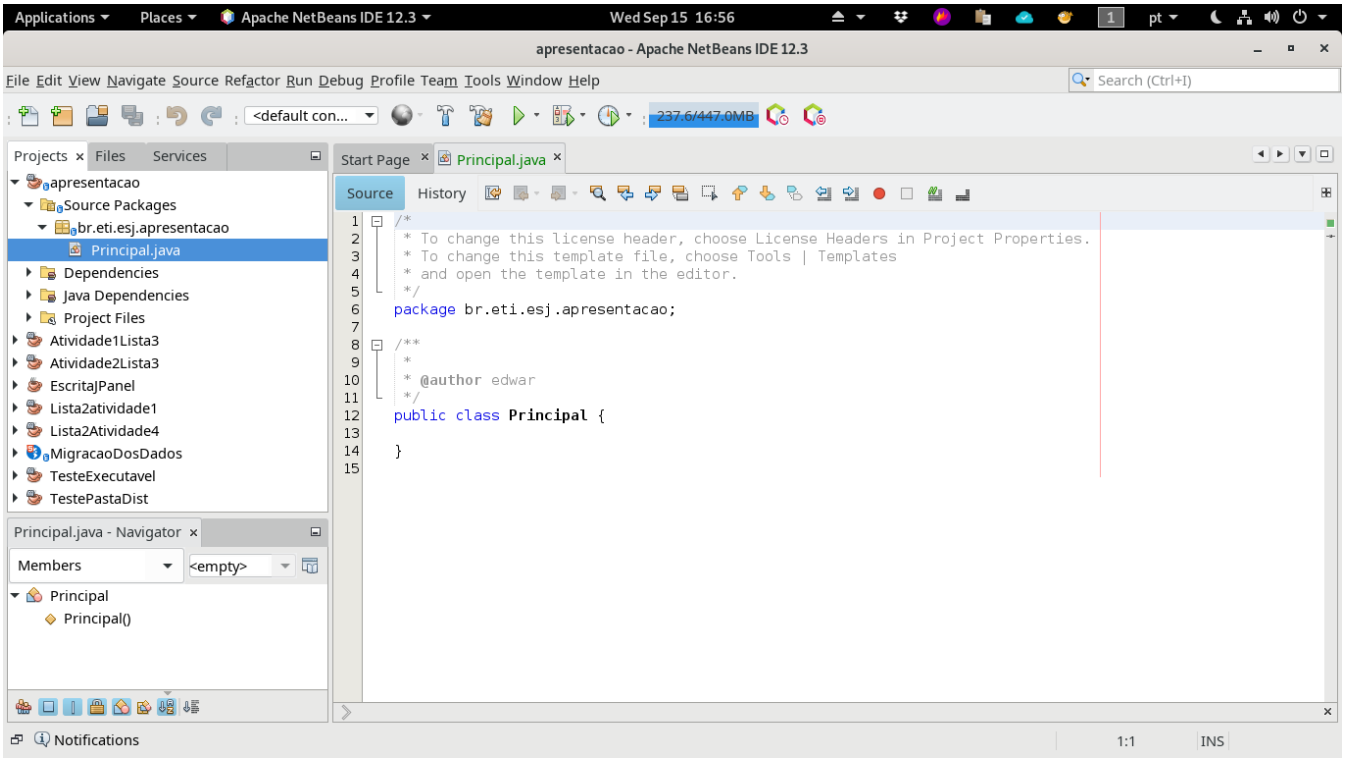

Figura 6: Classe Principal criada.

9) na Figura 6 pode-se ver a classe Principal criada pela IDE. Agora, para continuarmos nosso exemplo, vamos escrever o código-fonte a seguir na classe Principal:

```
public class Principal {
    public static void main(String[] args){
        int idadeAtual, idade2058, anoAtual, anoNascimento;
        Scanner input = new Scanner(System.in);

        System.out.println("Digite o ano atual: ");
        anoAtual = input.nextInt();
        System.out.println("Digite seu ano de nascimento: ");
        anoNascimento = input.nextInt();

        idadeAtual = anoAtual - anoNascimento;
        idade2058 = 2058 - anoNascimento;

        System.out.printf("\nSua idade atual é: %d anos.", idadeAtual);
        System.out.printf("\nSua idade em 2058 será: %d anos.", idade2058);
    }
}
```

10) após s escrita do código-fonte na classe Principal, você terá algo como mostrado na Figura 7. Observe que na linha 3, foi necessário importarmos o *package* `java.util.Scanner`
11) agora compile-o e execute-o. Para compilação e execução em sequência, basta pressionar a tecla F6 ou apertar o botão apontado pela seta vermelha na Figura 7;

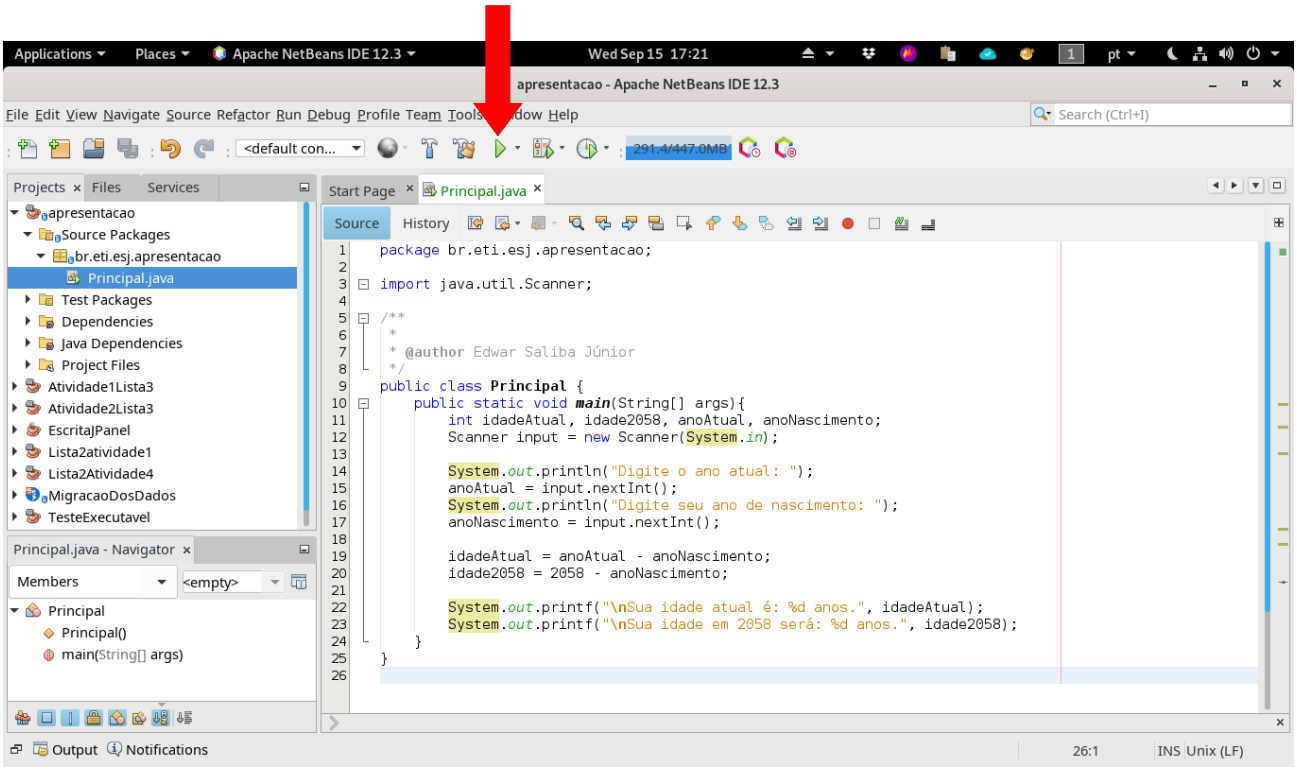

Figura 7: Código -fonte digitado.

Depurando Um Código-fonte:

12) agora vamos aprender a executar um programa passo a passo, ou seja, depurá-lo. Vamos colocar um *breakpoint* na linha 19, na linha do comando de atribuição da variável ”idadeAtual”. Para tanto, dê um clique com o ponteiro o *mouse* sobre o número da linha caso este esteja visível; caso não esteja visível, basta clicar próximo à borda da parte cinza onde são apresentados os números de linhas. Será colocado um quadradinho sobre o número da linha e também uma linha rosa sobre a linha inteira, algo como mostrado na Figura 8;

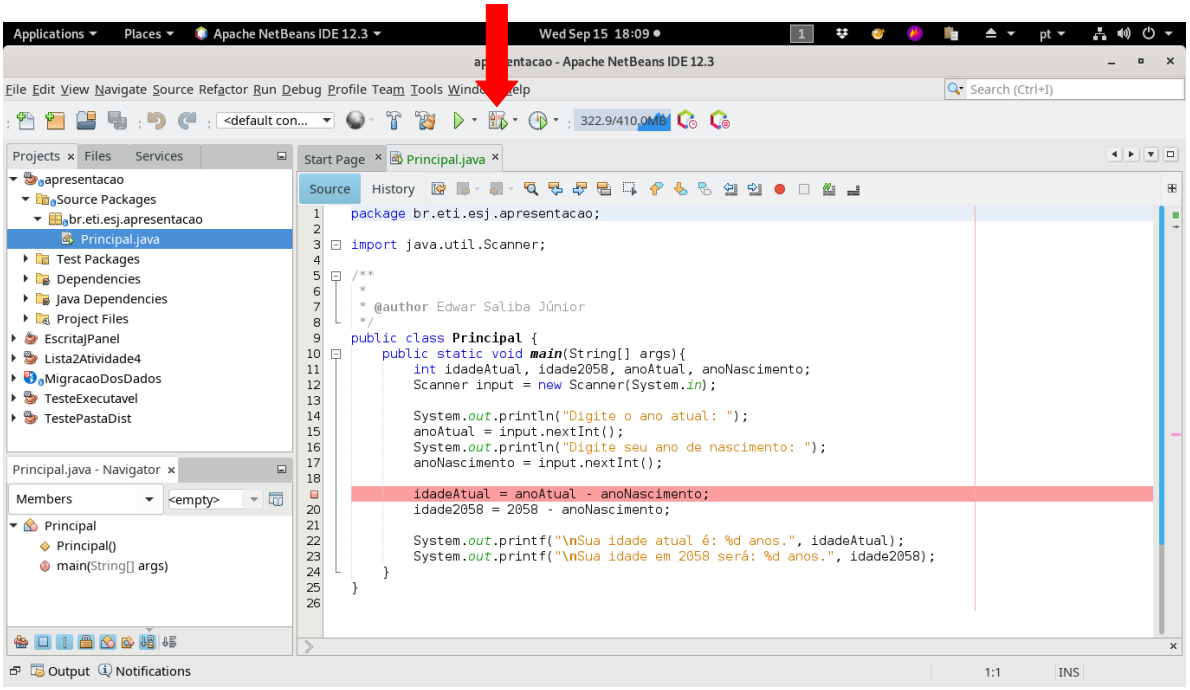

Figura 8: Breakpoint.

13) agora aperte as teclas “Ctrl + F5” (*Debug Main Project*) ou o botão de depuração (apontado pela seta vermelha na Figura 8). O *software* será executado, solicitando normalmente que você entre com o “Ano Atual” e o “Ano de Nascimento” (Figura 9);

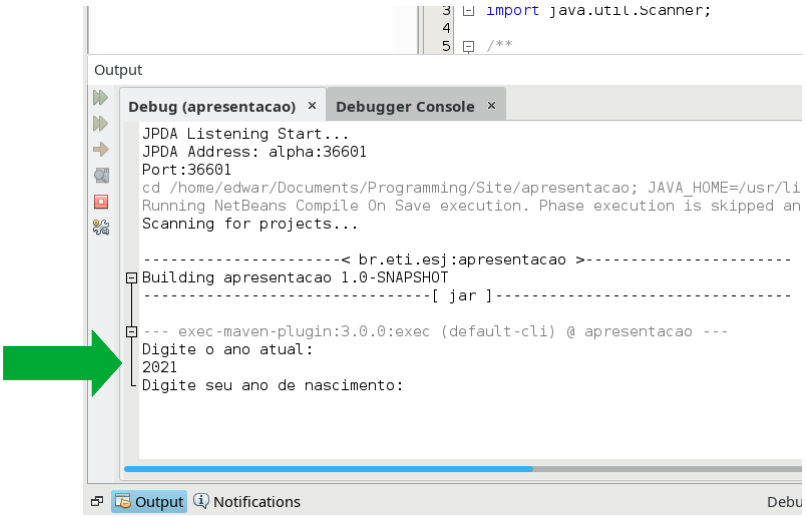

Figura 9: Depuração - entrada de dados.

14) após a segunda entrada de dados, o compilador encontrará o seu *breakpoint* e ficará parado sobre ele. Isto quer dizer que a linha onde está o *breakpoint* ainda não terá sido executada. Você poderá observar, que a linha do *breakpoint* ficará verde (Figura 10).

15) observe, na Figura 10, que abaixo do código-fonte apareceu uma janela chamada *Variables* e que esta janela mostra algumas variáveis do nosso programa e seus respectivos valores atuais (setas rosas na Figura 10). A medida que o programa vai sendo executado (linha a linha) as demais variáveis que estão adiante aparecerão na tela *Variables*. Caso uma variável esteja fora do escopo de execução, então ela não aparecerá na tela *Variables*, mas se ainda assim ela existir e você quiser observar o valor dela, basta entrar com os dados corretamente no campo `<Enter new watch>` (apontado pela seta azul na Figura 10).

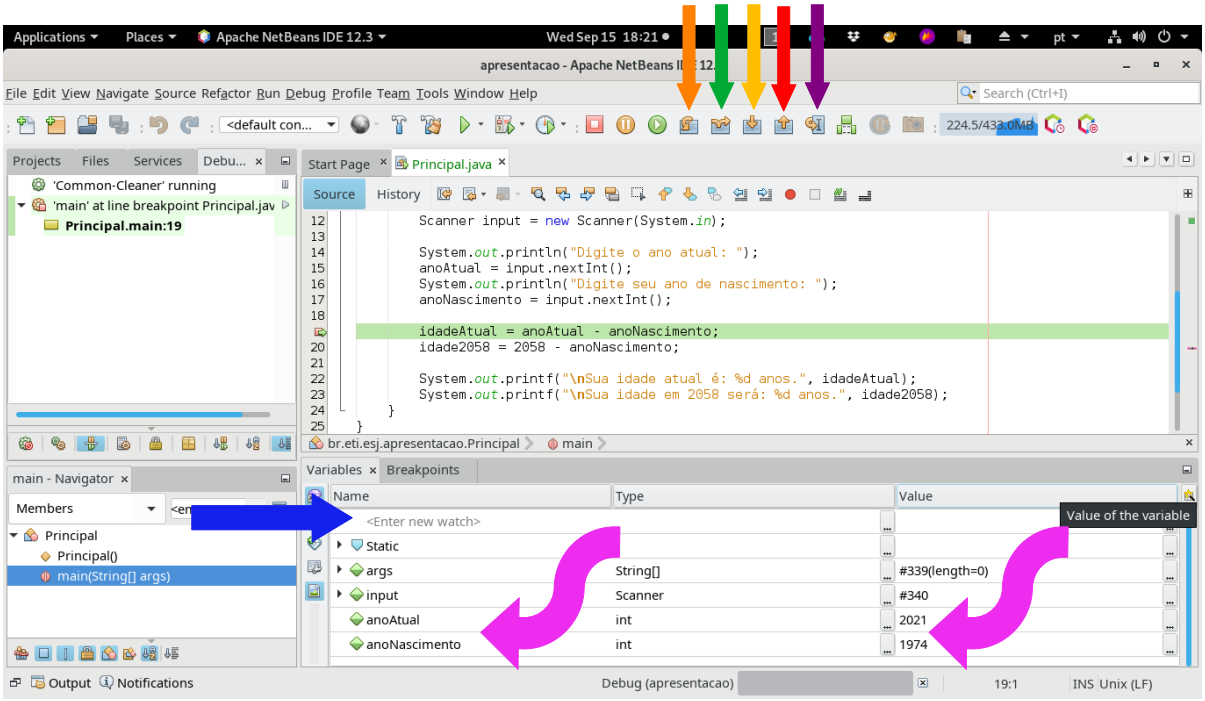

Figura 10: Depuração

16) para continuar a depuração do programa, basta usar um dos seguintes botões (que só aparecem se a IDE estiver em modo de depuração):

- botão ***Step Over*** (tecla de atalho `F8` seta laranja na Figura 10) executa uma linha de código-fonte sem entrar dentro das funções (métodos) que nesta linha existirem, se for o caso;
- botão ***Step Over Expression*** (tecla de atalho `Shift + F8` seta verde na Figura 10) semelhante ao *Step Over*, porém cada linha de código-fonte é executada em dois passos (dois apertos de botão). No primeiro passo é mostrado o valor de cada argumento que cada função (método) está recebendo e, no segundo passo, a linha inteira completa sua execução. Ao executar a linha o programador poderá visualizar ambos os passos na tela *Variables*. Se não houver funções (métodos) na linha de código-fonte executada, então o funcionamento deste botão fica idêntico ao do botão *Step Over*;
- botão ***Step Into*** (tecla de atalho `F7` seta amarela na Figura 10) executa uma linha de código-fonte, porém, se existirem funções (métodos) nesta linha, o depurador entrará em uma por uma e as executará linha a linha;
- botão ***Step Out*** (tecla de atalho `Ctrl + F7` seta vermelha na Figura 10) ao usar o botão *Step Into* para entrar numa função (método) e, por um motivo qualquer

você quer sair desta função (método) sem ter que executá-la linha a linha até o final, então basta apertar o botão *Step Out* que ele providenciará a saída da função (método) em que estiver e voltará exatamente para a linha onde ocorreu a chamada da função (método) a qual acabou de sair de dentro. Porém, a saída se dá sem que esta linha ainda tenha sido executada por completo. Ou seja, será necessário apertar o botão mais uma vez para que a execução da linha esteja completa e ocorra o salto para a próxima linha de código-fonte e

- botão ***Run to Cursor*** (tecla de atalho `F4` seta roxa na Figura 10) executa linhas do código-fonte e para na linha em que o cursor estiver posicionado. Isto, se houver código-fonte na linha;

17) pressionando a tecla `F8` ou o botão *Step Over* (seta laranja na Figura 10), pode-se ver a execução linha a linha do código-fonte em questão. E, automaticamente, as mudanças nos valores das variáveis, através da tela *Variables* (Figura 11 - seta vermelha).

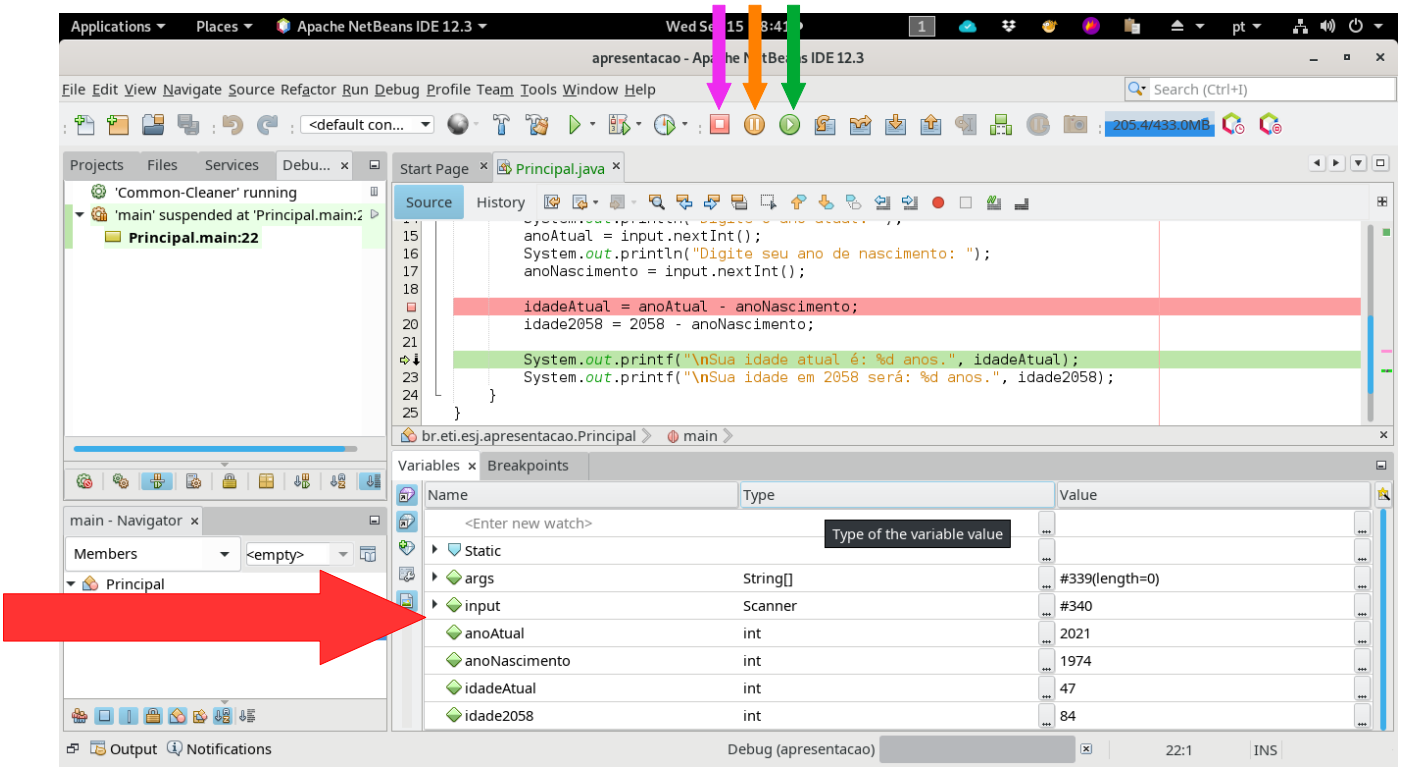

Figura 11: Variables

18) a medida que pressiona-se a tecla `F8` e vai-se navegando pelo código-fonte, você poderá observar o comportamento das variáveis na tela *Variables*. Desta forma, você poderá depurar o seu *software* e eventualmente consertar todos os erros de lógica que encontrar. Existem outros três botões que podem ser úteis neste trabalho, são eles:

- o botão ***Finish Debugger Session*** (tecla de atalho `Shift + F5` seta rosa na Figura 11), uma vez apertado este botão interrompe imediatamente a execução do código-fonte;
- o botão ***Pause*** (sem tecla de atalho, seta laranja na Figura 11) pausa o processo de execução. Muito útil quando se estiver depurando, por exemplo, uma situação específica que aconteça dentro de uma estrutura de repetição e
- o botão ***Continue*** (tecla de atalho `F5` seta verde na Figura 11) executa o código-fonte sem parar até encontrar um *breakpoint* ou o fim do programa;

19) **observação:** depura-se um código-fonte quando o programa possui um erro de lógica, como por exemplo, uma conta ou alguma operação que não está se comportando como deveria;

Entendendo o Código-fonte:

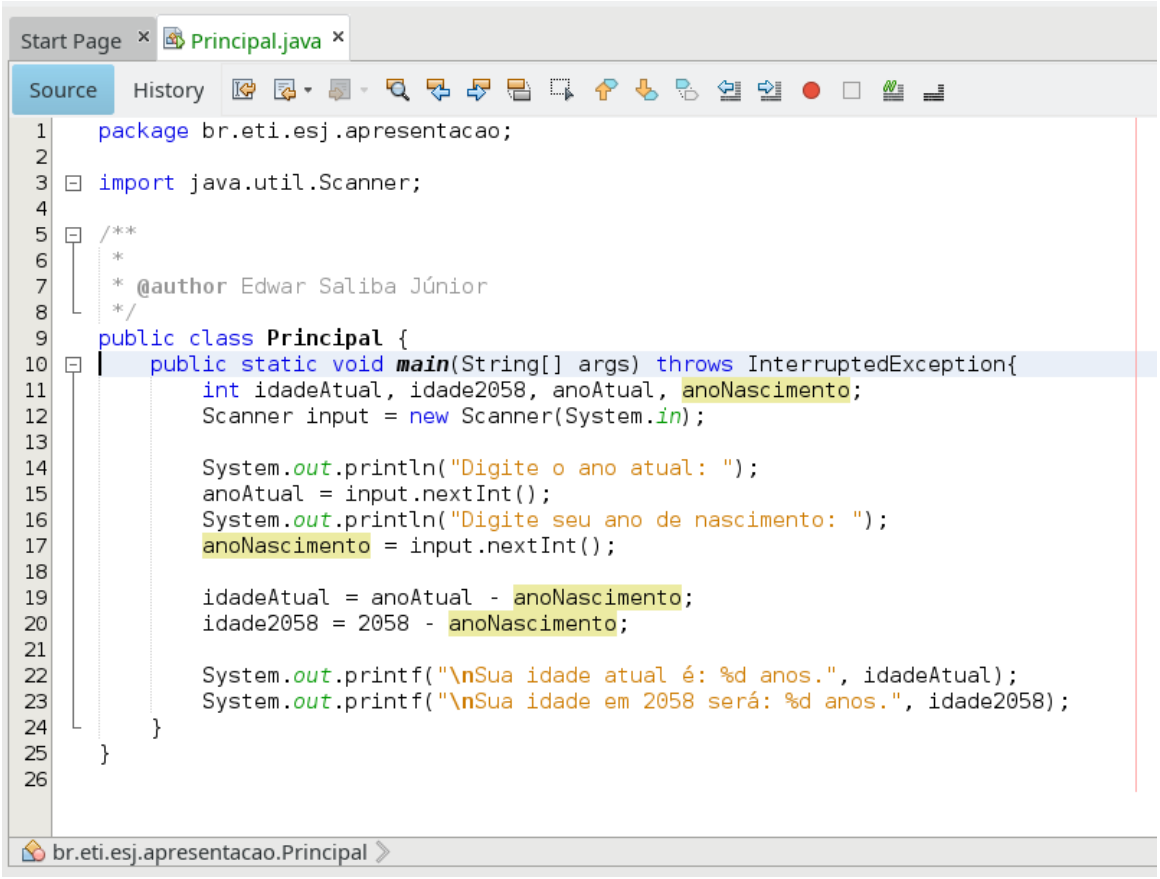

Figura 12: Código-fonte "apresentacao".

package – mecanismo para organização de classes que dizem respeito a um mesmo módulo ou assunto tratado pelo sistema. Um *package* ou pacote (em Português) nada mais é do que uma pasta (diretório) no sistema operacional;

import – semelhante ao comando *#include* da linguagem C. Através do comando *import*, pode-se utilizar outras classes (que estão em outros *packages*) ou bibliotecas predefinidas na linguagem, como está sendo feito no exemplo acima (Figura 12);

public – modificador de acesso, refere-se a acessibilidade que se pode ter em relação a um método, classe ou atributo;

Acessibilidade a atributos e métodos: depende dos modificadores de acesso que os prefixam em sua definição e também que prefixam as classes:

- ***public***: dá acessibilidade completa ao atributo, este pode ser acessado a partir da própria classe e também por outras classes, estando estas classes no mesmo *package* ou não. Use este modificador de acesso com extrema moderação! Para atributos;

- ***protected ou a falta do modificador de acesso***: dá acessibilidade total aos atributos e métodos para todas as classes que estão no mesmo *package*;
- ***private***: só se pode ser acessado por membros da própria classe;

static – o modificador *static* nos garante que somente haverá uma, e não mais que uma, referência para determinada variável ou método disponível em memória. Em outras palavras, declarando alguma coisa como *static*, todas as instâncias da classe compartilharão a mesma cópia da variável ou método. Declarar algo como *static* também permite você acessar métodos e atributos diretamente, ou seja, sem precisar criar uma instância da classe;

void – palavra-chave utilizada na declaração de funções que não precisam retornar algum valor;

int – palavra-chave utilizada na declaração/criação de variáveis do tipo inteiro, ou seja, criação de espaços na memória do computador, capazes de armazenar valores do tipo inteiro;

public static void main(String[] args) - assinatura do método principal;

{ - início do corpo do método (função);

} - fim do corpo do método;

"**int idadeAtual**" - criação de variável do tipo inteiro. Ou seja, criação de um espaço na memória do computador, onde pode-se guardar valores do tipo inteiro;

// Este é um comentário de uma só linha;

/* Este é um comentário de uma ou mais linhas. *****/***

/** Este é um comentário de documentação. *****/***

o que estiver como comentário não é executado pelo compilador. Utilizado para documentar o *software*;

System.out.println - comando para impressão de valores na tela do computador;

System.out.printf - comando para impressão de valores na tela do computador. Similar ao comando *printf* da Linguagem C;

new – comando utilizado para instanciação de objetos;

Scanner – classe utilizada para captação de dados via teclado;

Dica de Padronização em Java (padrão *Camel Case*):

Em Java, o:

- nome de toda classe começa com letra maiúscula. Exemplo:
  - `public class Carro {};`
- nome de todo atributo e método começa com letra minúscula. E quando este é formado por mais de uma palavra, então as demais palavras que compõem o nome, com exceção da primeira, iniciarão com letra maiúscula. Exemplo:
  - `int meuAtributoNovo;`

- `void meuMetodoNovo( ){};`

Como gerar um arquivo “executável” em Java:

Um arquivo executável em Java possui a extensão “`.jar`” e deve ser executado com a seguinte linha de comando:

```
java -jar nomeDoExecutavel.jar
```

- antes do código-fonte do NetBeans ser transferido para os cuidados da Apache, bastava apertar o botão “*Build*” da IDE que automaticamente era gerada uma pasta “`dist`”, dentro da pasta do projeto do *software*, com o arquivo “`.jar`” e suas respectivas bibliotecas. Após a transferência para a Apache e a incorporação obrigatória do Maven (biblioteca para atualização de bibliotecas de *softwares*) aos *softwares* produzidos, agora é necessário adicionar um *script*, dentro do arquivo `pom.xml`, para que o arquivo “`.jar`” seja devidamente gerado dentro da pasta “`target`”.

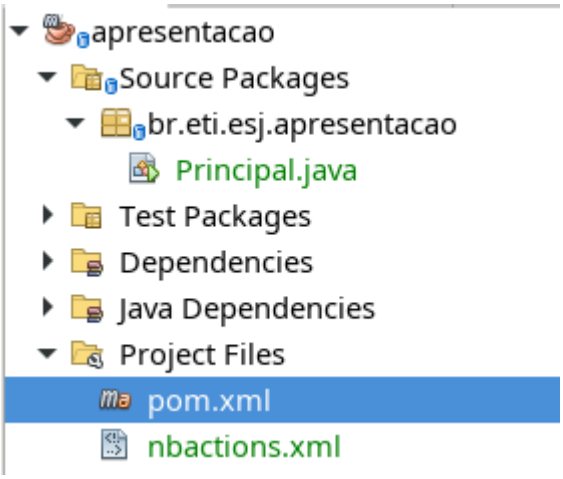

Figura 13: Arquivo "pom.xml".

A seguir o *script* de um arquivo `pom.xml` já com o *script* específico para a geração do arquivo executável; ou seja, tudo que estiver dentro da tag `<build>` (em destaque abaixo):

```
<?xml version="1.0" encoding="UTF-8"?>
<project                          xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.oorg/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>apresentacao</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <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>
    <build>
        <resources>
            <resource>
                <directory>src/main/java</directory>
                <includes>
                    <include>**/*.java</include>
                    <include>**/*.png</include>
                </includes>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <configuration>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <mainClass>br.eti.esj.apresentacao.Principal</
mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Bibliografia:

NETBEANSVIDEOS. **How to Use the NetBeans Debugger for Java** Disponível em: <https://www.youtube.com/watch?v=2Z9B8wYhKWw>. Acesso em: 15 Set. 2021.

STACKOVERFLOW. **What is the difference between "step over" and "step over expression" in debugging of NetBeans?**. Disponível em: <https://stackoverflow.com/questions/42103846/what-is-the-difference-between-step-over-and-step-over-expression-in-debuggi>. Acesso em 15 Set. 2021.