Prof. MSc. Marcos Alexandruk

E-mail: alexandruk@uni9.pro.br

https://github.com/alexandruk/poo

Resumão

aula 01 a 04

Java

Java é uma linguagem multiplataforma.

Os programas Java são executados a partir da JVM (Java Virtual Machine).

A JVM está disponível para os sistemas operacionais Windows, Linux, MacOS, etc.

Extensões dos arquivos Java

.java - Arquivos de código fonte Java.

.class - Arquivos compilados Java (bytecodes).

.jar - Pacote de arquivos .class. Possui informações de execução do projeto.

IDE - Integrated Development Environment

Os Ambientes de Desenvolvimento Integrados (IDE, em inglês) fornecem um conjunto de ferramentas de desenvolvimento que aumentam a produtividade.

Algumas IDEs para desenvolvimento em Java:

- Netbeans
- Eclipse
- BlueJ
- JCreator
- Visual Studio Code

Classes

Classes representam objetos do mundo real.

Para criar uma classe em Java utilizamos a palavra reservada **class** seguida pelo nome da classe. (Nomes de classes, em Java, começam com letras maiúsculas. Exemplos: Aluno, PessoaFisica, etc.)

Dentro das classes são declarados **atributos** e **métodos**.

atributos: são os elementos que definem a estrutura da classe e são também conhecidos como variáveis de classe. (Nomes de atributos começam com minúsculas)

métodos: executam tarefas quando chamados. São equivalentes a funções ou procedimentos em outras linguagens de programação. (Nomes de atributos começam com minúsculas)

Métodos, dependendo da aplicação, recebem ou não valores de entrada (parâmetros) e também podem ou não retornar valores para quem os chamou.

Métodos "tipados" retornam valores e métodos vazios (void) não retornam valores.

Classes (Exemplo)

public é um modificador de acesso que permite acesso externo à classe.
Os outros modificadores de acesso são: **protected** e **private**.
Modificadores de acesso serão considerados em detalhes mais adiante.

String é um tipo de dado que armazena um conjunto de caracteres.
int é um tipo de dado que armazena números inteiros.

return indica que o método retornará um dado ao ambiente que o chamou.
O método **getNome** retornará o **nome**.
O método **getIdade** retornará a **idade**.

this é uma referência ao objeto atual, ou seja, o objeto cujo método está sendo chamado.

```
public class Pessoa {
```

Classe Pessoa

```
  String nome;
  int idade;
```

Atributos

```
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public int getIdade() {
    return idade;
  }
  public void setIdade(int idade) {
    this.idade = idade;
  }
}
```

Métodos

Objetos

Um objeto ou instância é uma materialização da classe.

Para que os objetos ou instâncias possam ser manipulados, é necessária a criação de referências a estes objetos, que são basicamente variáveis do tipo da classe.

Para criar novos objetos utilizamos o operador **new**:

```
Pessoa pessoa1 = new Pessoa();
Pessoa pessoa2 = new Pessoa();
```

(atribuímos às variável pessoa1 e pessoa2 as referências ao objeto Pessoa)

Objetos (Exemplo)

```
public class Pessoa {
  String nome;
  int idade;
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public int getIdade() {
    return idade;
  }
  public void setIdade(int idade) {
    this.idade = idade;
  }
  public static void main(String[] args) {
    Pessoa pessoa1 = new Pessoa();
    pessoa1.setNome("Fulano");
    pessoa1.setIdade(25);
    System.out.println(pessoa1.getNome());
    System.out.println(pessoa1.getIdade());
  }
}
```

Neste exemplo estamos criando um objeto ou instância da Classe **Pessoa**.
O objeto é referenciado como **pessoa1**.
pessoa1 tem os atributos da Classe **Pessoa**.
pessoa1 usa os métodos da Classe **Pessoa**.

Objeto (instância da classe Pessoa)

Tipos primitivos

	Tipo	Descrição	Bytes	Exemplo
	`char`	Armazena um único caractere. A atribuição de valor deve ser feita sempre com aspas simples;	2	`char sexo = 'M';`
inteiros	`byte`	Armazena valores inteiros de -128 a 127	1	`byte idade = 55;`
	`short`	Armazena valores inteiros de -32,768 a 32,767	2	`short x = 3456;`
	`int`	Armazena valores inteiros de -2.147.483.648 a 2.147.483.647	4	`int y = 678934;`
	`long`	Armazena valores inteiros de 9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	8	`long cod = 1756453;`
reais	`float`	Armazena valores de ponto flutuante, ou seja, valores reais (que necessitam de casas decimais) com precisão de 6 ou 7 dígitos decimais. Utiliza-se o ponto para separação do decimal.	4	`float pi = 3.1415f;`
	`double`	Armazena valores de ponto flutuante, ou seja, valores reais precisão de 15 dígitos decimais. Utiliza-se o ponto para separação do decimal.	8	`double valor = 34.56d;`
	`boolean`	Armazena informações que podem ser apenas de dois tipos: verdadeiro (true) ou falso (false).	1	`boolean casado = true;`

Tipos de referência

String: Este tipo de dados é usado para se armazenar texto (conjunto ou cadeia de caracteres). Esse tipo é implementado pelo Java através da classe **String.java**.

Deve-se utilizar aspas duplas para atribuição de valores a este tipo. Exemplo:

```
String nome = new String("Fulado de Tal");
```

Date: É utilizado para armazenamento de datas. É implementado pelo Java através da classe **Date.java**. Exemplo:

```
Date dtNascimento = new Date("10/10/2020").
```

IMPORTANTE:

Note que todos os tipos primitivos são declarados com todas as letras minúsculas e os tipos de referência começam com uma letra maiúscula. Porque? Porque String e Date são classes Java.

Converter tipos de dados (casting)

É muito comum a necessidade de converter um tipo em outro, isso, em programação, chama-se "casting".

Exemplo: Converter 20.5 (float) para 20 (int):

```
float x = 20.5f; //f indica que é um float (número real)

int y = (int)x; //neste caso o Y valerá 20
```

Expressões aritméticas

Utilizadas para cálculos matemáticos convencionais.

+ (soma)

- (subtração)

* (multiplicação)

/ (divisão)

```
double x = 30;
double y 10;
double a = x + y; // a = 40
double b = x - y; // a = 20
double c = x * y; // a = 300
double d = x / y; // a = 3
```

Incremento e decremento:

Incremento: Aumenta o valor da variável

```
a = a + 1;
```

Forma reduzida dessa mesma expressão:

```
a++
```

Incrementar de dois em dois:

```
a = a + 2;
```

Forma reduzida dessa mesma expressão:

```
a += 2;
```

Decremento: Diminui o valor da variável

```
a = a - 1;
```

Forma reduzida dessa mesma expressão:

```
a--;
```

Incrementar de dois em dois:

```
a = a – 2;
```

Forma reduzida dessa mesma expressão:

```
a -= 2;
```

Expressões relacionais

As expressões relacionas comparam dois valores.

Resultados possíveis: verdadeiro (true) ou falso (false).

== (igual)

!= (diferente)

< (menor)

<= (menor ou igual)

> (menor)

>= (menor ou igual)

```
int a, b, c;
a = 2;
b = 5;
c = 5;
//...
a == b  // false
b == c  // true
b < a   // false
b <= c  // true
a > b   // false
b >= c  // true
```

Expressões lógicas (booleanas)

Comparam de dois valores boolean e resultam em um terceiro valor boolean.

&& AND (e): Retorna true apenas se as duas expressões retornarem true

|| OR (OU): Retorna true se uma das duas expressões retornar true

! NOT (Negação): true vira false e false vira true

```
int a, b, c;
a = 2;
b = 5;
c = 5;
((a < b) && (b < c))     // false
((a < b) || (b < c))     // true
((a < b) && (b <= c))    // true
(!((a < b) || (b < c))) // false
```

Controle de fluxo (Estrutura de decisão): if...else

Um determinado trecho de código é executado caso o teste lógico seja verdadeiro (true) e outro caso o resultado seja falso.

```
//...
int a = 10;
int b = 10;
if (a == b) {
  System.out.println("a é igual a b"); // imprimirá isso, pois "a" é igual a "b"
}
else {
 System.out.println("a é diferente de b");
}
//...
```

Controle de fluxo (Estrutura de decisão): if...else if...else

Permite avaliar mais expressões (booleanas) na mesma estrutura de decisão.

```
//...
int a = 10;
int b = 10;
int c = 20;
if ((a < b) || (b == c)) {
  System.out.println("a é menor que b ou b é igual a c");
}
else if ((a <= b) && (b == c)) {
 System.out.println("a é menor ou igual a b e b é igual a c");
}
else if ((a <= b) || (b == c)) {
 System.out.println("a é menor ou igual a b ou b é igual a c"); // imprimirá isso
}
else {
 System.out.println("nenhuma expressão avaliada retornou true");
}
//...
```

Controle de fluxo (Estrutura de decisão): switch

Outra forma para fazer controle de fluxo.

Usamos uma única variável para realizar comparações encadeadas.

```
//...
String uf = "SC";
switch(uf) {
  case "RS":
  System.out.println("Rio Grande do Sul");
  break;
  case "SC":
  System.out.println("Santa Catarina");
  break;
  case "PA":
  System.out.println("Paraná");
  break;
  default:
  System.out.println("Não é um Estado da Região Sul");
}
//...
```

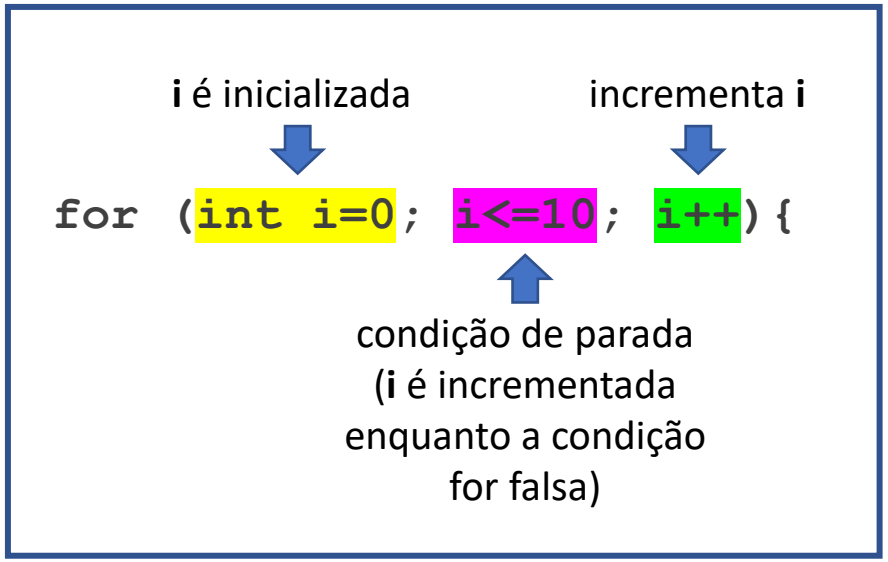
Laço (Estrutura de repetição): for

Utilizado quando queremos executar um conjunto de instruções um determinado número de vezes.

```
// imprime números de 1 a 10
for (int i=0; i<=10; i++){
System.out.println(i);
}

// imprime números de 1.1 a 10.10
for (int i=0; i<=10; i++){
  for (int j=0; j<=10; j++){
    System.out.println(i + "." + j);
  }
}

// imprime números ímpares de 1 a 10
for (int i=0; i<=10; i++){
  if (i % 2 == 0) // Verifica se o resto da divisão por 2 é 0
System.out.println(i);
}
```



Laço (Estrutura de repetição): while

Repete um conjunto de instruções enquanto o resultado de uma expressão lógica for verdadeiro.

```
// imprime números ímpares de 1 a 10
int i = 0;
while(i <= 10){
  if (i % 2 != 0) // Verifica se o resto da divisão por 2 é diferente de zero
  System.out.println(i); // Imprime a variável de controle
  i++; // Incrementa a variável de controle
}
//...
```

Laço (Estrutura de repetição): do...while

Avalia a expressão lógica no final do laço.

Utilizando **do...while** o laço será executado pelo menos uma vez, mesmo que a condição não seja verdadeira.

```
// imprime 10, pois a codição (i < 10) é verificada no final do laço
int i = 10;
do {
  System.out.println(i); // imprime a variável de controle
  i++; // incrementa a variável de controle
}
while (i < 10); // condição de parada
//...
```

Classe Scanner

A classe Scanner, pois tem a finalidade de facilitar a entrada de dados no modo Console.

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    String texto;
    Scanner leitor = new Scanner(System.in); // System.in lê o que se escreve no teclado
    System.out.print("Digite o texto: ");
    texto = leitor.nextLine();
    System.out.println("Você digitou: " + texto);
    leitor.close();
  }
}
```

Prof. MSc. Marcos Alexandruk

E-mail: alexandruk@uni9.pro.br

https://github.com/alexandruk/poo

Arrays e Coleções de Dados

aula 05

Objetivo

Implementar as principais formas estruturas de dados em Java, com aplicação das boas práticas de programação.

Arrays

Um array (ou vetor) é uma estrutura de dados que armazena uma série de objetos em sequência, todos do mesmo tipo.

Em Java, para declarar um array utilizamos um par de colchetes: **[ ]**.

Exemplo: Array que armazena até 10 valores inteiros:

```
//...
int x[] = new int[10];
//...
```

Cada posição do array armazena um número inteiro e **a numeração das posições inicia-se em 0 (zero) e termina em nove.**

Arrays

Exemplo: Array de três posições do tipo String:

```
//...
//declaração do array de strings de tamanho 3
String nomes[] = new String[3];
//atribui valor a cada posição do array, de 0 a 2
nomes[0] = "Fulano";
nomes[1] = "Beltrano";
nomes[2] = "Sicrano";
//para acessar cada posição, usamos um laço, de 0 a 2
for (int i = 0; i <= 2; i++) {
System.out.println("Na posição " + i + " do array, temos: " + nomes[i]);
}
//...
```

[aula05_01.txt]

Arrays

Graficamente, o exemplo apresentado cria a seguinte estrutura de dados:

Posição#	0	1	2
Valor	Fulano	Beltrano	Sicrano

É muito importante tomar muito cuidado na hora de percorrer ou acessar os valores de um array, para não acessar uma posição não existente.

No array acima, que apresenta posições de 0 a 2, não podemos acessar a posição 3, pois ela não existe. Isso daria um erro no programa que, se não tratado, finalizará inesperadamente o programa.

Arrays

Para praticar, vamos elaborar um programa que lê um array de 10 inteiros e imprime o maior valor inserido no array.

Dica: Usaremos uma variável auxiliar para armazenar o primeiro valor e, caso seja encontrado algum valor maior enquanto percorre o array, o valor da variável auxiliar é substituído pelo valor encontrado. A cada laço será preciso comparar o valor da posição com o valor da variável auxiliar.

[aula05_02.txt]

Vetores multidimensionais

Podemos criar arrays com duas ou mais dimensões.

Quando criamos arrays com duas dimensões (bidimensionais), o resultado é algo parecido com uma tabela de duas colunas.

Para praticar, criaremos um array bidimensional que armazena na primeira posição um nome e na segunda posição o sobrenome.

Nosso array terá o mesmo número de linhas e colunas. No entanto, em Java, podemos, criar um array com quantas linhas e colunas forem necessárias.

[aula05_03.txt]

Array tipado com uma classe criada por você

Além de criar arrays de tipos primitivos (int, float, etc.), podemos criar um array cujo tipo é determinado por nós.

Vamos criar, como exemplo, uma lista de pessoas em uma só estrutura de dados, ou seja, uma lista do tipo "Pessoa" na qual cada pessoa da lista tem suas próprias características.

Para isso, o primeiro passo será criar a classe que usaremos para "tipar" nosso array, neste caso, a classe "Pessoa".

Nossa classe **Pessoa** terá os seguintes atributos:

```
Nome (String)
Idade (int)
E-mail (String)
```

Array tipado com uma classe criada por você

A codificação dessa classe, poderá ser assim:

```
public class Pessoa {
   String nome;
   int idade;
   String email;
}
```

A seguir, vamos criar uma classe que poderá conter até 3 pessoas com as características mostradas acima e imprimir na console as informações das pessoas:

[aula05_04.txt]

Como descobrir o tamanho de um array?

Há uma função que retorna o tamanho total de um array:

```
nome_do_array.lenght
```

A função **length** retorna o tamanho total de posições endereçáveis. Portanto, se estiver iterando este array, não esqueça de ir de 0 até nome_do_array.lenght - 1.

```
public class Main {
  public static void main(String[] args) {
    String nomes[] = new String[3];
    System.out.println("Tamanho do vetor nomes: " + nomes.length);
  }
}
```

Implementações prontas no Java

Um **array** é uma **implementação manual** da estrutura de vetores.

Listas, por outro lado, são **implementações prontas** de estruturas de dados.

Embora gastem mais memória, obtemos algumas vantagens quando trabalhamos com estruturas de dados prontas em Java:

- Capacidade de alocação dinâmica de dados (a lista pode ter o tamanho que você quiser e este tamanho pode mudar em tempo de execução);
- Métodos prontos de ordenação, retirada e acréscimo de valores em posições da lista;
- Métodos prontos de consulta às listas.

ArrayList

ArrayList é a principal implementação de coleção em Java.

ArrayList é implementada pela classe **java.util.ArrayList.** Portanto, sempre que for utiliza-la, é preciso adicionar no início de seu código as instrução de importação dessa classe:

```
import java.util.ArrayList;
```

Para usar um ArrayList é preciso indicar o tipo de objeto que está sendo armazenado na estrutura de dados. A sintaxe de sua declaração é assim:

```
ArrayList<ObjetoUsado> nomeDaColecao = new ArrayList<ObjetoUsado>();
```

Para criar, por exemplo, uma coleção nomes de clientes, ou seja, de valores do tipo String, a ArrayList ficará assim:

```
ArrayList<String> clientes = new ArrayList<String>();
```

ArrayList

Um objeto do tipo ArrayList, possui à disposição os métodos que a classe implementa.

Os principais **métodos** são:

add(VALOR): Acrescenta um item na estrutura. O valor a ser inserido, deve ser do mesmo tipo da lista.
Exemplo: `clientes.add(1,"Fulano");`

add (ÍNDICE, VALOR): Acrescenta um valor na posição indicada no parâmetro ÍNDICE. A contagem nos ArrayLists também começa no índice 0.
Exemplo: `clientes.add(1,"Fulano");`

size(): Retorna o tamanho da lista.
Exemplo: `int x = clientes.size();`

get(POSIÇÃO): Retorna a objeto que está na POSIÇÃO (índice) informada.
Exemplo: `String cliente1 = clientes.get(1);`

remove(VALOR): Remove o valor informado. O objeto passado deve ser do mesmo tipo da ArrayList.
Exemplo: `clientes.remove("Beltrano");`

clear(): Limpa, completamente, a lista.
Exemplo: `clientes.clear();`

ArrayList

Para praticar vamos elaborar uma ArrayList tipada com a classe "Pessoa" (já criada).

O exemplo apresentado faz o seguinte:

- Cria uma lista contendo 3 pessoas;
- Imprime os dados de todas elas (uma a uma) usando um laço;
- Procura uma pessoa chamada "Beltrano" e remove a da lista;
- Imprime a lista de nomes que permaneceram na estrutura.

[aula05_05.txt]

ArrayList

Note que, neste exemplo, após a remoção do item, o índice que era ocupado pelo objeto que foi removido não existe mais, ou seja, a própria lista cuidou da remoção e realocação dos objetos que permaneceram na lista, automaticamente.

Se estivéssemos trabalhando com vetores essa remoção seria bem mais complicada, pois seria necessário mover todos os demais itens manualmente, após a remoção.


CERTIFIED
PROFESSIONAL

Prof. MSc. Marcos Alexandruk

E-mail: alexandruk@uni9.pro.br

https://github.com/alexandruk/poo

Gerando e Interceptando Exceções

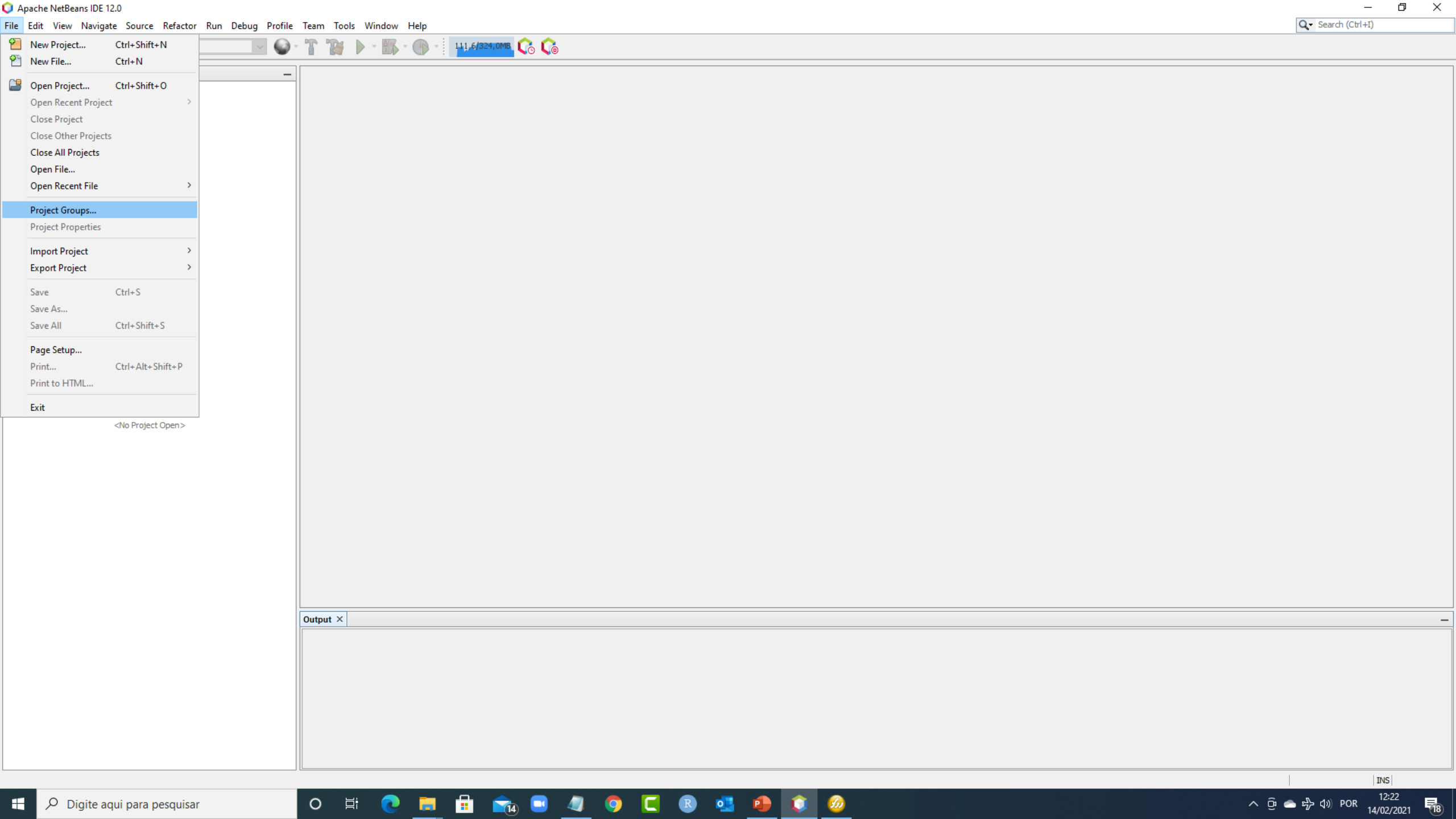
aula 06

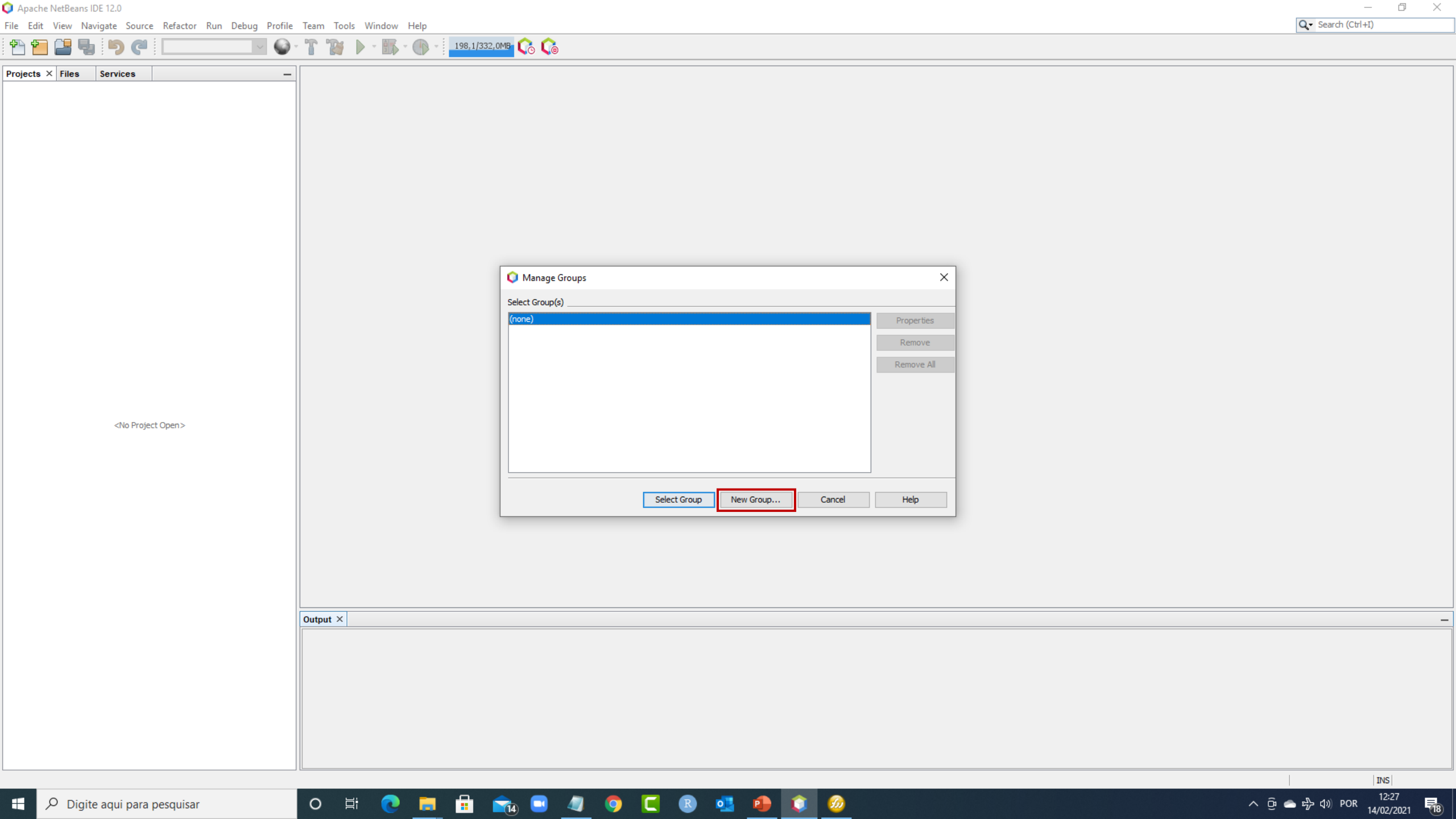
Objetivo

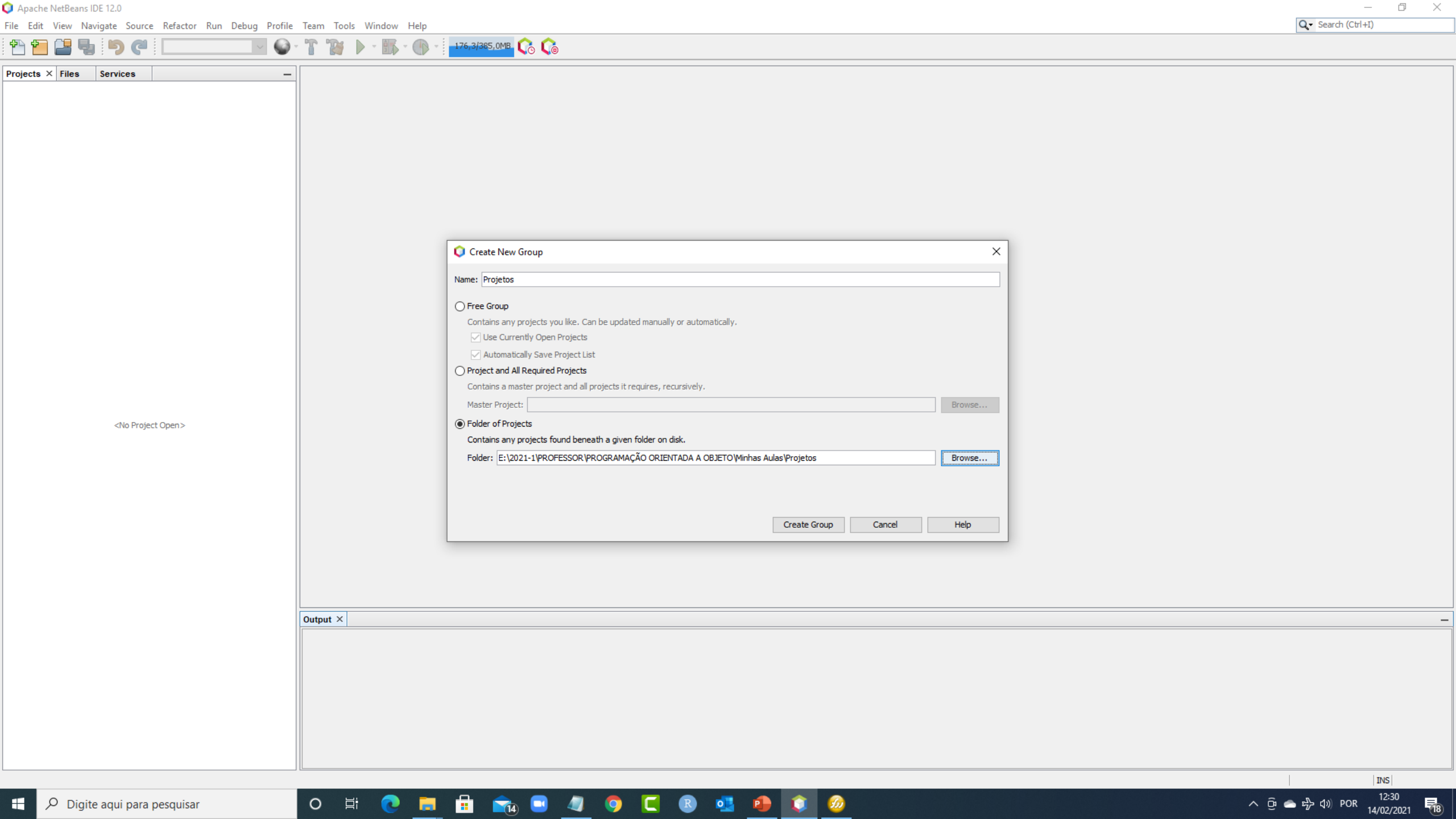
Tratamento de erros ou exceções durante a execução do programa, evitando que o programa encerre inesperadamente.

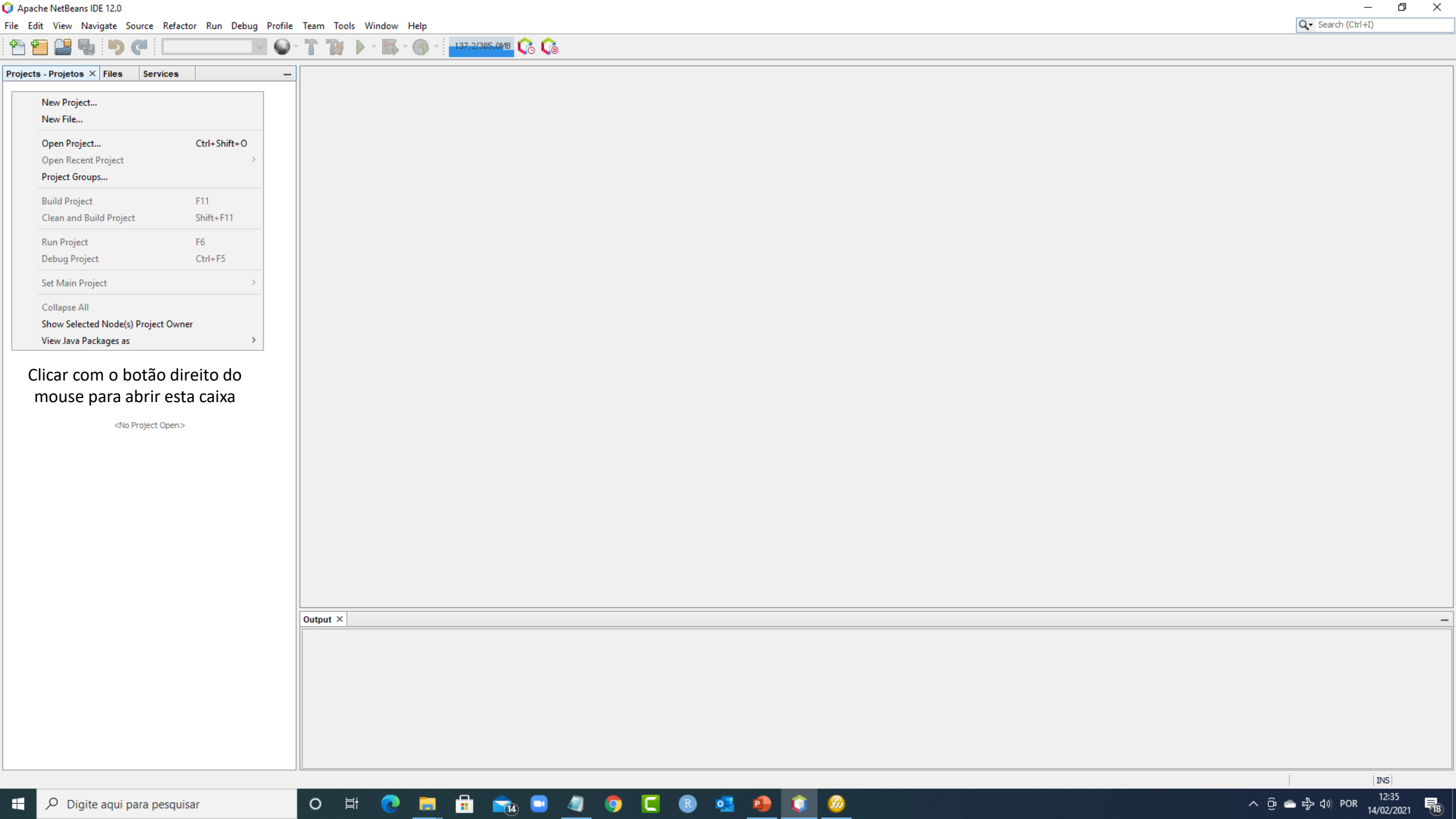
NetBeans

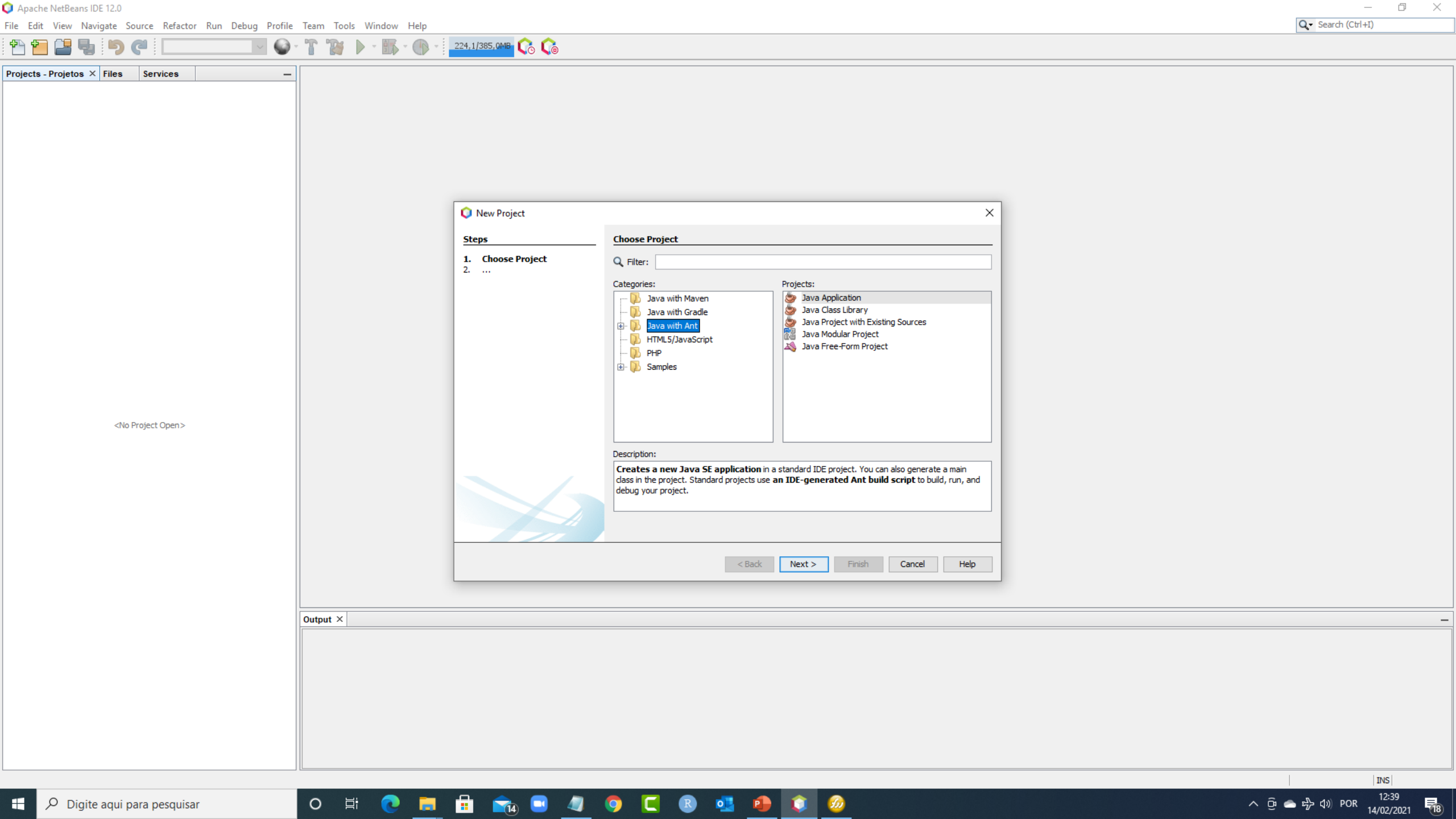
Breve Tutorial

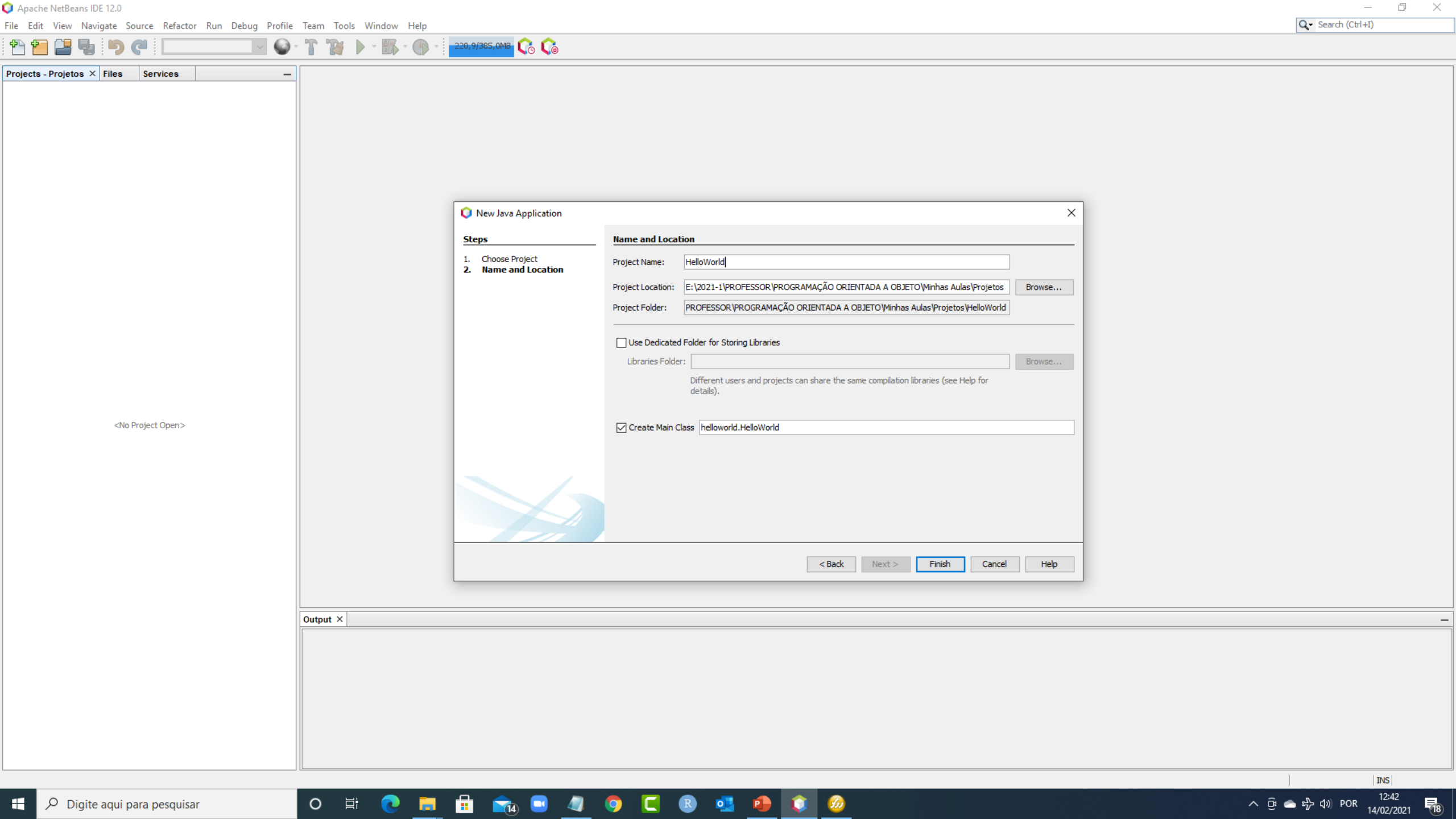

Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Close Project
Close Other Projects
Close All Projects
Open File...
Open Recent File
Project Groups...
Project Properties
Import Project
Export Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+Alt+Shift+P
Print to HTML...
Exit
111,6/324,0MB
<No Project Open>
Output
INS
Digite aqui para pesquisar
POR
12:22
14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
198,1/332,0MB
Projects
Files
Services
<No Project Open>
Manage Groups
Select Group(s)
(none)
Properties
Remove
Remove All
Select Group
New Group...
Cancel
Help
Output
INS
Digite aqui para pesquisar
POR
12:27
14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
176,3/385,0MB
Projects
Files
Services
<No Project Open>
Create New Group
Name: Projetos
Free Group
Contains any projects you like. Can be updated manually or automatically.
Use Currently Open Projects
Automatically Save Project List
Project and All Required Projects
Contains a master project and all projects it requires, recursively.
Master Project:
Browse...
Folder of Projects
Contains any projects found beneath a given folder on disk.
Folder: E:\2021-1\PROFESSOR\PROGRAMAÇÃO ORIENTADA A OBJETO\Minhas Aulas\Projetos
Browse...
Create Group
Cancel
Help
Output
INS
Digite aqui para pesquisar
POR
12:30
14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
137,2/385,0MB
Projects - Projetos
Files
Services
New Project...
New File...
Open Project... Ctrl+Shift+O
Open Recent Project
Project Groups...
Build Project F11
Clean and Build Project Shift+F11
Run Project F6
Debug Project Ctrl+F5
Set Main Project
Collapse All
Show Selected Node(s) Project Owner
View Java Packages as
Clicar com o botão direito do mouse para abrir esta caixa
<No Project Open>
Output
INS
Digite aqui para pesquisar
POR
12:35
14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
224,1/385,0MB
Projects - Projetos
Files
Services
<No Project Open>
New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
Java with Maven
Java with Gradle
Java with Ant
HTML5/JavaScript
PHP
Samples
Projects:
Java Application
Java Class Library
Java Project with Existing Sources
Java Modular Project
Java Free-Form Project
Description:
Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use an IDE-generated Ant build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help
Output
INS
Digite aqui para pesquisar
POR
12:39
14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
220,9/385,0MB
Projects - Projetos
Files
Services
<No Project Open>
New Java Application
Steps
1. Choose Project
2. Name and Location
Name and Location
Project Name: HelloWorld
Project Location: E:\2021-1\PROFESSOR\PROGRAMAÇÃO ORIENTADA A OBJETO\Minhas Aulas\Projetos
Browse...
Project Folder: PROFESSOR\PROGRAMAÇÃO ORIENTADA A OBJETO\Minhas Aulas\Projetos\HelloWorld
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
Create Main Class helloworld.HelloWorld
< Back
Next >
Finish
Cancel
Help
Output
INS
Digite aqui para pesquisar
POR
12:42
14/02/2021

HelloWorld - Apache NetBeans IDE 12.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config> 352,7/468,0MB

Projects - Projetos | Files | Services

- HelloWorld
  - Source Packages
    - helloworld
      - HelloWorld.java
  - Libraries

HelloWorld.java - Navigator

Members <empty>

- HelloWorld
  - HelloWorld()
  - main(String[] args)

HelloWorld.java

Source History

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package helloworld;
7
8  /**
9   *
10  * @author malex
11  */
12 public class HelloWorld {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         // TODO code application logic here
19     }
20
21 }
22
```

Output

1:1 INS Windows (CRLF)

Digite aqui para pesquisar

POR 12:43 14/02/2021

HelloWorld - Apache NetBeans IDE 12.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

348,4/485,0MB

Projects - Projetos | Files | Services

- HelloWorld
  - Source Packages
    - helloworld
      - HelloWorld.java
  - Test Packages
  - Libraries
  - Test Libraries

HelloWorld.java - Navigator

Members <empty>

- HelloWorld
  - HelloWorld()
  - main(String[] args)

HelloWorld.java

Source History

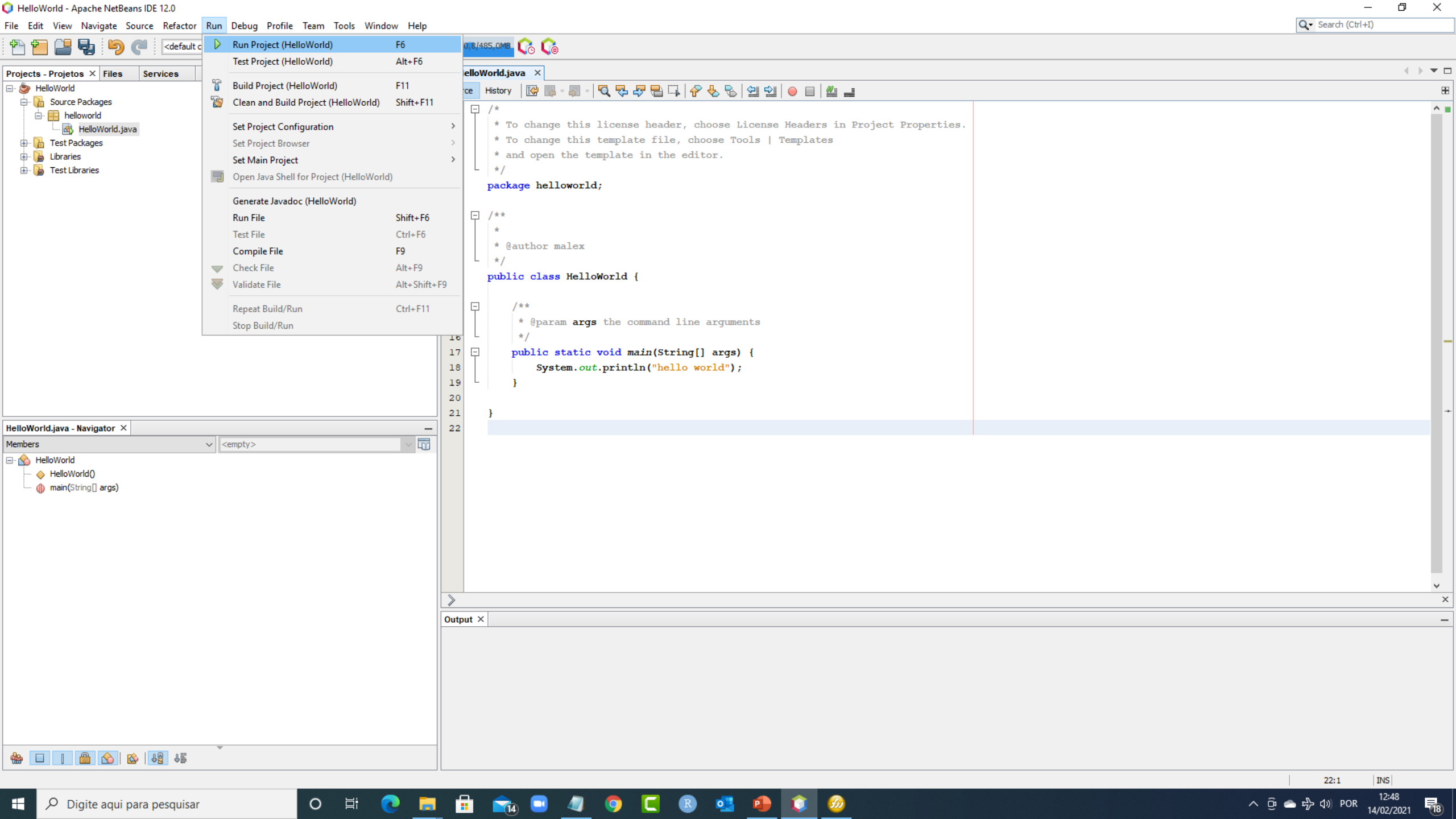
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package helloworld;
7
8  /**
9   *
10  * @author malex
11  */
12 public class HelloWorld {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         System.out.println("hello world");
19     }
20
21 }
22
```

Output

22:1 INS Windows (CRLF)

Digite aqui para pesquisar

POR 12:46 14/02/2021


HelloWorld - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Run Project (HelloWorld) F6
Test Project (HelloWorld) Alt+F6
Build Project (HelloWorld) F11
Clean and Build Project (HelloWorld) Shift+F11
Set Project Configuration
Set Project Browser
Set Main Project
Open Java Shell for Project (HelloWorld)
Generate Javadoc (HelloWorld)
Run File Shift+F6
Test File Ctrl+F6
Compile File F9
Check File Alt+F9
Validate File Alt+Shift+F9
Repeat Build/Run Ctrl+F11
Stop Build/Run
Projects - Projetos
Files
Services
HelloWorld
Source Packages
helloworld
HelloWorld.java
Test Packages
Libraries
Test Libraries
HelloWorld.java
History
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package helloworld;
/**
*
* @author malex
*/
public class HelloWorld {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
System.out.println("hello world");
}
}
HelloWorld.java - Navigator
Members
<empty>
HelloWorld
HelloWorld()
main(String[] args)
Output
22:1
INS
Digite aqui para pesquisar
POR
12:48
14/02/2021

HelloWorld - Apache NetBeans IDE 12.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config> 344,7/485,0MB

Projects - Projetos | Files | Services

- HelloWorld
  - Source Packages
    - helloworld
      - HelloWorld.java
  - Test Packages
  - Libraries
  - Test Libraries

HelloWorld.java - Navigator

Members <empty>

- HelloWorld
  - HelloWorld()
  - main(String[] args)

HelloWorld.java

Source History

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package helloworld;
7
8  /**
9   *
10  * @author malex
11  */
12 public class HelloWorld {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         System.out.println("hello world");
19     }
20
21 }
22
```

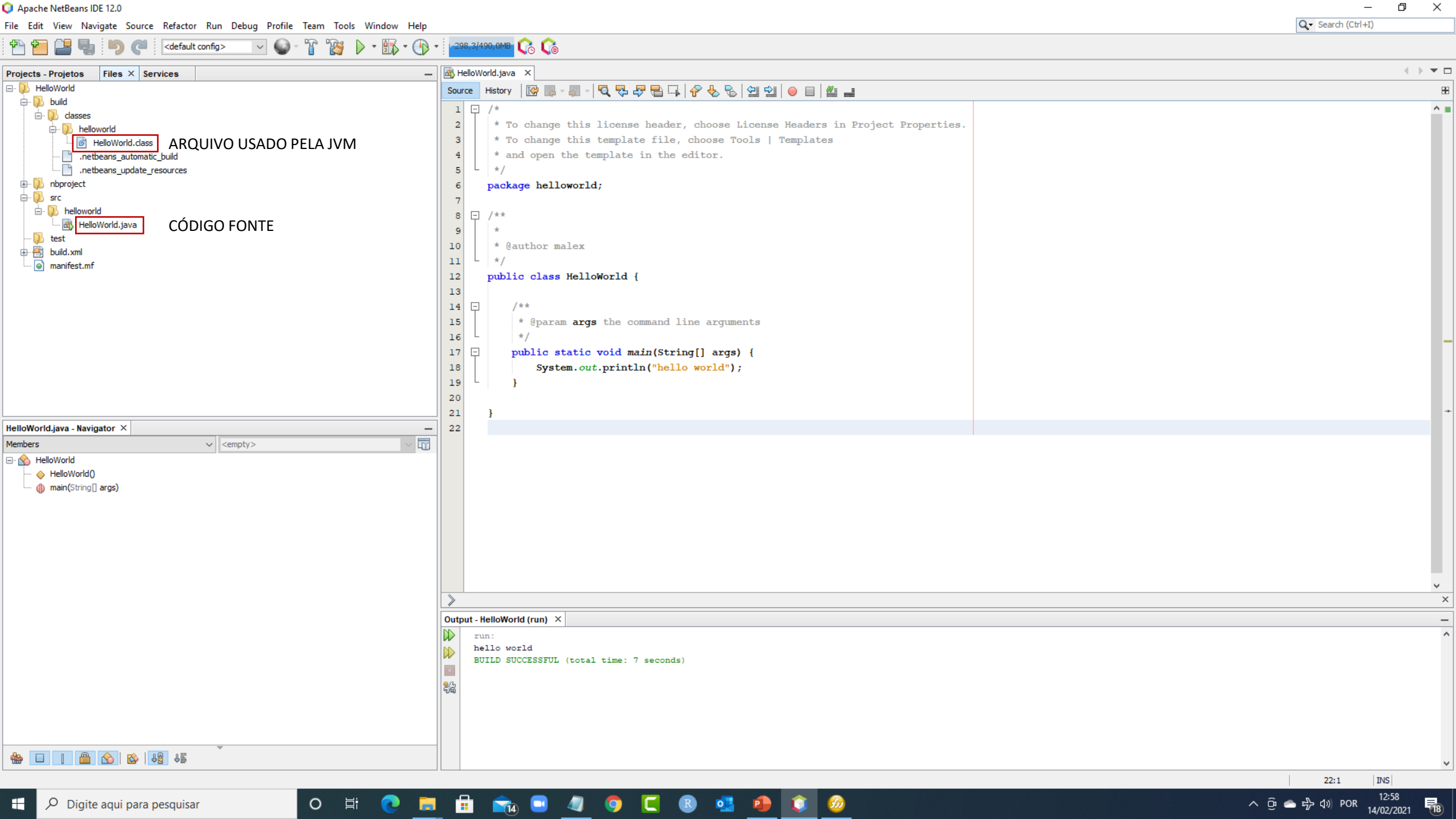
Output - HelloWorld (run)

```
run:
hello world
BUILD SUCCESSFUL (total time: 7 seconds)
```

22:1 INS Windows (CRLF)

Digite aqui para pesquisar

POR 12:49 14/02/2021


Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
298,3/490,0MB
Projects - Projetos
Files
Services
HelloWorld
build
classes
helloworld
HelloWorld.class
ARQUIVO USADO PELA JVM
.netbeans_automatic_build
.netbeans_update_resources
nbproject
src
helloworld
HelloWorld.java
CÓDIGO FONTE
test
build.xml
manifest.mf
HelloWorld.java - Navigator
Members
<empty>
HelloWorld
HelloWorld()
main(String[] args)
HelloWorld.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package helloworld;
7
8 /**
9 *
10 * @author malex
11 */
12 public class HelloWorld {
13
14 /**
15 * @param args the command line arguments
16 */
17 public static void main(String[] args) {
18 System.out.println("hello world");
19 }
20
21 }
22
Output - HelloWorld (run)
run:
hello world
BUILD SUCCESSFUL (total time: 7 seconds)
22:1
INS
Digite aqui para pesquisar
POR
12:58
14/02/2021