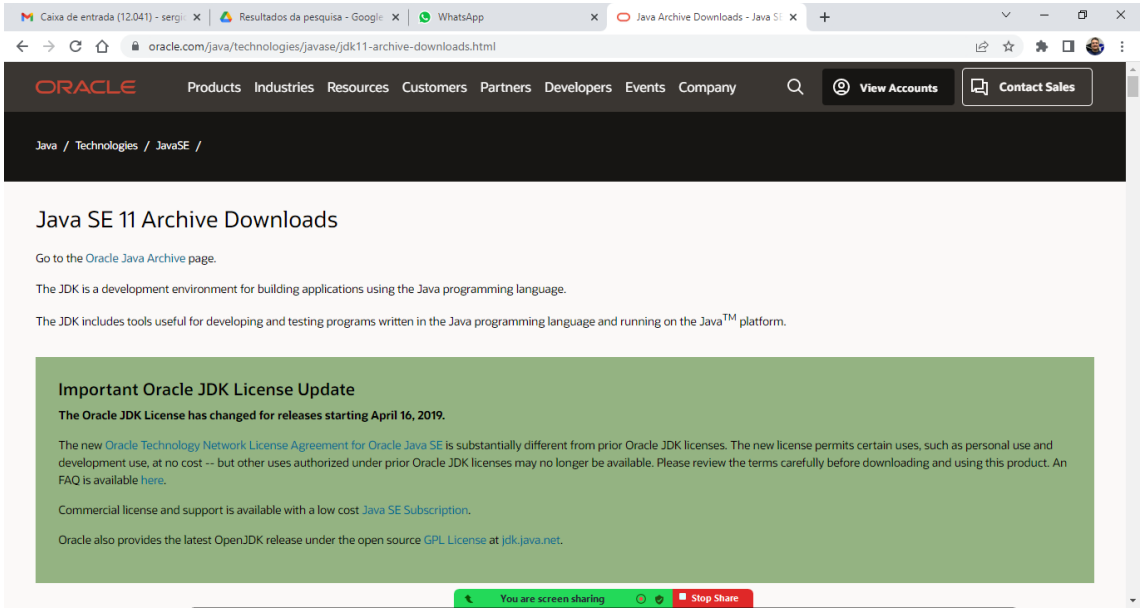
Setup do ambiente:

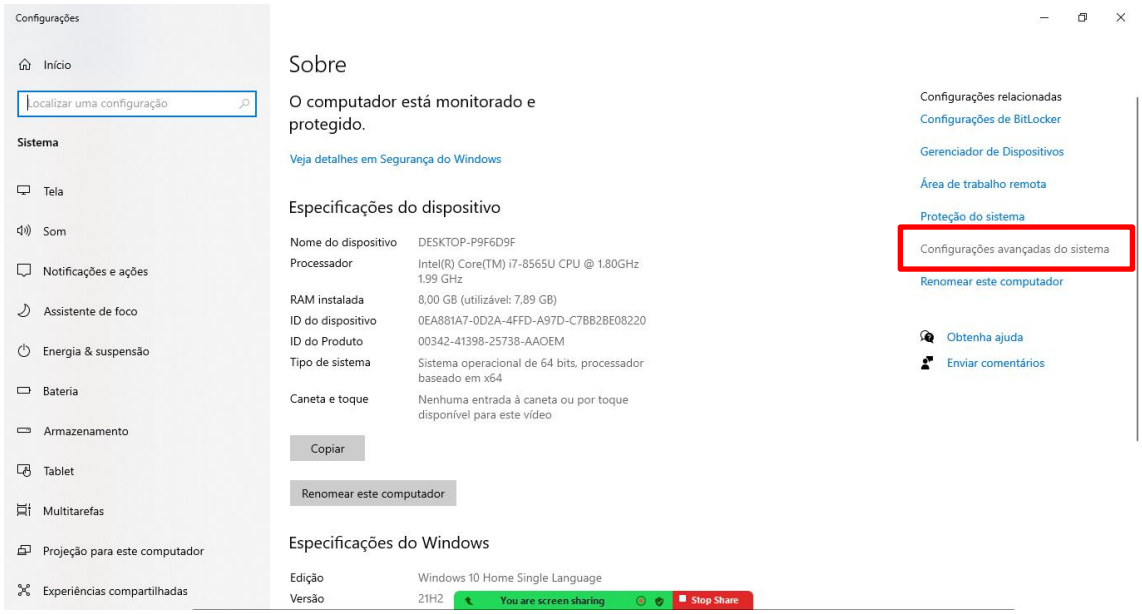
JDK – Kit de desenvolvimento Java

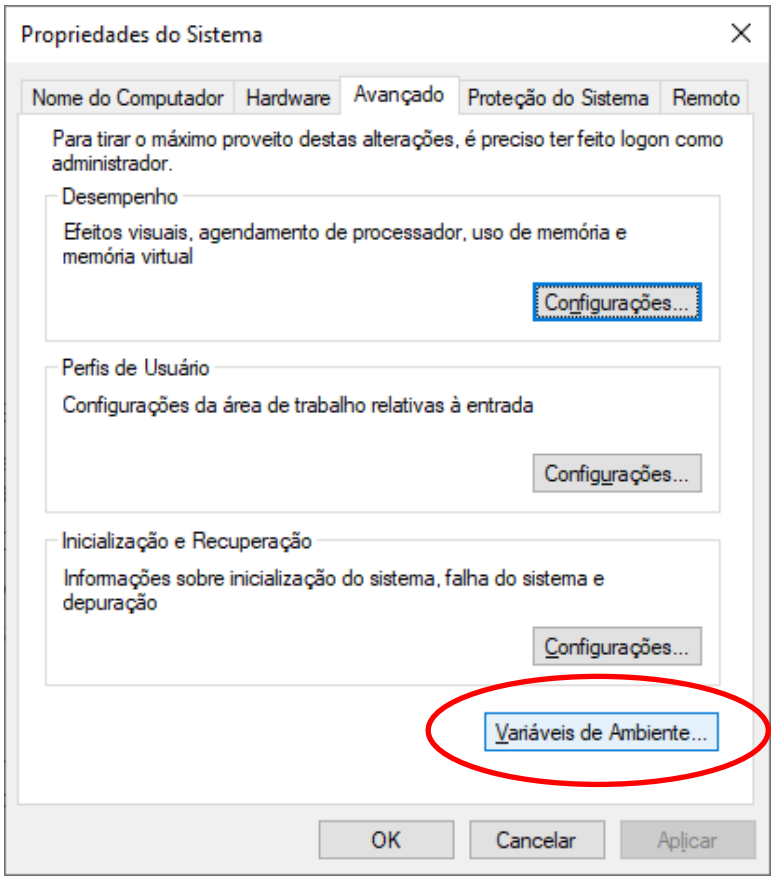
Conjunto de bibliotecas para execução de projetos Java.
https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html

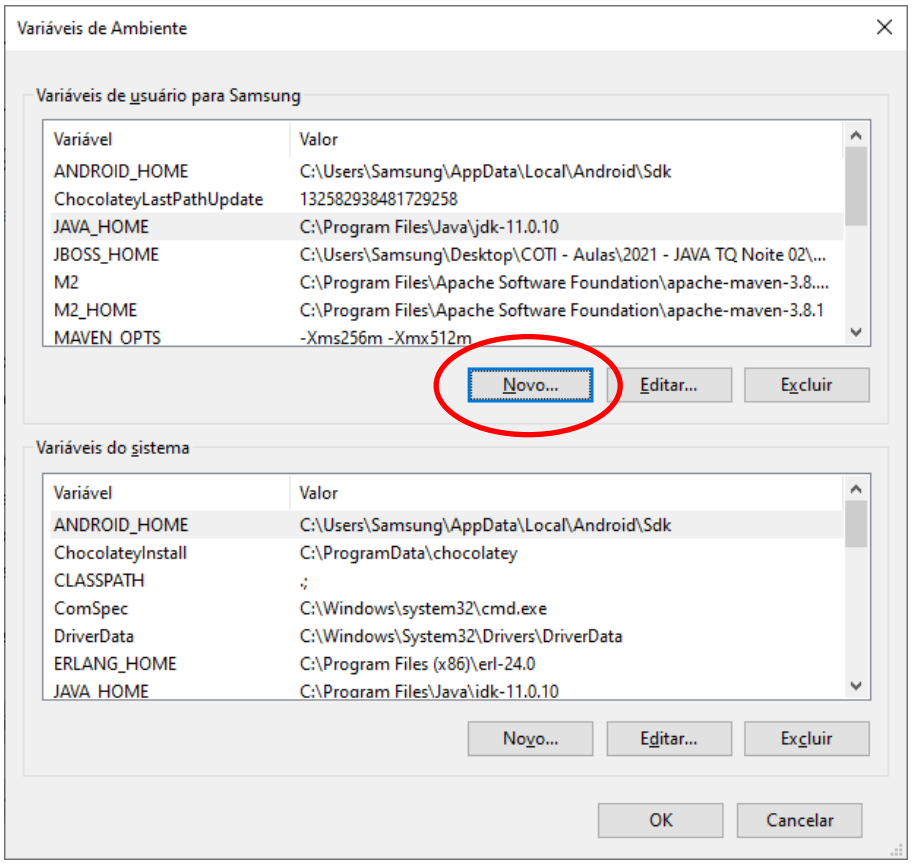


Precisamos configurar uma variável de ambiente apontando para a pasta de instalação do JDK:

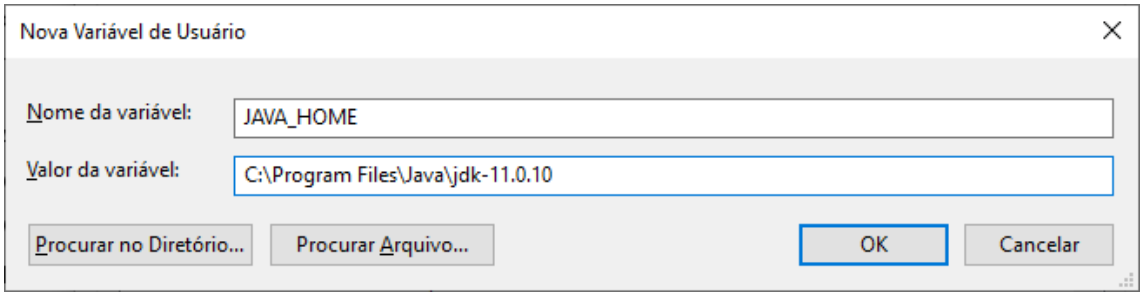
Acesse:
Sistema / Configurações avançadas do sistema:




Propriedades do Sistema
Nome do Computador
Hardware
Avançado
Proteção do Sistema
Remoto
Para tirar o máximo proveito destas alterações, é preciso ter feito logon como administrador.
Desempenho
Efeitos visuais, agendamento de processador, uso de memória e memória virtual
Configurações...
Perfis de Usuário
Configurações da área de trabalho relativas à entrada
Configurações...
Inicialização e Recuperação
Informações sobre inicialização do sistema, falha do sistema e depuração
Configurações...
Variáveis de Ambiente...
OK
Cancelar
Aplicar


Variáveis de Ambiente
Variáveis de usuário para Samsung
Variável
Valor
ANDROID_HOME
C:\Users\Samsung\AppData\Local\Android\Sdk
ChocolateyLastPathUpdate
132582938481729258
JAVA_HOME
C:\Program Files\Java\jdk-11.0.10
JBOSS_HOME
C:\Users\Samsung\Desktop\COTI - Aulas\2021 - JAVA TQ Noite 02\...
M2
C:\Program Files\Apache Software Foundation\apache-maven-3.8....
M2_HOME
C:\Program Files\Apache Software Foundation\apache-maven-3.8.1
MAVEN_OPTS
-Xms256m -Xmx512m
Novo...
Editar...
Excluir
Variáveis do sistema
Variável
Valor
ANDROID_HOME
C:\Users\Samsung\AppData\Local\Android\Sdk
ChocolateyInstall
C:\ProgramData\chocolatey
CLASSPATH
.;
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
ERLANG_HOME
C:\Program Files (x86)\erl-24.0
JAVA_HOME
C:\Program Files\Java\jdk-11.0.10
Novo...
Editar...
Excluir
OK
Cancelar

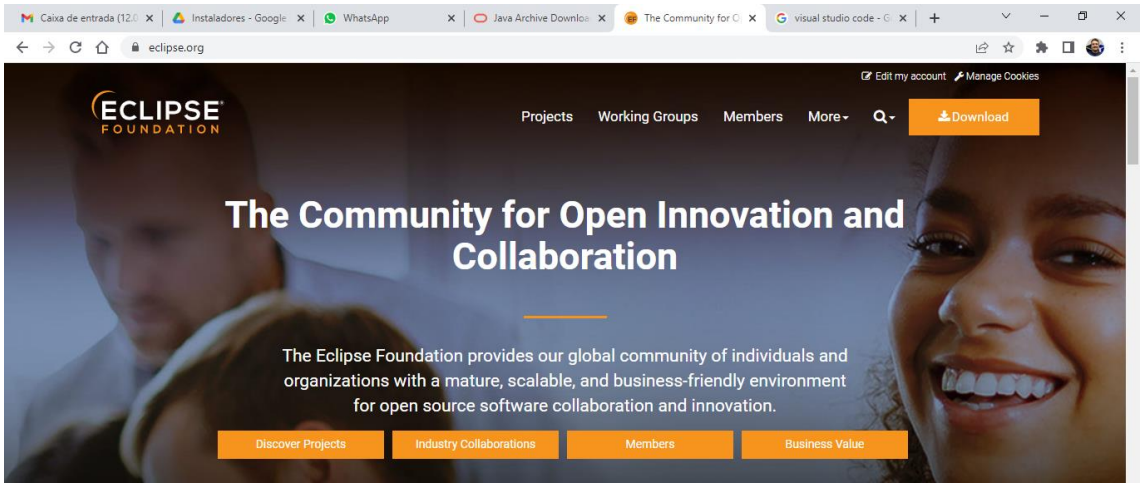
Configure a variável abaixo:

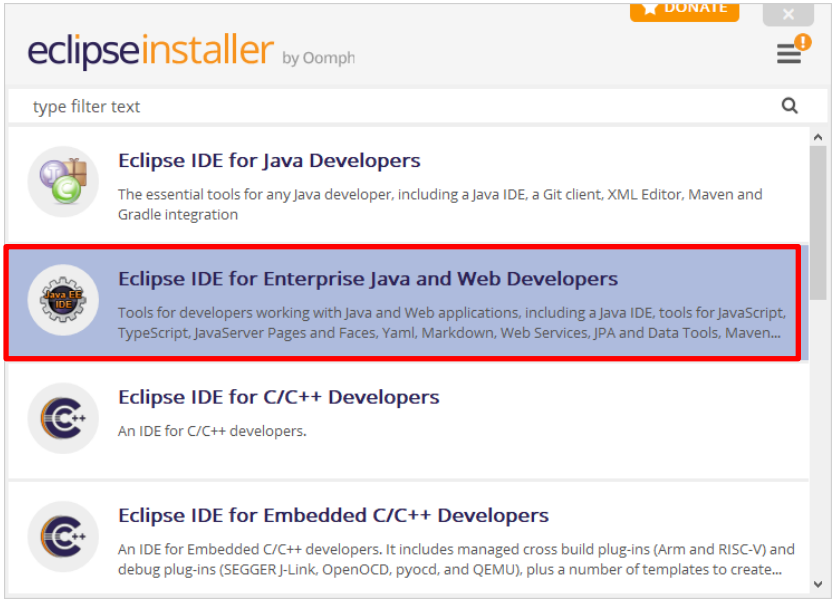


Instalando o Eclipse:

IDE: Ambiente integrado de desenvolvimento
Software utilizado para desenvolvimento dos projetos Java.

https://www.eclipse.org/

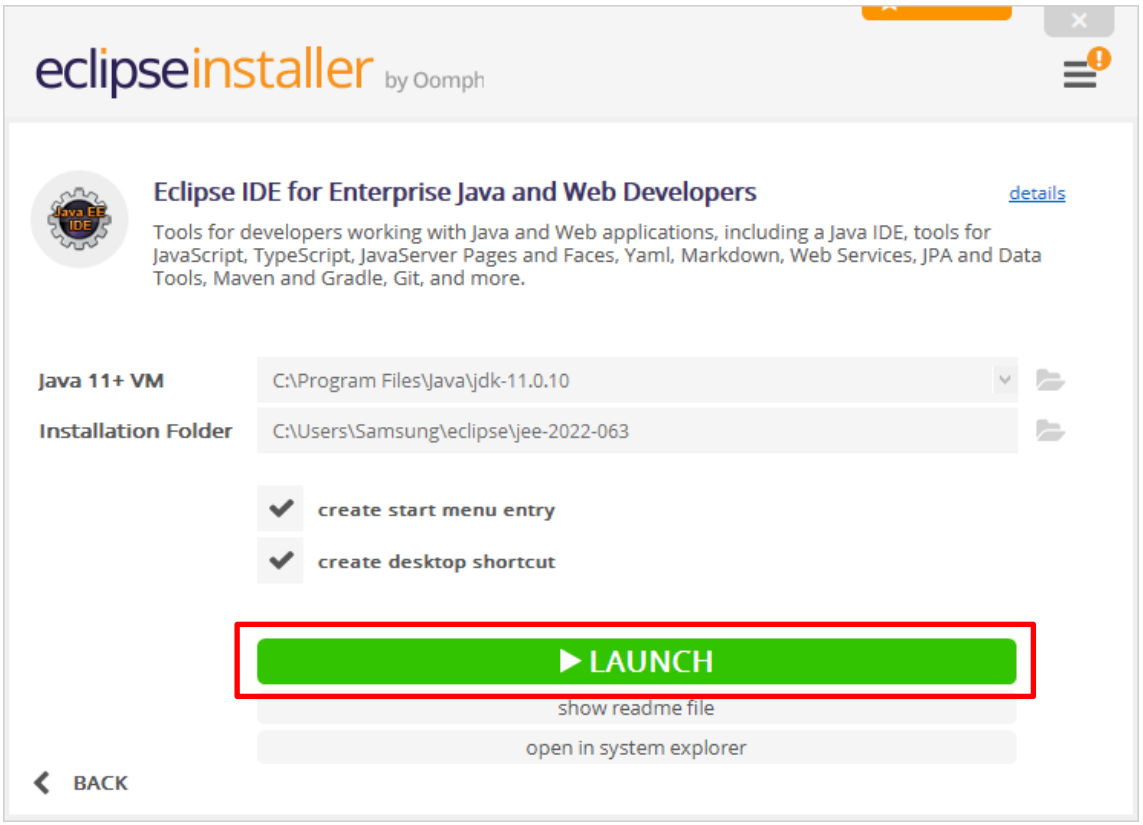





DONATE
eclipseinstaller by Oomph
Eclipse IDE for Enterprise Java and Web Developers
details
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.
Java 11+ VM
C:\Program Files\Java\jdk-11.0.10
Installation Folder
C:\Users\Samsung\eclipse\jee-2022-063
create start menu entry
create desktop shortcut
INSTALL
BACK

Instalando:

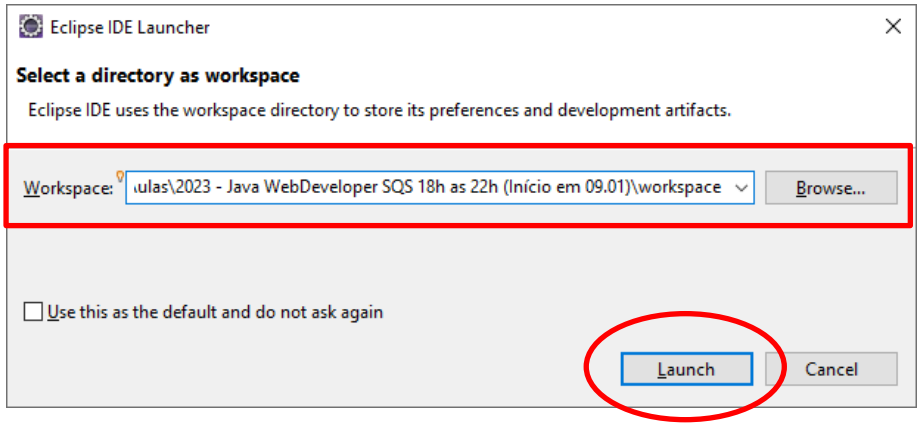

DONATE
eclipseinstaller by Oomph
Eclipse IDE for Enterprise Java and Web Developers
details
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.
Java 11+ VM
C:\Program Files\Java\jdk-11.0.10
Installation Folder
C:\Users\Samsung\eclipse\jee-2022-063
create start menu entry
create desktop shortcut
INSTALLING
Cancel Installation
BACK

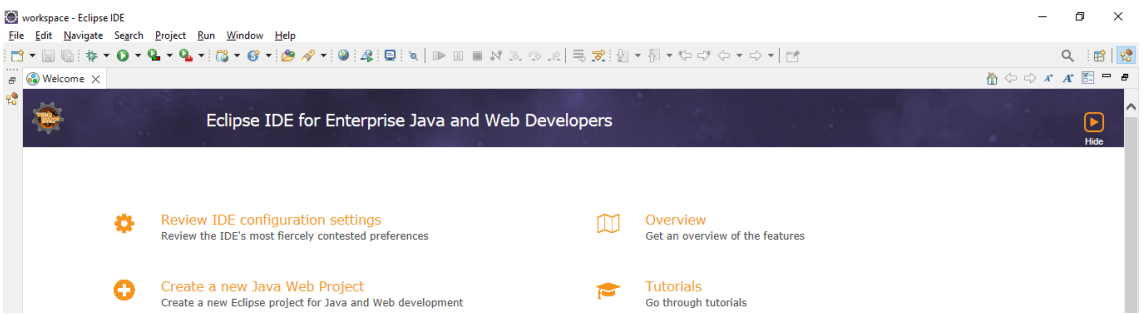


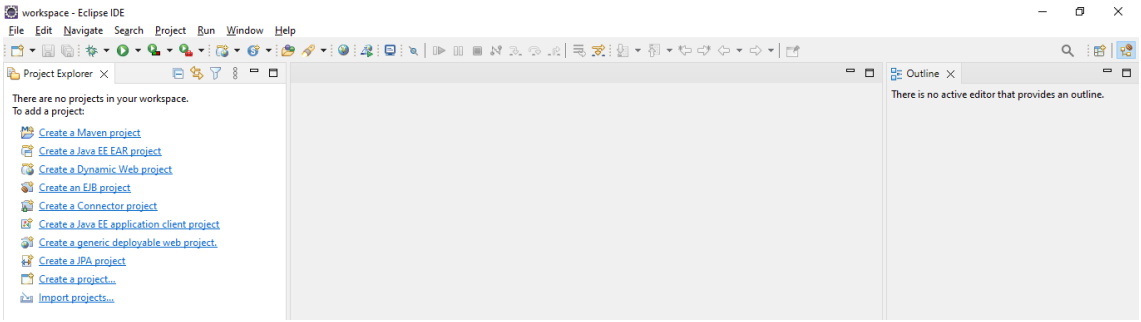
Abrindo o Eclipse e selecionando o workspace:

Workspace

Pasta para armazenar os projetos do curso e configurações da IDE.

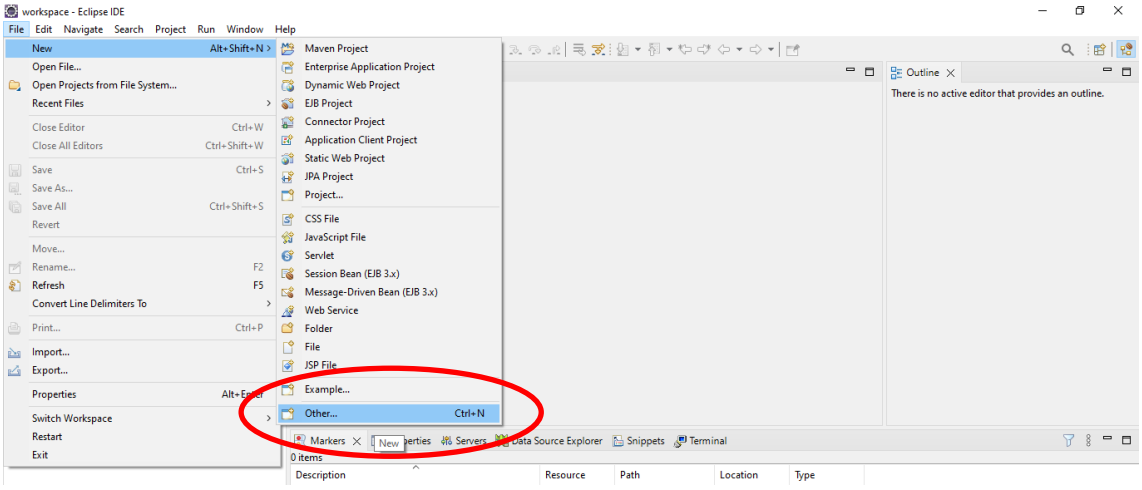


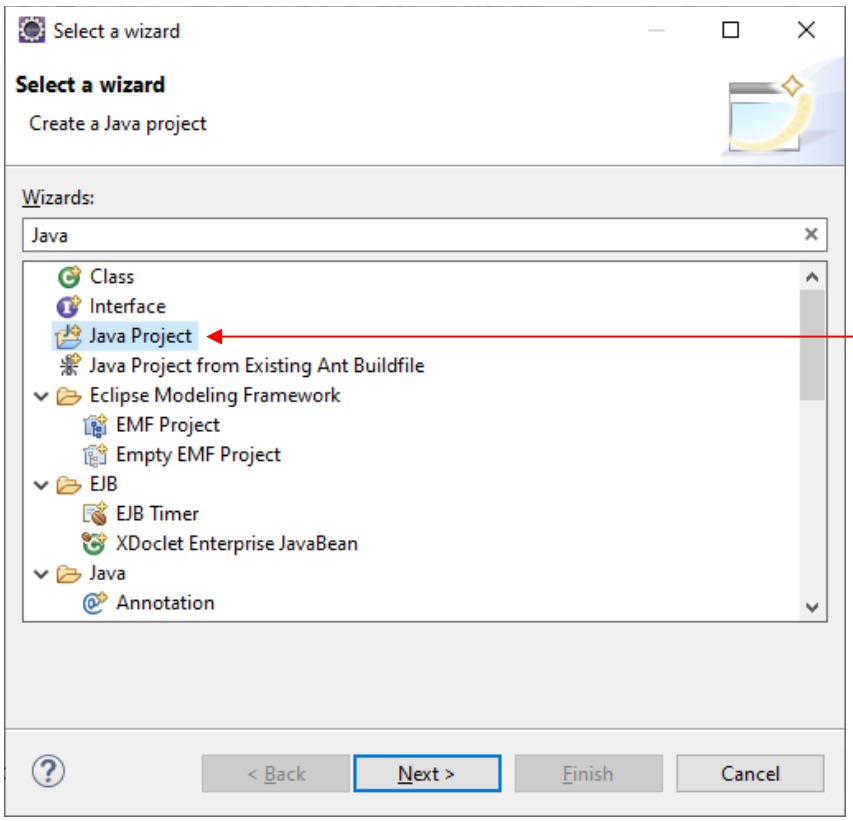




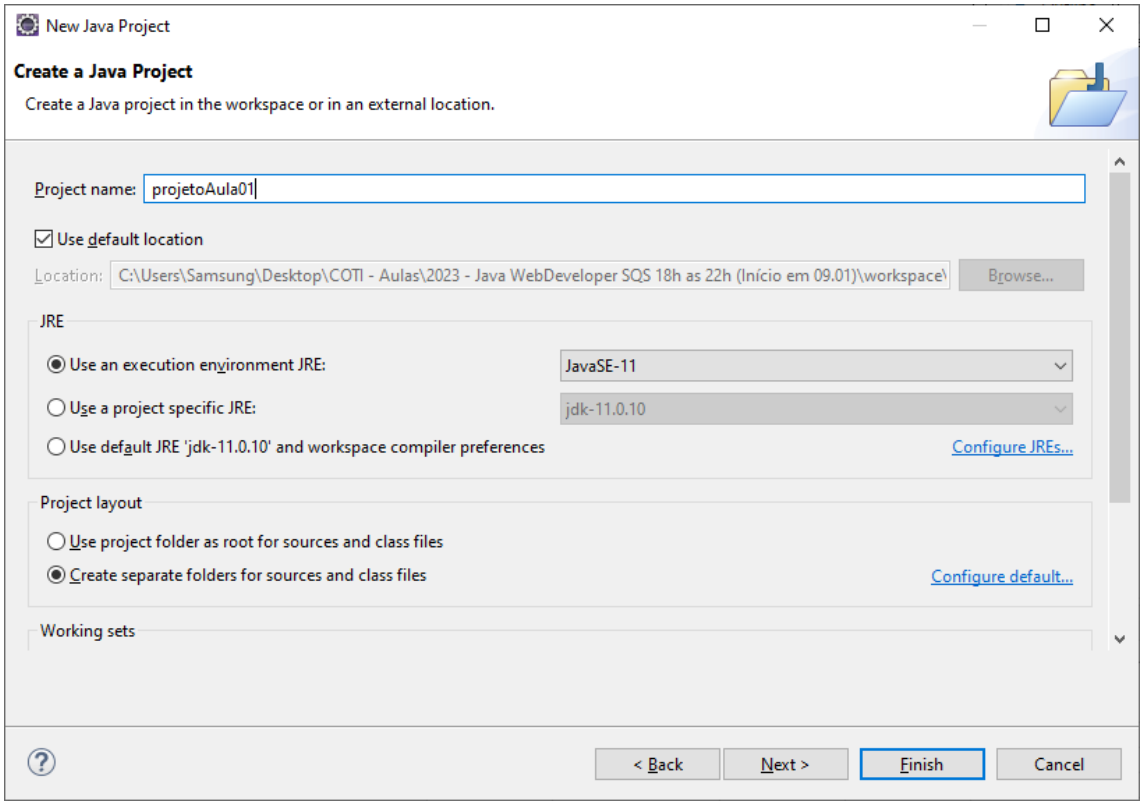
Criando um projeto Java local:

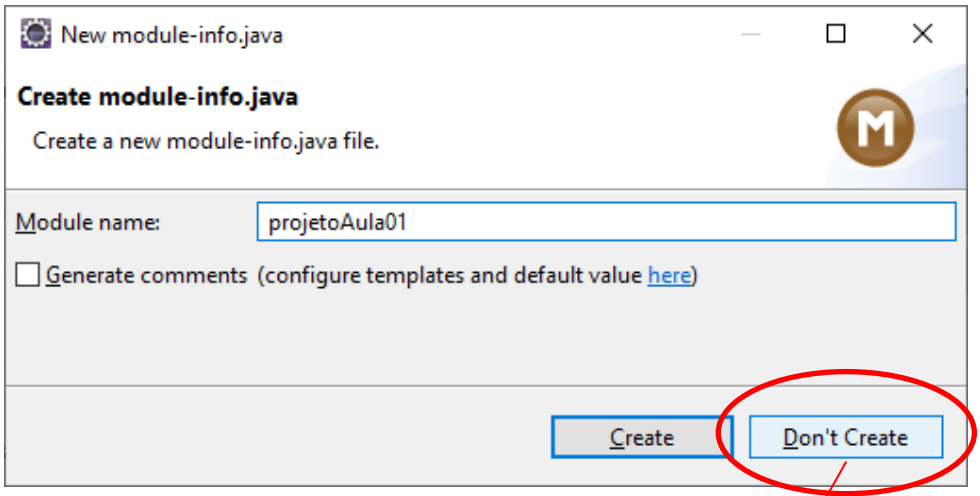
File / New / Other

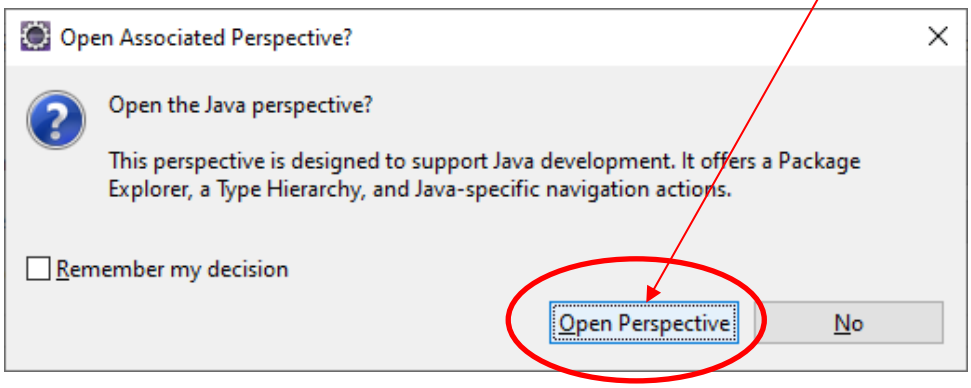




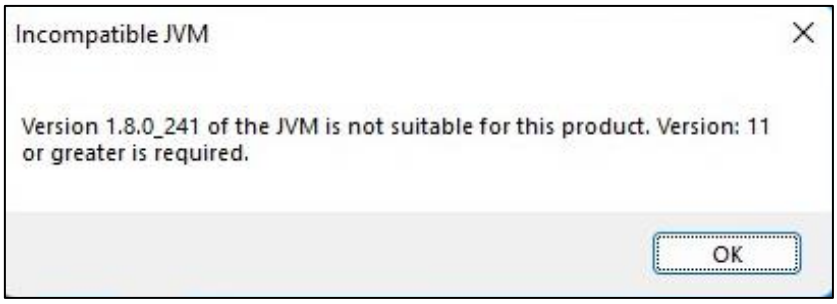
Nome: **projetoAula01**



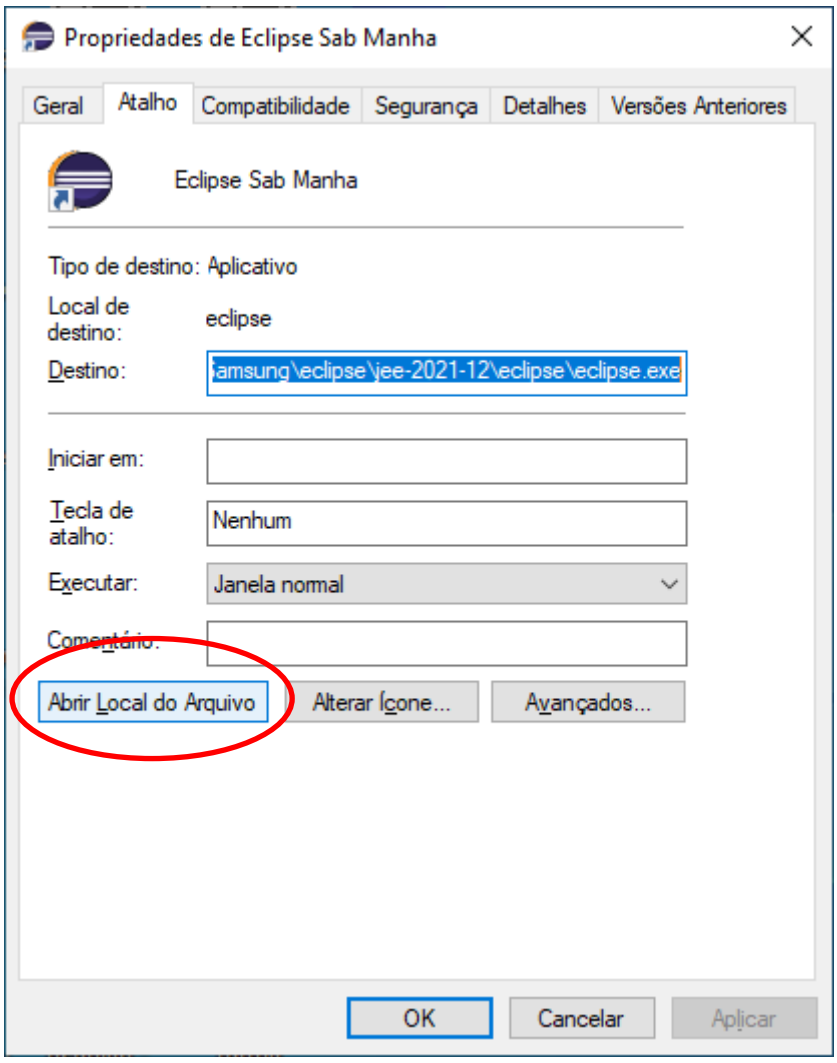




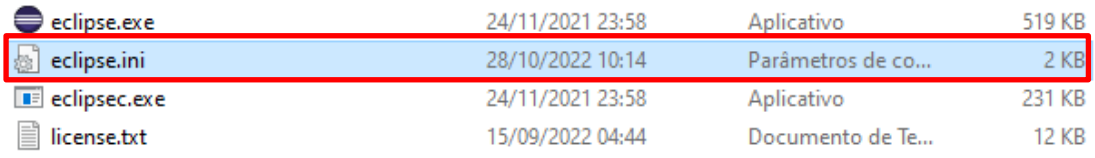
Corrigindo erro de incompatibilidade da IDE com a versão do Java:



Clique nas propriedades do atalho:

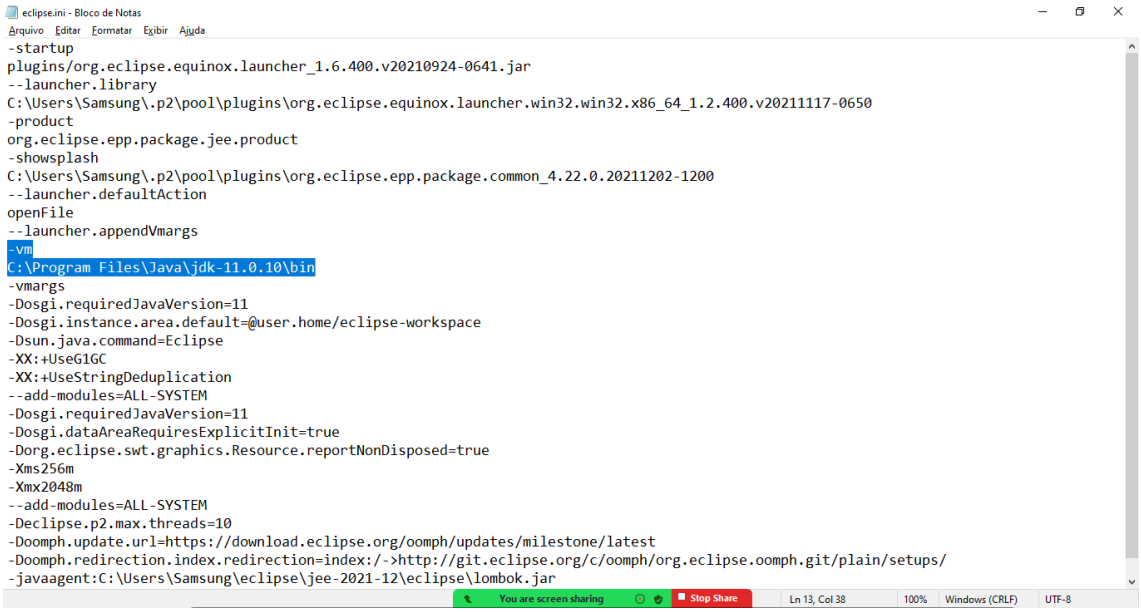


Abra o arquivo **eclipse.ini**



Insira a chave abaixo:

-vm
C:\Program Files\Java\jdk-11.0.10\bin



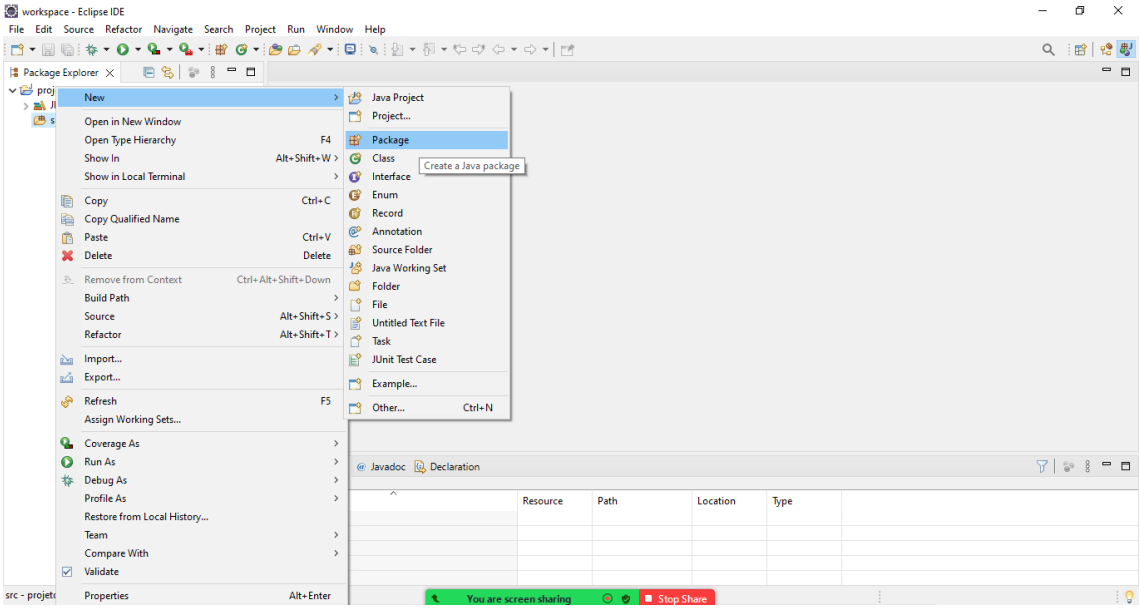
Voltando ao nosso projeto...

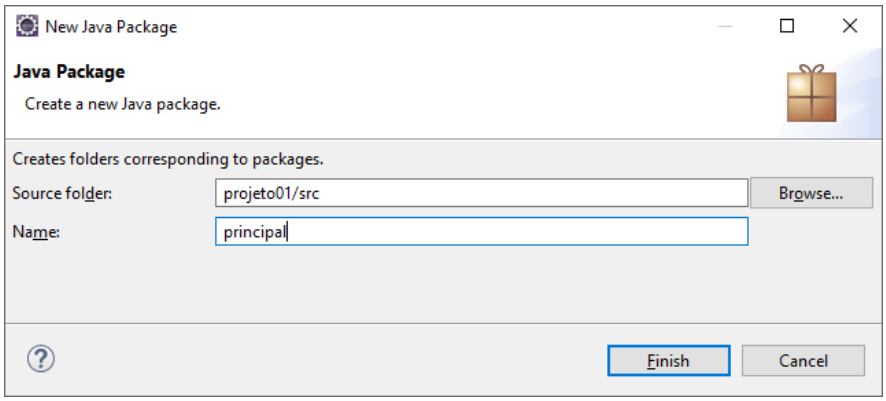
/src/

Pasta raiz de código fonte do projeto. A partir dela iremos criar pacotes (pastas) e dentro destes as classes e artefatos do projeto.

/src/**principal**/

Este será o primeiro package (pacote) do projeto.



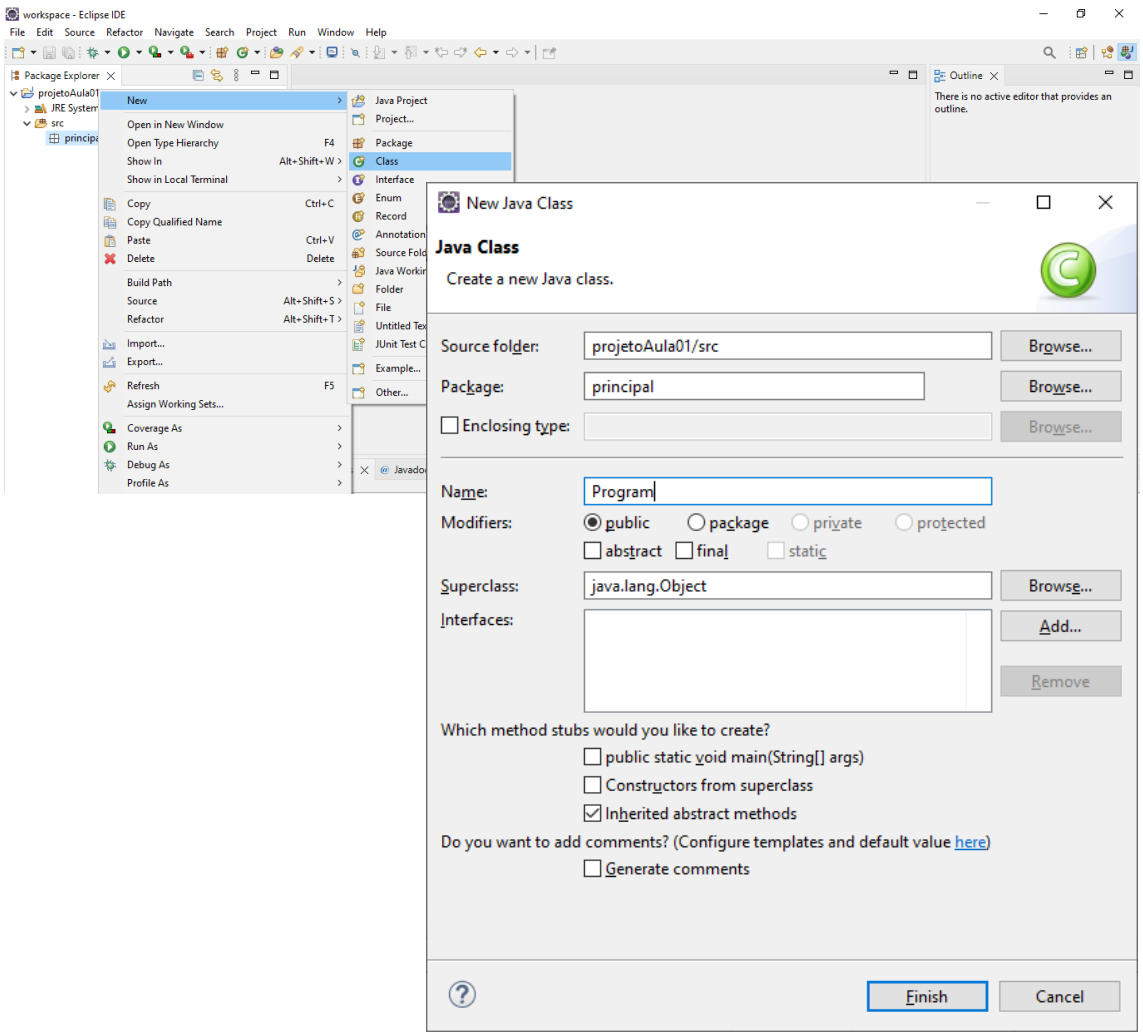


Classe

É um artefato de programação Java composta, basicamente, de: **atributos (dados)** e **métodos (funções)**. Nomes de classes sempre devem começar com a primeira letra em caixa alta.

Iremos criar uma classe dentro do pacote principal contendo um método (função) para executar o projeto.

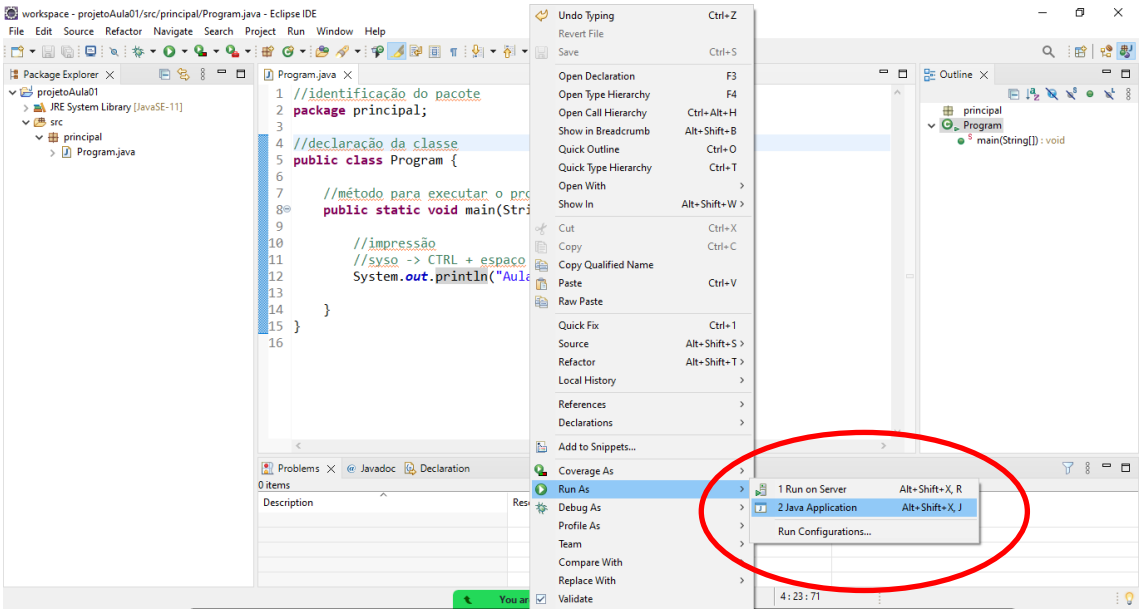
/src/principal/Program.java

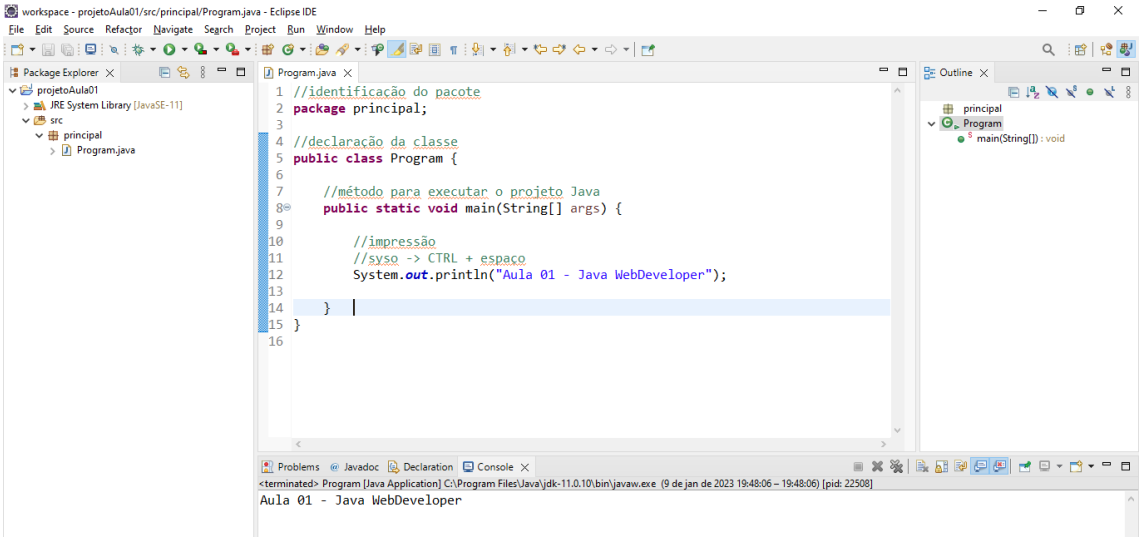


Criando a primeira classe do projeto:

```
//definição do pacote
package principal;
//definição da classe
public class Program {
	//método padrão para execução do projeto/classe
	public static void main(String[] args) {
		//imprimindo mensagem no console do eclipse..
		System.out.println
		("Aula 01 - Java Webdeveloper COTI Informática");
	}
}
```

Executando:
RUN AS / JAVA APPLICATION





Tarefa:
Escrever um programa em Java capaz de ler os dados de um cliente.
Este cliente deverá ser composto de:

- Id (identificador numérico)
- Nome
- Telefone
- Email
- Cpf

O sistema deverá fazer a captura dos dados do cliente e gravar este registro em um arquivo TXT na máquina do usuário.

Desenvolvimento baseado em camadas

Cada classe criada em um projeto deve ser feita pensando em uma única responsabilidade, por exemplo, podemos criar em um projeto uma classe para armazenar os dados de um cliente e uma outra classe somente para pegar essas e gravá-los em um arquivo. Mas nunca uma única classe que faça tudo em um só local.

Neste exemplo, iremos criar uma classe inicialmente para apenas armazenar os dados de um cliente. Esta classe é chamada de **entidade**.

/src/entities/**Cliente.java**

/entities

Cliente
- idCliente : Integer - nome : String - telefone : String - email : String - cpf : String

Atributos (Campos)

Responsabilidade:
Modelar e armazenar os dados de um cliente.

Modificadores de visibilidade:

`public`
Define acesso / visibilidade total para um elemento (classe, método etc.)

`protected`
Define acesso somente dentro da própria classe ou por meio de herança.

default
Define acesso somente dentro do pacote (package) onde a classe foi criada.

private
Define acesso somente dentro da própria classe, é o tipo de visibilidade mais restritivo do Java.

```
package entities;

public class Cliente {

	//atributos (campos)
	private Integer idCliente;
	private String nome;
	private String telefone;
	private String email;
	private String cpf;
}
```

Para que possamos acessar cada um dos atributos, precisamos fazer com o Java forneça métodos através do qual possamos atribuir valor aos campos ou ler o valor dos campos.

Estes métodos são chamados de **set** (Entrada de dados)
e **get** (Saída de dados).

Encapsulamento

Nome dado a prática em POO (Programação Orientado a Objetos) para a prática de declaramos atributos privados em uma classe e criando para cada atributo um método (função) de entrada / saída de dados.

- Estas funções são chamadas de **setters** e **getters**.

Tecla de atalho:

CTRL + SHIFT + F
Formatação / endentação do código.

```
package entities;

public class Cliente {

	// atributos (campos)
	private Integer idCliente;
	private String nome;
	private String telefone;
	private String email;
	private String cpf;
```

```java
// Encapsulamento (set/get)
// Para cada atributo da classe iremos criar uma função
// de entrada de dados e saída de dados
// set -> nome da função para entrada de dados
// get -> nome da função para saída de dados

// função para preenchimento do campo idCliente
public void setIdCliente(Integer idCliente) {
	this.idCliente = idCliente;
}

// função para saída (retorno) do campo idCliente
public Integer getIdCliente() {
	return idCliente;
}

// função para preenchimento do campo nome
public void setNome(String nome) {
	this.nome = nome;
}

// função para saída (retorno) do campo nome
public String getNome() {
	return nome;
}

// função para preenchimento do campo telefone
public void setTelefone(String telefone) {
	this.telefone = telefone;
}

// função para saída (retorno) do campo telefone
public String getTelefone() {
	return telefone;
}

// função para preenchimento do campo email
public void setEmail(String email) {
	this.email = email;
}

// função para saída (retorno) do campo email
public String getEmail() {
	return email;
}

// função para preenchimento do campo cpf
public void setCpf(String cpf) {
	this.cpf = cpf;
}
```

```
	// função para saída (retorno) do campo cpf
	public String getCpf() {
		return cpf;
	}
}
```

Tecla de atalho:

CTRL + SHIFT + O
Faz o import das classes e pacotes necessários.

Objeto (variável de instância)

Consiste basicamente em uma **variável de instância** inicializada a partir do espaço de memória (**construtor**) de uma classe.

Exemplo:

```
Cliente cliente = new Cliente();
```

[Classe] [Objeto] (Variável) [Inicializando / Instanciando]

Através da variável de instancia de cliente (objeto)
podemos acessar os métodos **set** e **get**.

```
//identificação do pacote
package principal;

import java.io.PrintWriter;

import entities.Cliente;

//declaração da classe
public class Program {

	//método para executar o projeto Java
	public static void main(String[] args) {

		//impressão
		//syso -> CTRL + espaço
		System.out.println("\n *** CADASTRO DE CLIENTES *** \n");

		//Criando um objeto para a classe Cliente
		Cliente cliente = new Cliente();

		//preechendo os dados do cliente (set -> entrada)
		cliente.setIdCliente(1);
```

```
                cliente.setNome("Sergio Mendes");
                cliente.setEmail("sergio.coti@gmail.com");
                cliente.setTelefone("21 96957-5900");
                cliente.setCpf("123.456.789-00");

                //gravando os dados do cliente em um arquivo
                try { //tentativa

                        //abrindo um arquivo em modo de escrita
                        PrintWriter printWriter
                                = new PrintWriter("c:\\clientes\\cliente.txt");

                        //escrevendo no arqquivo:
                        printWriter.write("\nID DO CLIENTE...: "
                                + cliente.getIdCliente());
                        printWriter.write("\nNOME.....: "
                                + cliente.getNome());
                        printWriter.write("\nTELEFONE.....: "
                                + cliente.getTelefone());
                        printWriter.write("\nEMAIL.....: "
                                + cliente.getEmail());
                        printWriter.write("\nCPF.....: "
                                + cliente.getCpf());

                        printWriter.flush(); //salvar as alterações do arquivo
                        printWriter.close(); //fechar o arquivo

                        System.out.println("Dados do cliente
                                        cadastrados com sucesso!");
                }
                catch(Exception e) { //captura do erro

                        System.out.println("Erro ao cadastrar cliente.");
                        e.printStackTrace(); //imprimir o log (detalhamento) do erro
                }

        }

}
```

Tecla de atalho:

CTRL + SHIFT + S
Salvar todos os arquivos abertos.

CTRL + F11
Executa o método Main().

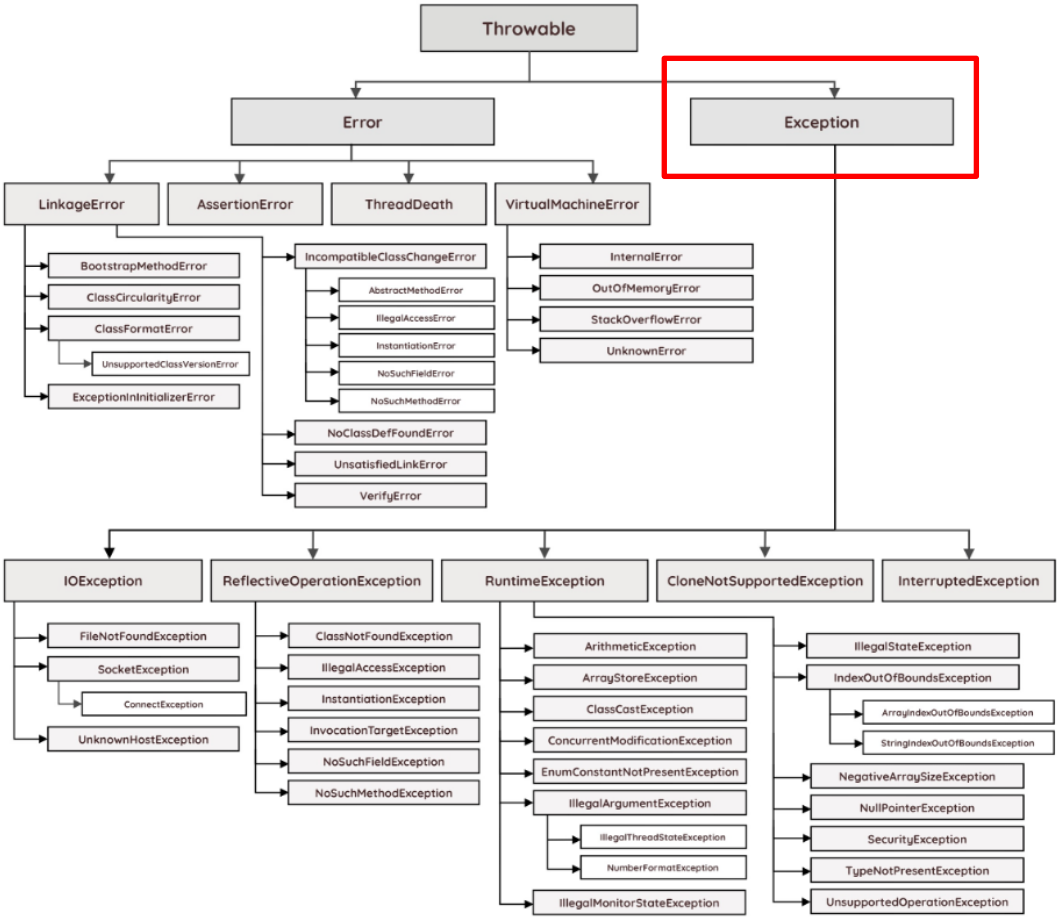
Tratamento de exceções:

Exceções são erros que ocorrem em tempo de execução em um programa Java, ou seja, não são erros de compilação, mas sim erros que ocorrem somente na execução do projeto (Exceções).

Para fazermos o tratamento destes erros podemos usar um bloco denominado **try** e **catch**.

Exception

É uma classe Java capaz de capturar erros em tempo de execução (exceções).



Saida do programa: