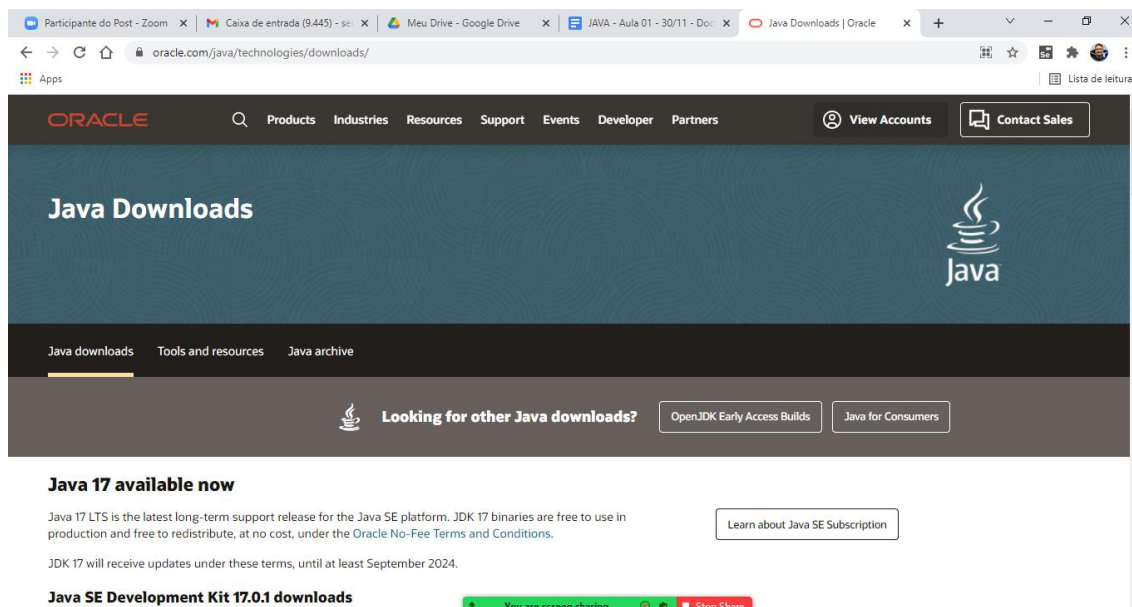


JDK – KIT de desenvolvimento Java

<https://www.oracle.com/java/technologies/downloads/>

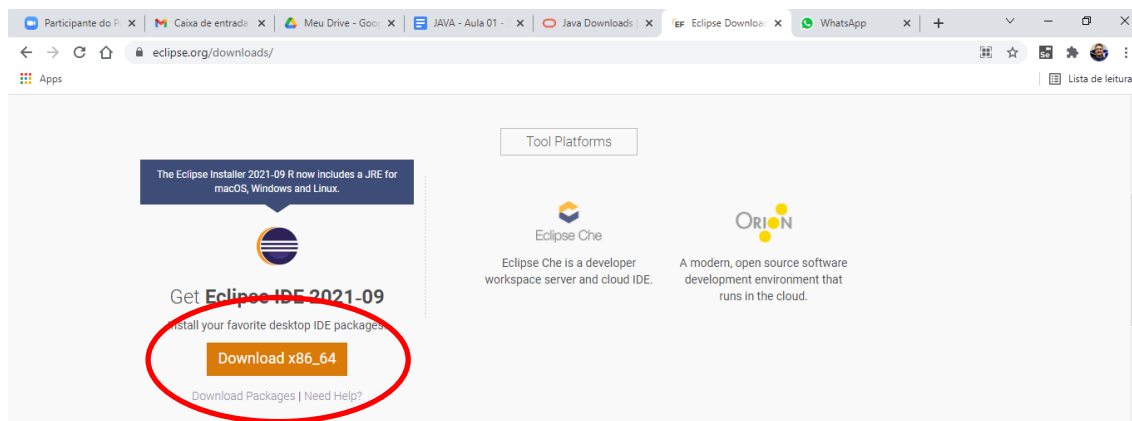
Versão 11.



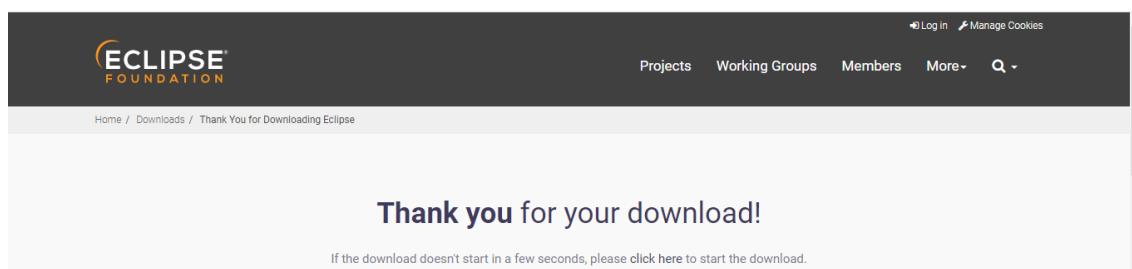
Eclipse

IDE (Software utilizado para desenvolvimento dos projetos)

<https://www.eclipse.org/downloads/>



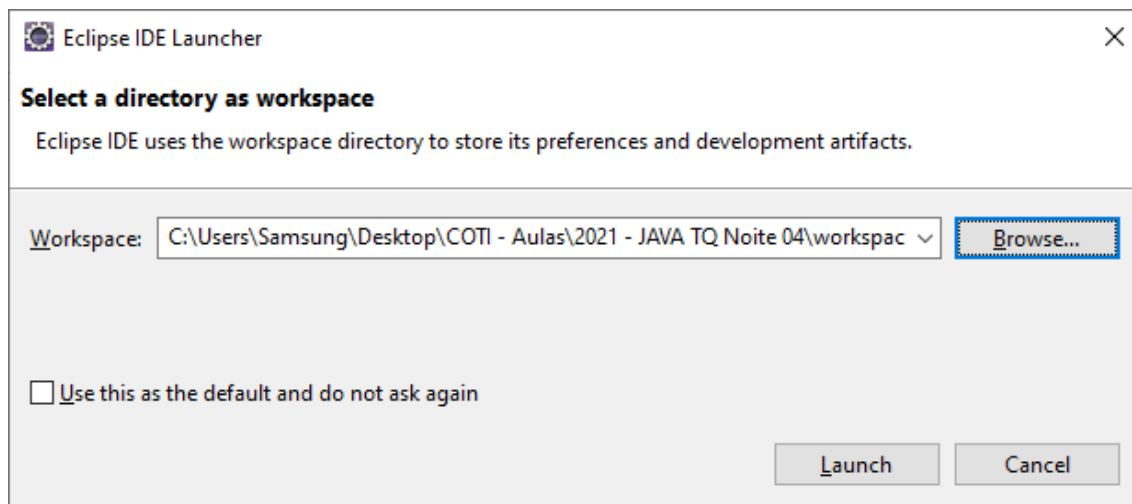
<https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2021-09/R/eclipse-inst-jre-win64.exe>



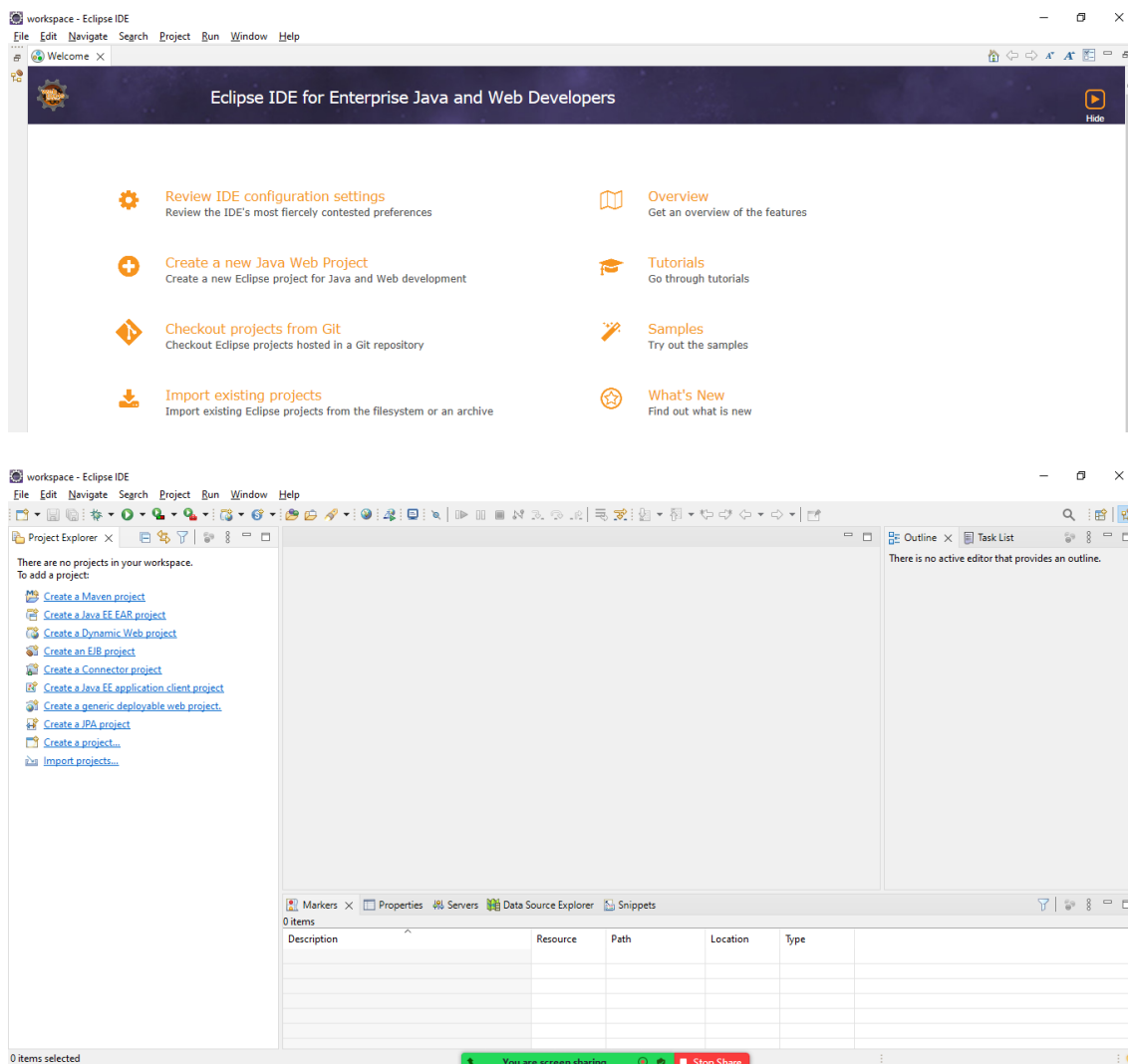


Criando um workspace:

Pasta de trabalho para criar os projetos.

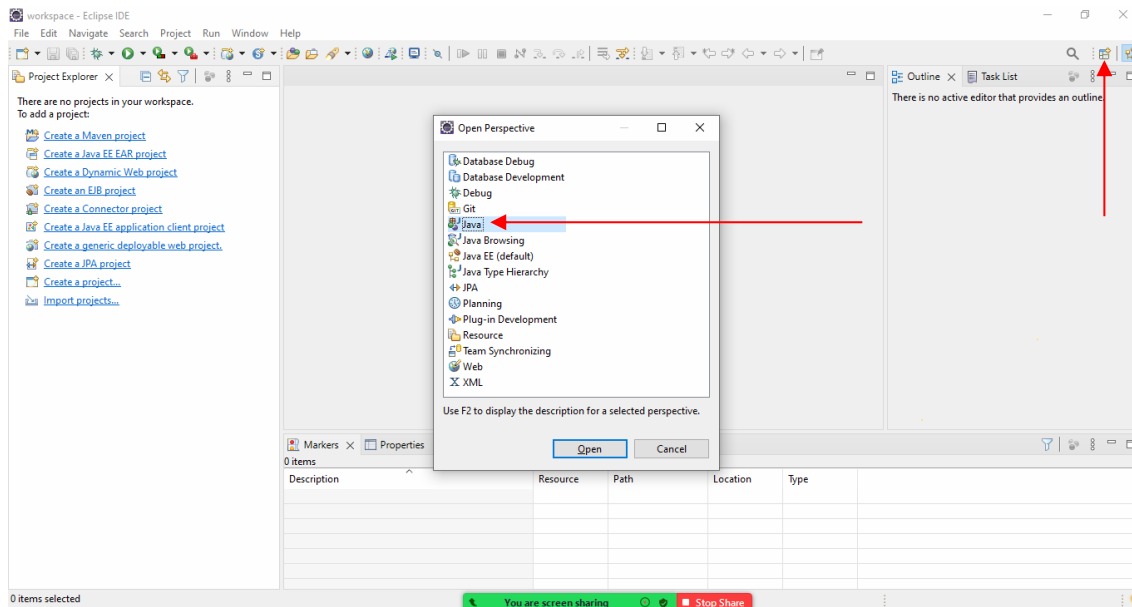


Tela inicial do eclipse:



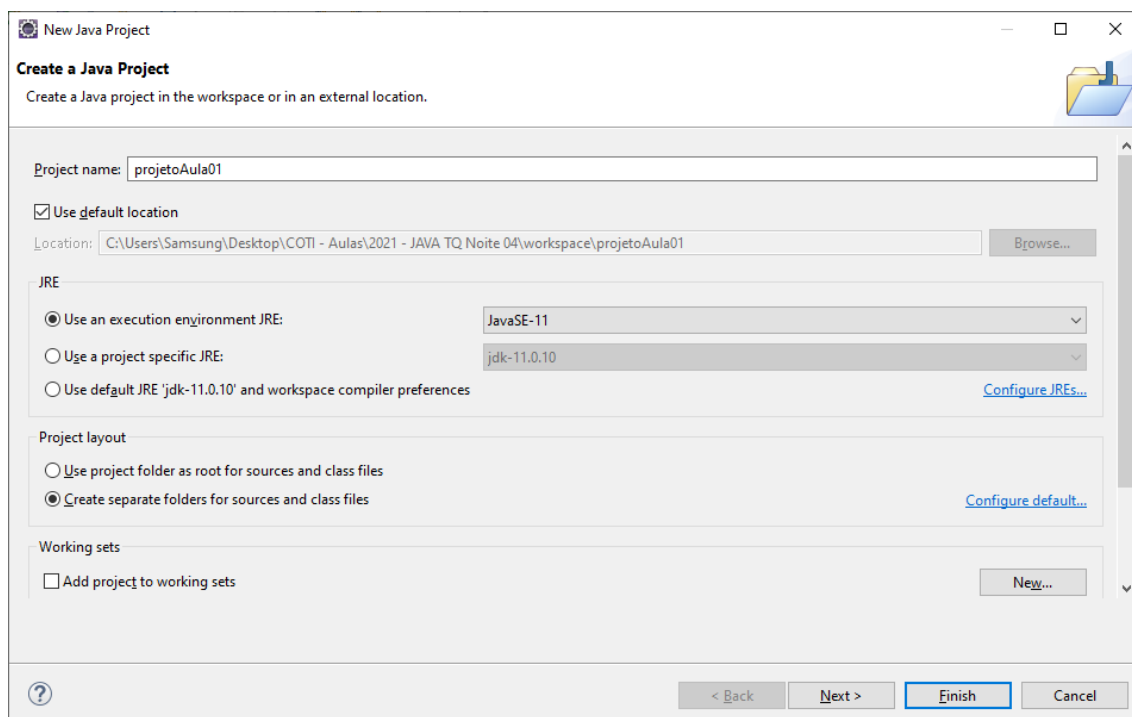
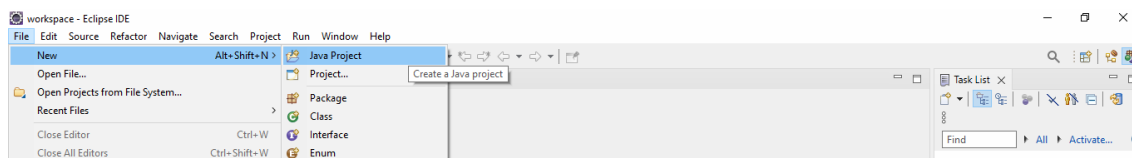
Mudança a perspectiva do eclipse:

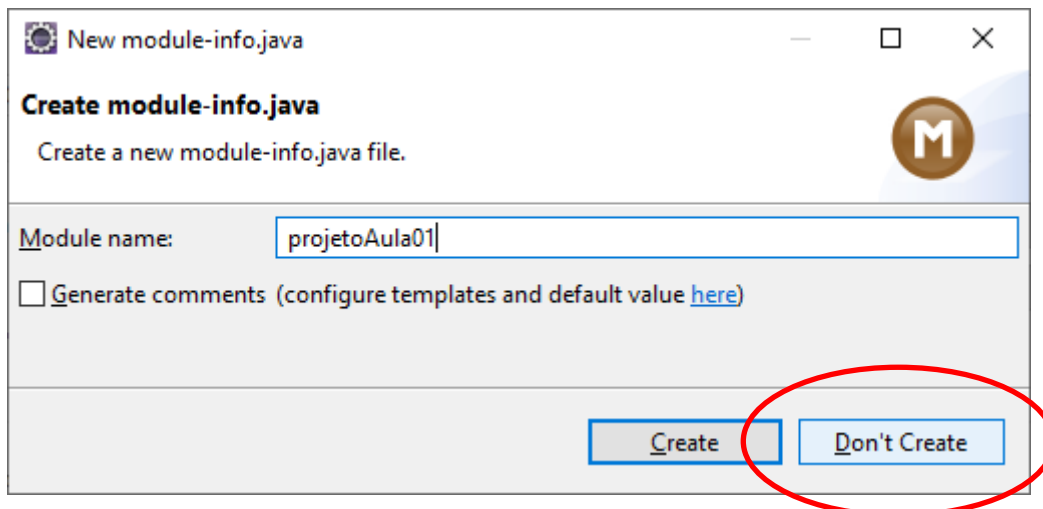
Configuração das ferramentas e painéis que o eclipse exige.



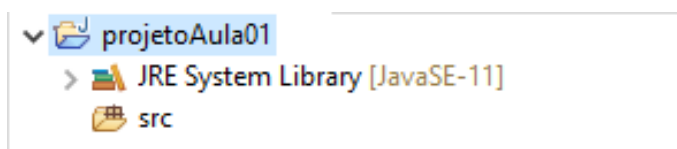
Criando o projeto JAVA:

FILE / NEW / JAVA PROJECT





Projeto criado:

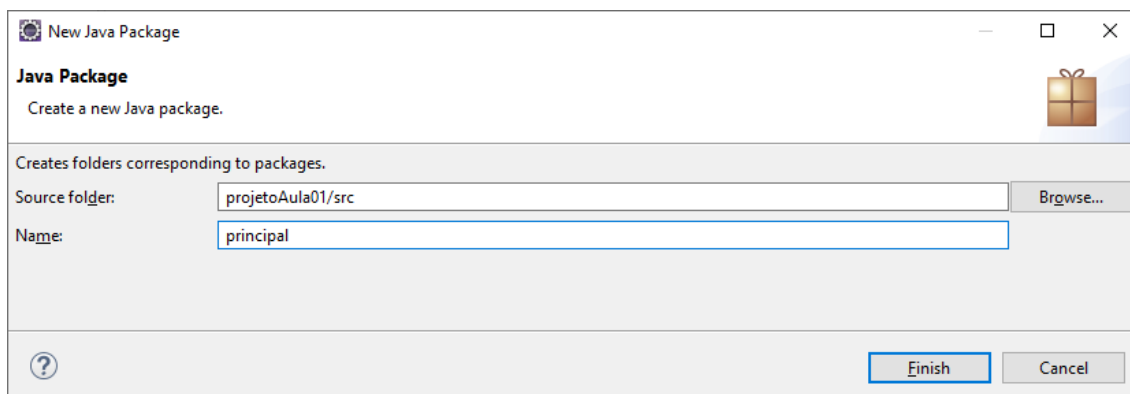
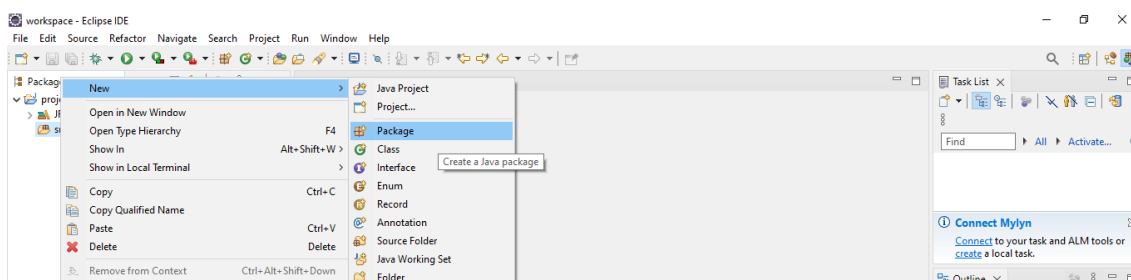


/src/

Source Folder

É a pasta a partir do qual começamos a construir as classes, componentes etc. do projeto.

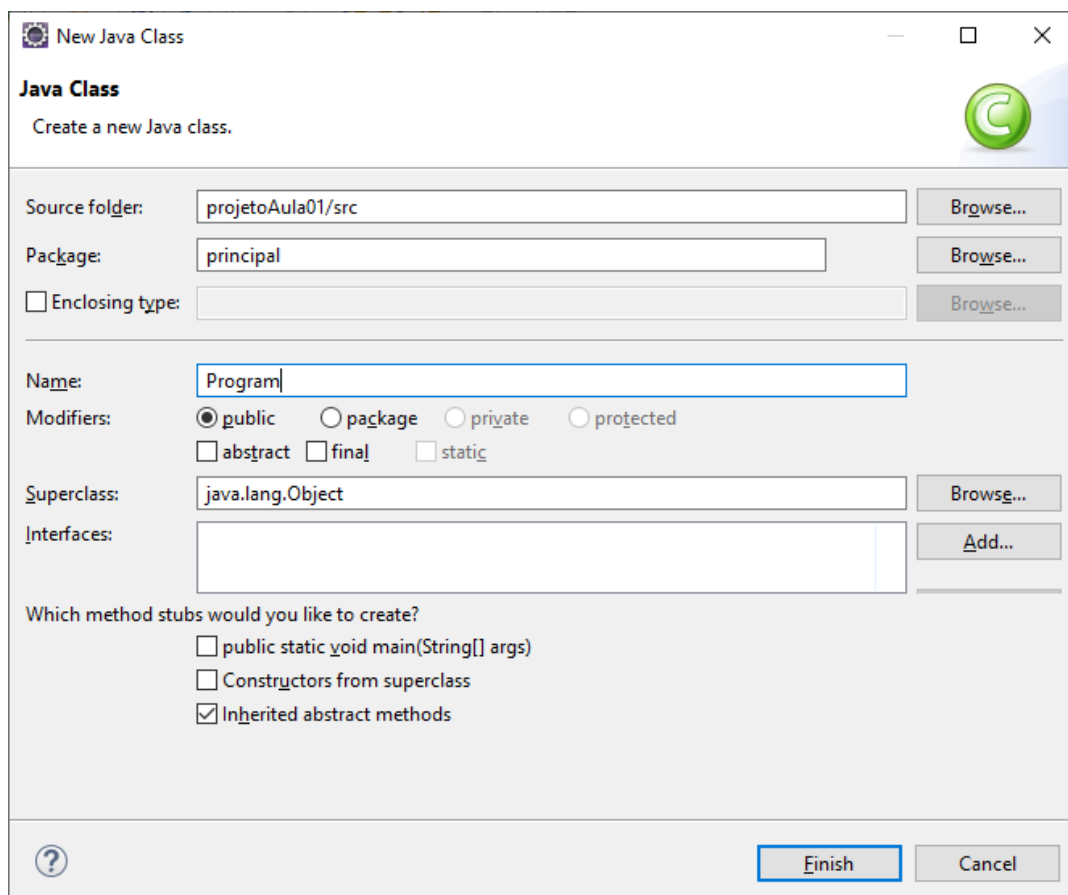
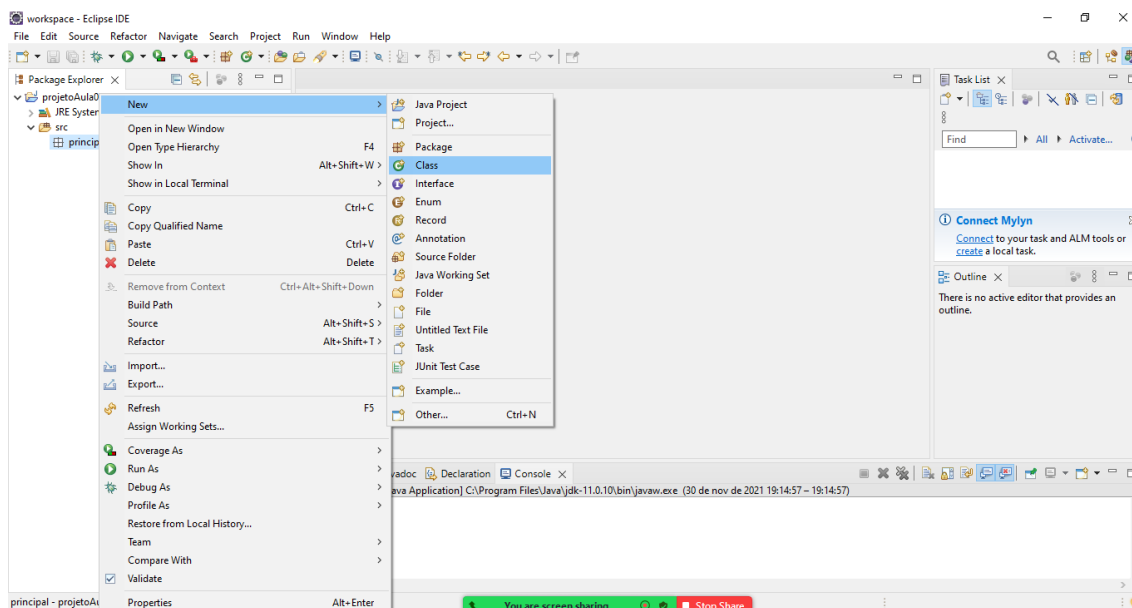
**** Regra:** Toda classe Java deve, por boa prática, ser criada dentro de um **pacote (PACKAGE)**

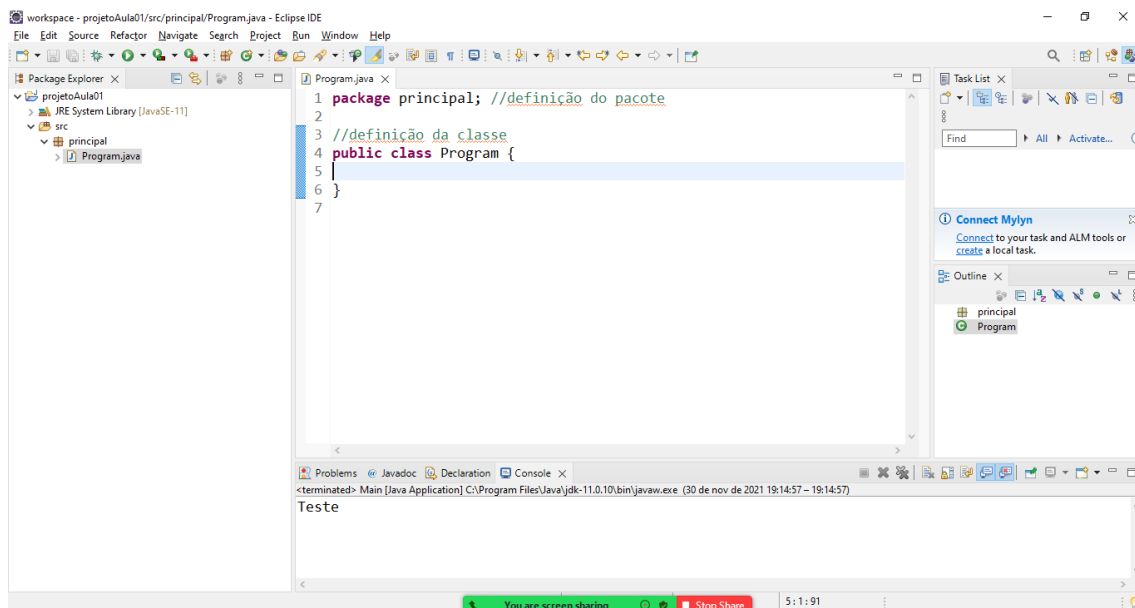


Dentro deste pacote, iremos criar uma classe Java que terá como objetivo executar o projeto.

/principal/**Program.java**

Classe que iremos utilizar para executar o projeto.





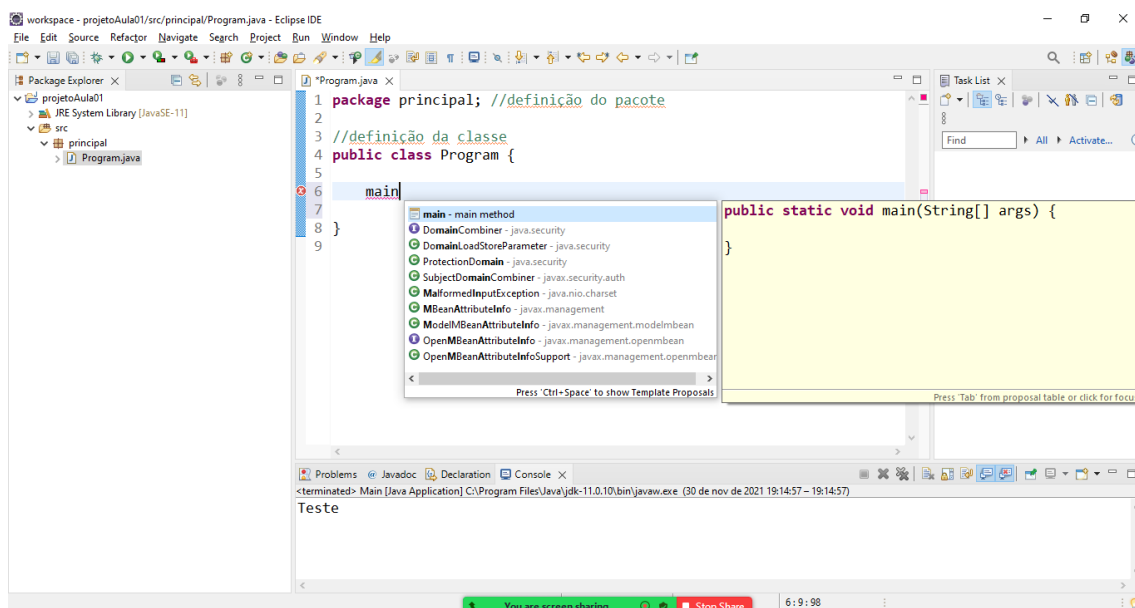
package

Define o nome do pacote onde a classe foi criada.

public

Modificador de visibilidade que define a classe como publica dentro do projeto, ou seja, a classe poderá ser acessada por qualquer outro componente dentro do projeto.

Iremos criar dentro desta classe um método (função) para executar o projeto. Este método tem uma assinatura padrão do JAVA e é chamado de **main()**



```
package principal; //definição do pacote

//definição da classe
public class Program {

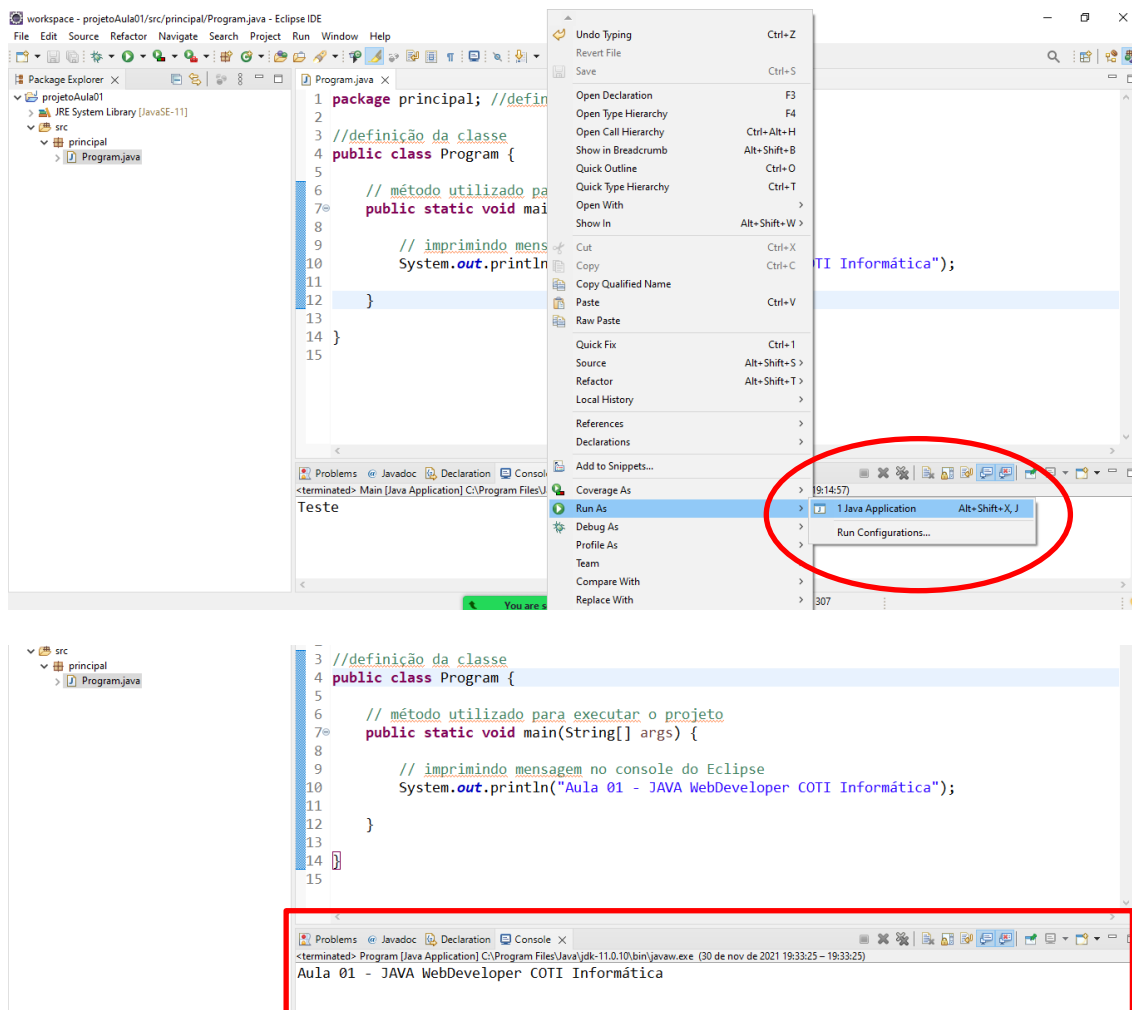
    // método utilizado para executar o projeto
    public static void main(String[] args) {

        // imprimindo mensagem no console do Eclipse
        System.out.println("Aula 01 - JAVA
                           WebDeveloper COTI Informática");
    }
}
```

Tecla de atalho:

CTRL + SHIFT + F (Formatar /indentar o código fonte)

Executando a classe **Program.java**



Tarefa:

Desenvolva um projeto em Java local que possa capturar os dados de um cliente informado pelo usuário da aplicação. Este cliente deverá ter os campos: id, nome, email e telefone.

O programa deverá solicitar ao usuário que entre com os dados, capturá-los e em seguida gravá-los em um arquivo de texto (TXT) na máquina do usuário.

Passo 1)

Iremos identificar e modelar as entidades do projeto.

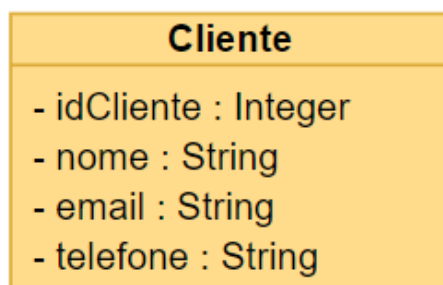
De acordo com o enunciado, precisamos ler os dados de um **Cliente**.

Este Cliente é composto pelos campos:

- id (numérico inteiro)
- nome (texto)
- email (texto)
- telefone (texto)

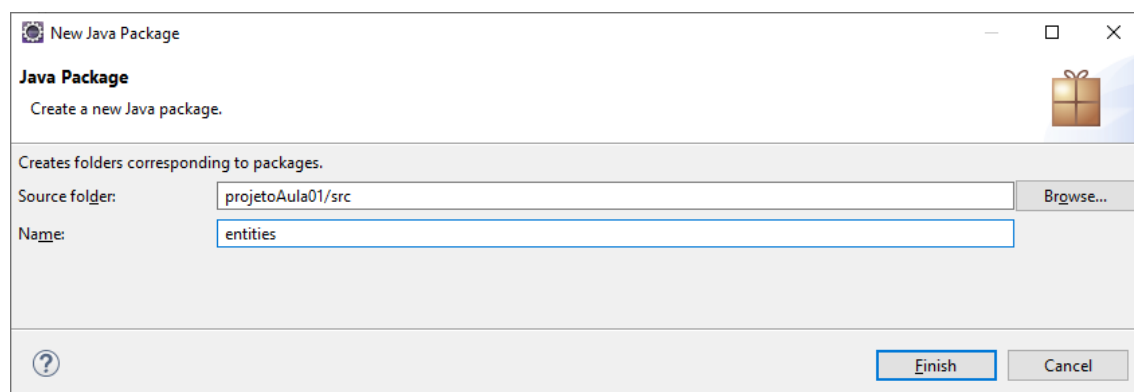
Para fazermos a modelagem destes dados iremos criar uma classe Java seguindo o padrão chamado de **JAVABEAN**, conforme abaixo:

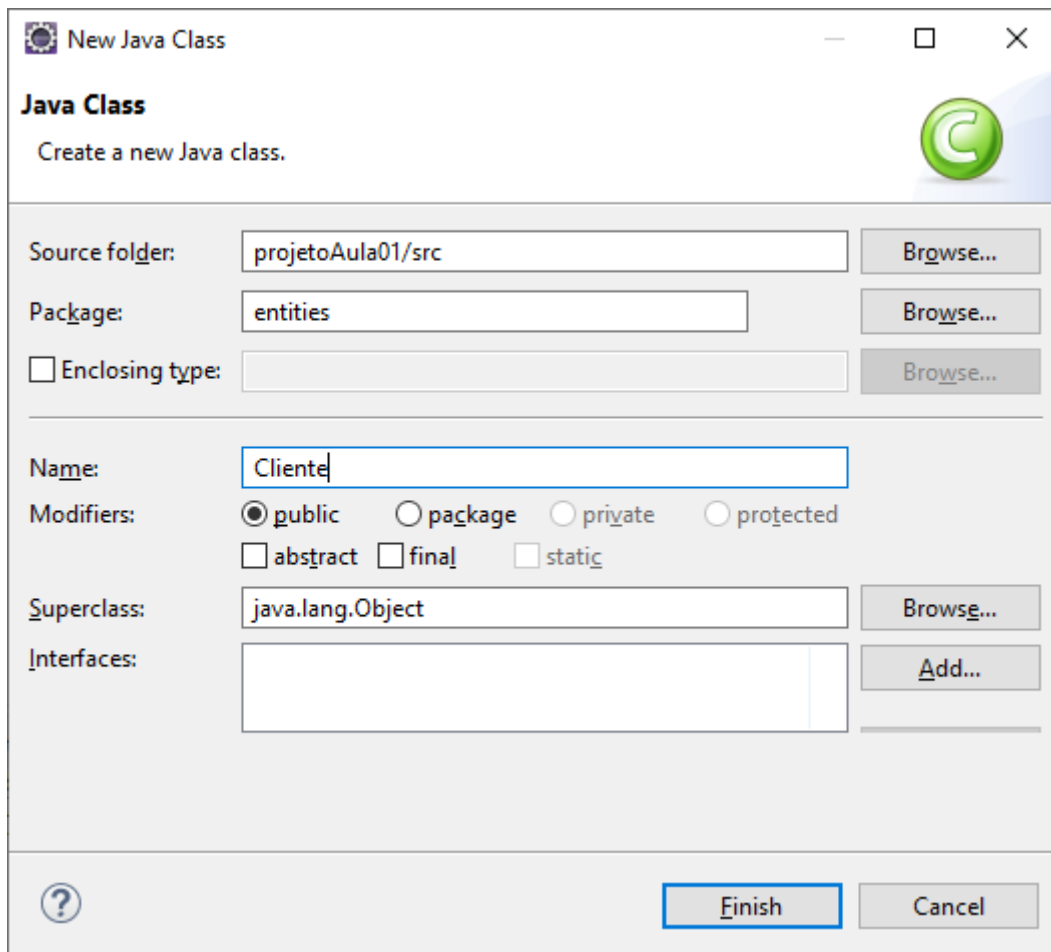
/entities



*Modelagem de dados
(JAVABEAN)*

Vamos criar um pacote no projeto chamado **/entities** para colocarmos todas as classes de modelo de dados (JAVABEANS)





Primeiro, iremos criar os atributos da classe:

Atributos

São campos declarados no corpo de uma classe, definem os dados que uma classe possui. De acordo com o padrão JAVABEAN os atributos devem ser criados com visibilidade **private**.

private

Modificador de visibilidade que, diferente do public, só permite acesso ao atributo ou método dentro da própria classe em que ele está.

```
package entities;
```

```
//definição da classe
```

```
public class Cliente {
```

```
    //atributos da classe (campos)
```

```
    private Integer idCliente;
```

```
    private String nome;
```

```
    private String email;
```

```
    private String telefone;
```

```
}
```

Modificadores de visibilidade vistos até o momento:

public

Permite acesso total a um elemento, geralmente é usado em classes e métodos, não em atributos.

private

Permite acesso somente dentro da própria classe, é o tipo de visibilidade mais restritivo do Java, é comumente utilizado para declaração de atributos em classes que seguem o padrão JAVABEAN.

Encapsulamento

Nome dado em POO (Programação Orientada a Objetos) para o processo de "proteger" os atributos de uma classe do acesso externo. O Encapsulamento é feito quando criamos métodos de entrada e de saída de dados para os atributos de uma classe. Estes métodos são chamados de **set** e **get**.

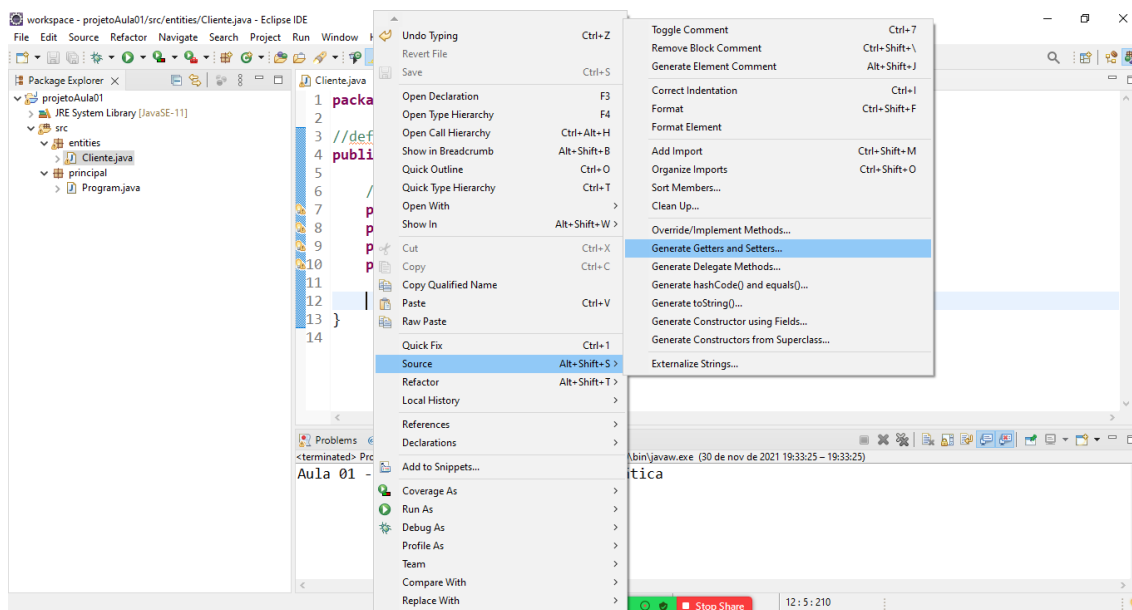
set

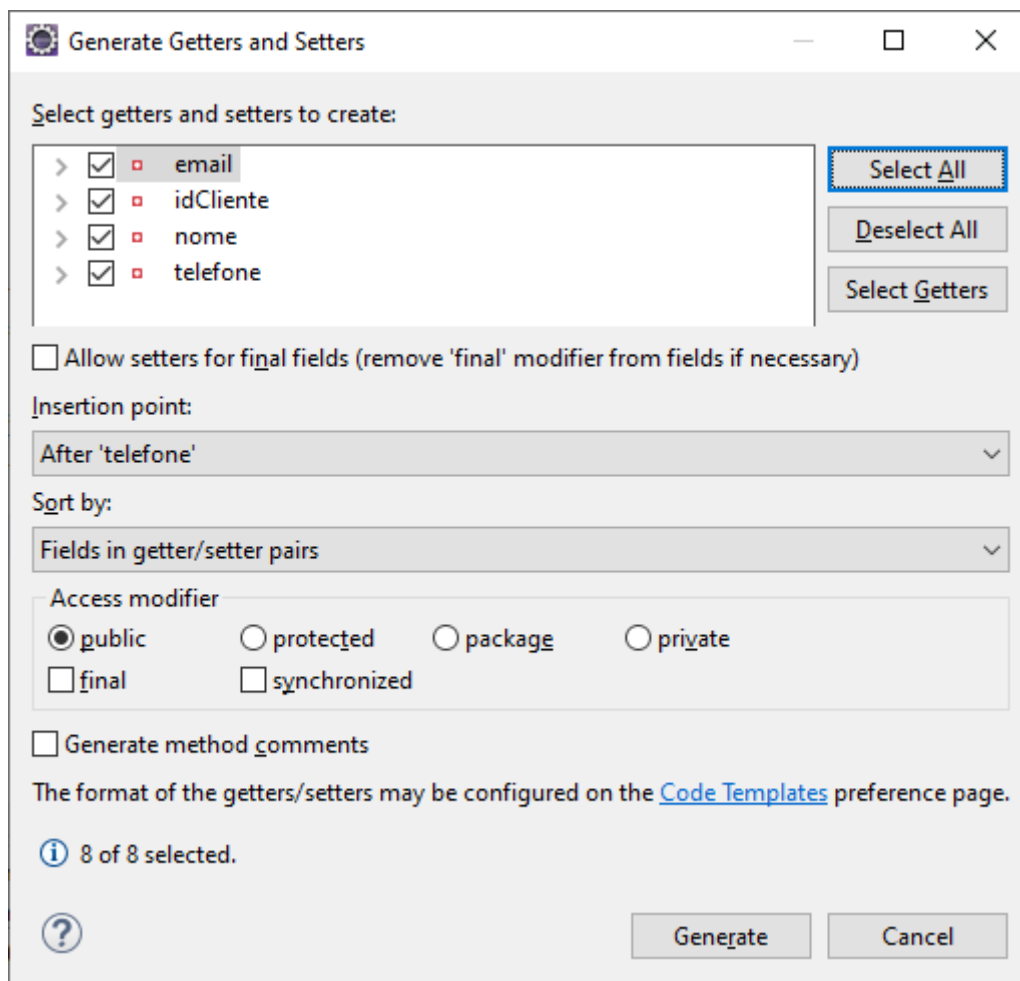
Nome dos métodos que fazem entrada de dados em uma classe JAVABEAN

get

Nome dos métodos que fazem retorno de dados em uma classe JAVABEAN

Para cada atributo privado da classe teremos um método set e um método get para encapsular o atributo. Podemos gerá-los no eclipse da seguinte forma:





`package` entities;

`//definição da classe`

`public class` Cliente {

`// atributos da classe (campos)`

`private` Integer idCliente;

`private` String nome;

`private` String email;

`private` String telefone;

`public` Integer getIdCliente() {

`return` idCliente;

}

`public void` setIdCliente(Integer idCliente) {

`this.idCliente` = idCliente;

}

`public` String getNome() {

`return` nome;

}

```
public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getTelefone() {
    return telefone;
}

public void setTelefone(String telefone) {
    this.telefone = telefone;
}
}
```

Tecla de atalho:

CTRL + SHIFT + O: Faz a importação dos pacotes em uma classe.

Objeto (**Variável de instância**)

Consiste em uma variável criada a partir da instância de uma classe. Ou seja, é uma variável que contém o espaço de memória / referência de uma classe.

```
package principal; //definição do pacote

import entities.Cliente;

//definição da classe
public class Program {

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

        // imprimindo mensagem no console do Eclipse
        System.out.println("Aula 01 - JAVA WebDeveloper
                           COTI Informática");

        //criando uma variável de instância para a classe Cliente
        //objeto = variável de instância
        Cliente cliente = new Cliente();

    }
}
```

Sintaxe para criação de objetos:

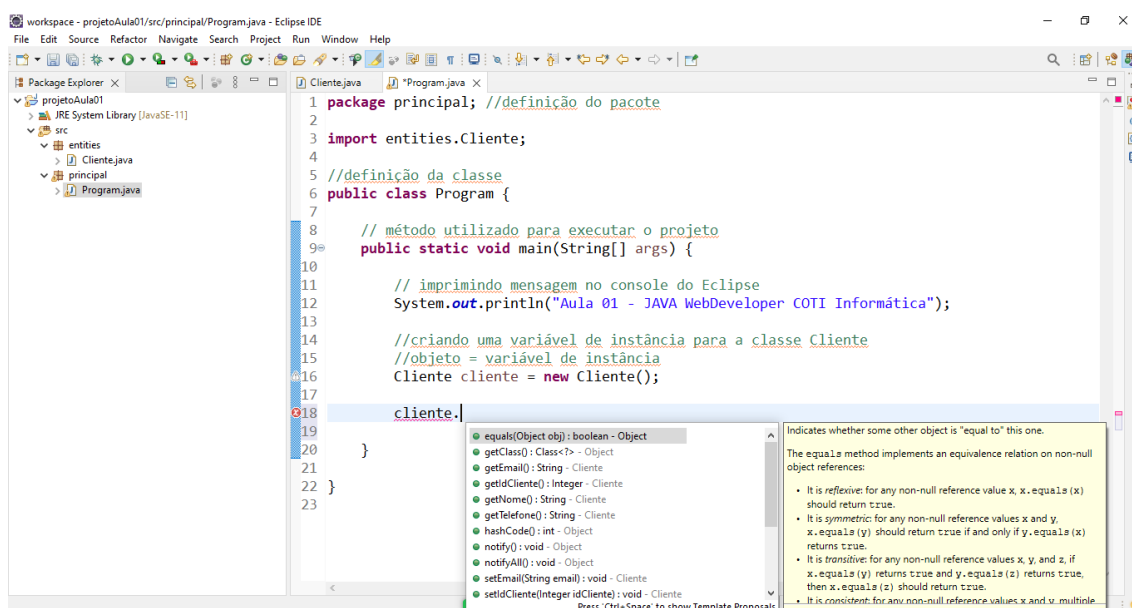
Exemplo:

Cliente **cliente** = **new** **Cliente**();

[Classe] [Nome da variável] [Construtor da Classe]

**** Construtor da classe:** Método da classe responsável por inicializar o objeto, ou seja, abrir espaço de memória para a variável.

Através do objeto podemos acessar o conteúdo da classe, desde que o método ou atributo não seja **privado**.



Se quisermos passar valor para os atributos da classe iremos utilizar os métodos de entrada (**set**)

Exemplo:

```
package principal; //definição do pacote
```

```
import entities.Cliente;
```

```
//definição da classe
```

```
public class Program {
```

```
    // método utilizado para executar o projeto
```

```
    public static void main(String[] args) {
```

```
        // imprimindo mensagem no console do Eclipse
```

```
        System.out.println("Aula 01 - JAVA  
                             WebDeveloper COTI Informática");
```

```
        //criando uma variável de instância para a classe Cliente
```

```
        //objeto = variável de instância
```

```
Cliente cliente = new Cliente();

cliente.setIdCliente(1);
cliente.setNome("Sergio Mendes");
cliente.setEmail("sergio.coti@gmail.com");
cliente.setTelefone("(21) 96957 5900");
    }
}
```

Se quisermos ler os dados do cliente podemos usar os métodos **get**.
Exemplo:

```
package principal; //definição do pacote

import entities.Cliente;

//definição da classe
public class Program {

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

        // imprimindo mensagem no console do Eclipse
        System.out.println("Aula 01 - JAVA WebDeveloper
                           COTI Informática");

        //criando uma variável de instância para a classe Cliente
        //objeto = variável de instância
        Cliente cliente = new Cliente();

        cliente.setIdCliente(1);
        cliente.setNome("Sergio Mendes");
        cliente.setEmail("sergio.coti@gmail.com");
        cliente.setTelefone("(21) 96957 5900");

        System.out.println("\nDADOS DO CLIENTE:");

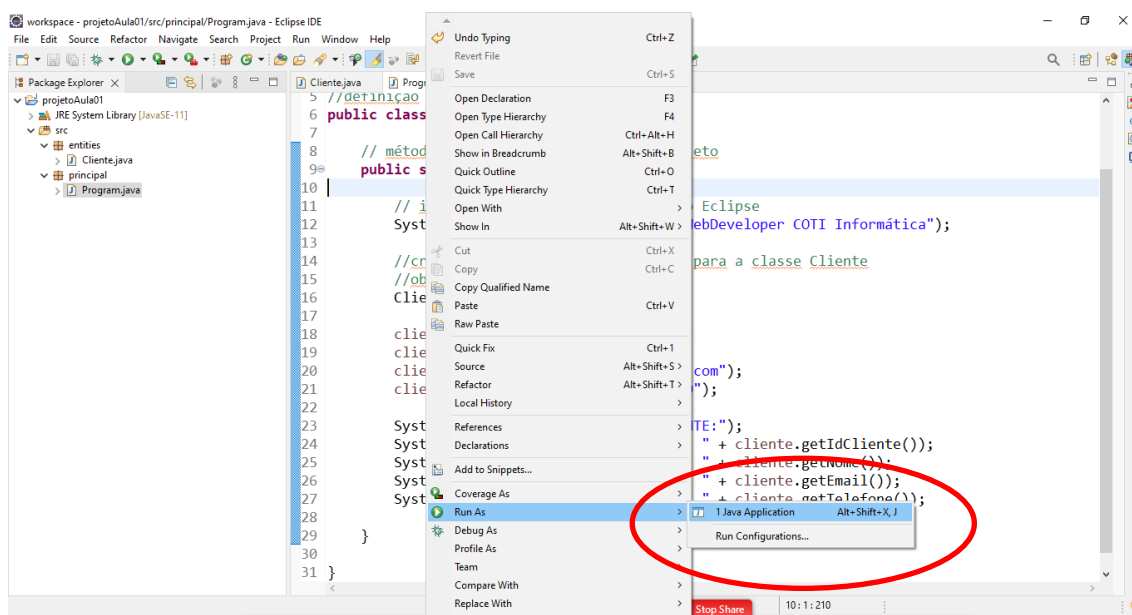
        System.out.println("\tID DO CLIENTE: "
                           + cliente.getIdCliente());

        System.out.println("\tNOME.....: "
                           + cliente.getNome());

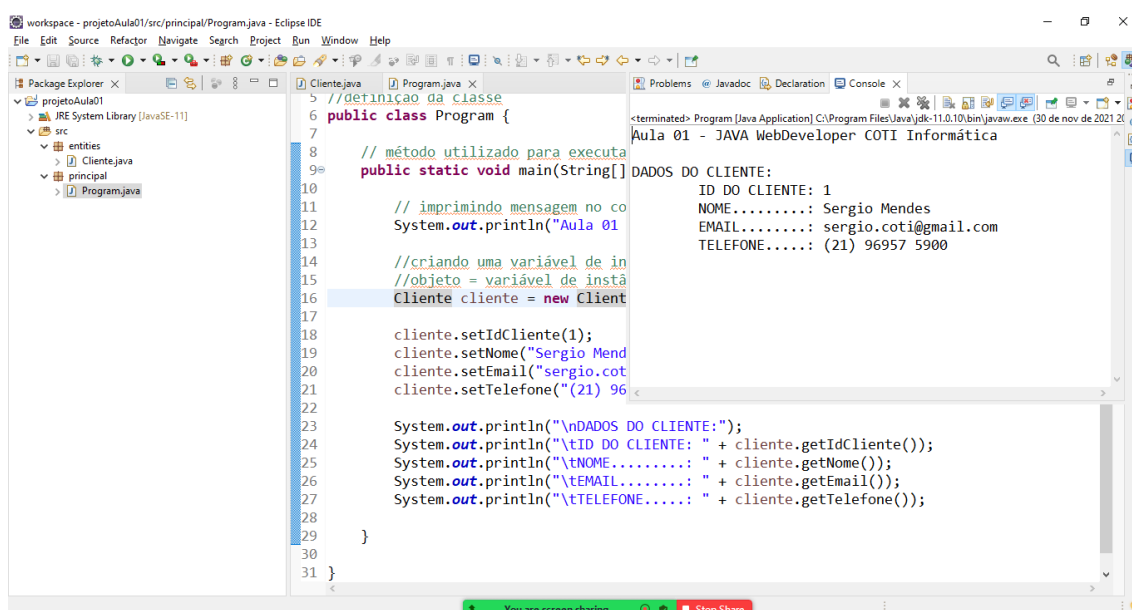
        System.out.println("\tEMAIL.....: "
                           + cliente.getEmail());

        System.out.println("\tTELEFONE.....: "
                           + cliente.getTelefone());
    }
}
```

Executando:



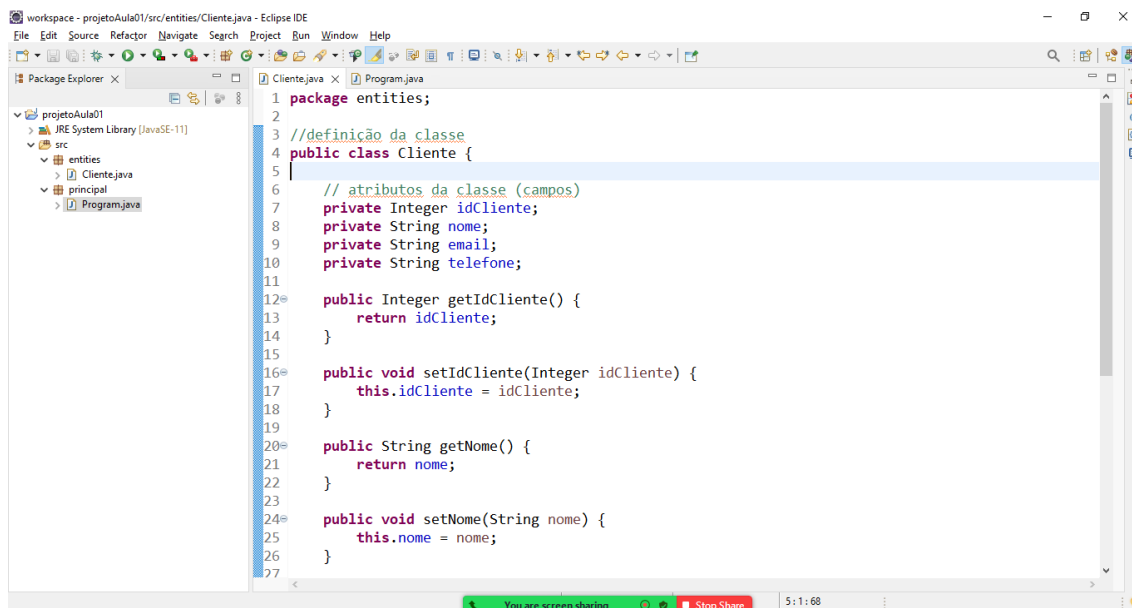
Saída do programa:



JAVABEAN

Nome de uma especificação Java para criação de classes que tem como objetivo **modelar entidades** de um projeto. São características de um JAVABEAN:

- Atributos privados
- Métodos de encapsulamento (set e get)
- Sobrecarga de métodos construtores
- Sobrescrita dos métodos da classe Object (toString, equals e hashCode)



```

1 package entities;
2
3 //definição da classe
4 public class Cliente {
5
6     // atributos da classe (campos)
7     private Integer idCliente;
8     private String nome;
9     private String email;
10    private String telefone;
11
12    public Integer getIdCliente() {
13        return idCliente;
14    }
15
16    public void setIdCliente(Integer idCliente) {
17        this.idCliente = idCliente;
18    }
19
20    public String getNome() {
21        return nome;
22    }
23
24    public void setNome(String nome) {
25        this.nome = nome;
26    }
27

```

Próxima etapa da tarefa:

Desenvolver um programa Java que seja capaz de gravar os dados de um cliente em um arquivo do tipo TXT.

Para isso, iremos criar uma nova classe no projeto somente para fazer esta tarefa de gravação de dados em arquivo.

Dessa forma, cada classe do nosso projeto terá apenas 1 única responsabilidade.

S.O.L.I.D

Acronônimo para 5 boas práticas de programação orientada a objetos. São elas:

- **SRP** Princípio de responsabilidade única;
- **OCP** Princípio de aberto e fechado
- **LSP** Princípio de substituição de Liskov
- **ISP** Princípio de segregação de interfaces
- **DIP** Princípio de inversão de dependência



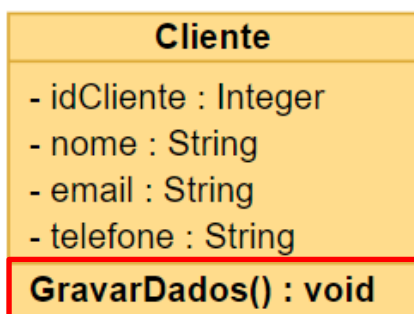


PRINCÍPIO DA RESPONSABILIDADE ÚNICA

Define que uma classe deve ter somente uma única responsabilidade, de forma que as classes do projeto sejam coesas em resolver apenas os problemas para os quais elas foram criadas.

Solução ruim para o problema de gravação de dados do cliente em arquivo:

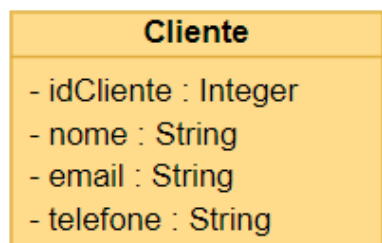
/entities



*Modelagem de dados
(JAVABEAN)*

Solução melhor para o problema de gravação de dados do cliente em arquivo:

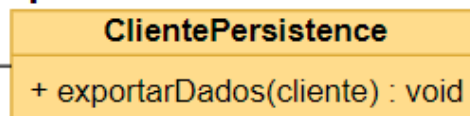
/entities



*Modelagem de dados
(JAVABEAN)*

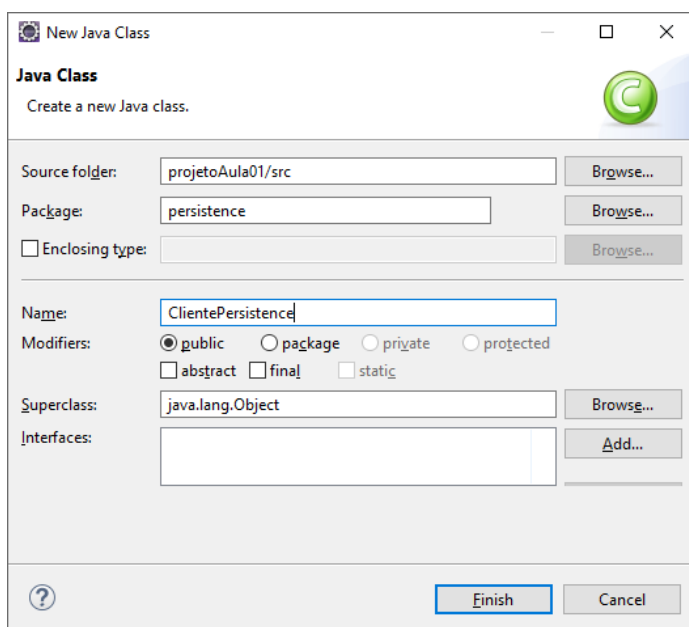
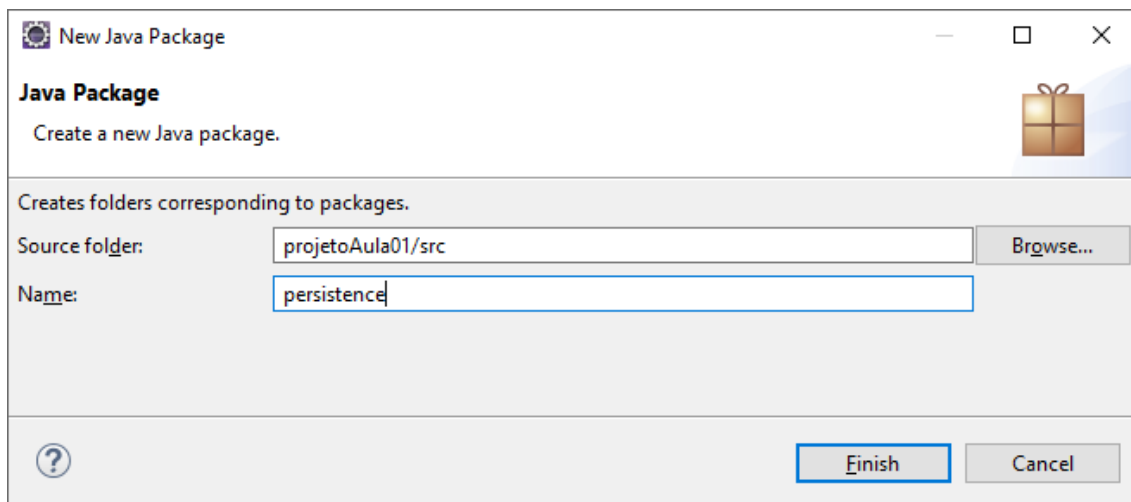
Utiliza

/persistence

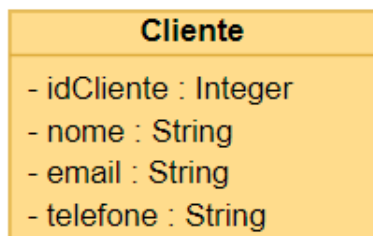


*Classe para persistência de dados
(gravação, consulta, exclusão, edição
etc)*

Criando um pacote para classes de persistência de dados:



/entities



Modelagem de dados
(JAVABEAN)

Utiliza

/persistence

ClientePersistence

```

classDiagram
    class ClientePersistence {
        + exportarDados(cliente) : void
    }
  
```

Classe para persistência de dados
(gravação, consulta, exclusão, edição
etc)

void

Define métodos em JAVA que não retornam valor, ou seja, métodos que não utilizam a palavra reservada **return** seguida de valor.

```
package persistence;

import entities.Cliente;

public class ClientePersistence {

    // método para receber os dados de um
    // cliente e grava-los em arquivo
    public void exportarDados(Cliente cliente) {

    }

}
```

Implementar a gravação dos dados do cliente em arquivo:

```
package persistence;

import java.io.File;
import java.io.PrintWriter;

import entities.Cliente;

public class ClientePersistence {

    // método para receber os dados de
    //um cliente e grava-los em arquivo
    public void exportarDados(Cliente cliente) {

        try { //tentativa

            //criando o arquivo
            PrintWriter print = new PrintWriter
                (new File("c:\\temp\\cliente.txt"));

            //escrevendo no arquivo
            print.write("\nDADOS DO CLIENTE:");
            print.write("\n\tId.....: "
                + cliente.getIdCliente());
            print.write("\n\tNome.....: "
                + cliente.getNome());
            print.write("\n\tEmail.....: "
                + cliente.getEmail());
            print.write("\n\tTelefone...: "
                + cliente.getTelefone());

        }

    }

}
```

```
        print.flush(); //salvar o arquivo
        print.close(); //fechar o arquivo
    }
    catch(Exception e) { //captura do erro
        //imprimir o log de erro
        e.printStackTrace();
    }
}
}
```

Voltando na classe **Program.java**

```
ClientePersistence persistence = new ClientePersistence();
    [Classe]      [variável de instância]      [Inicialização]

package principal; //definição do pacote

import entities.Cliente;
import persistence.ClientePersistence;

//definição da classe
public class Program {

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

        // imprimindo mensagem no console do Eclipse
        System.out.println("Aula 01 - JAVA WebDeveloper
                           COTI Informática");

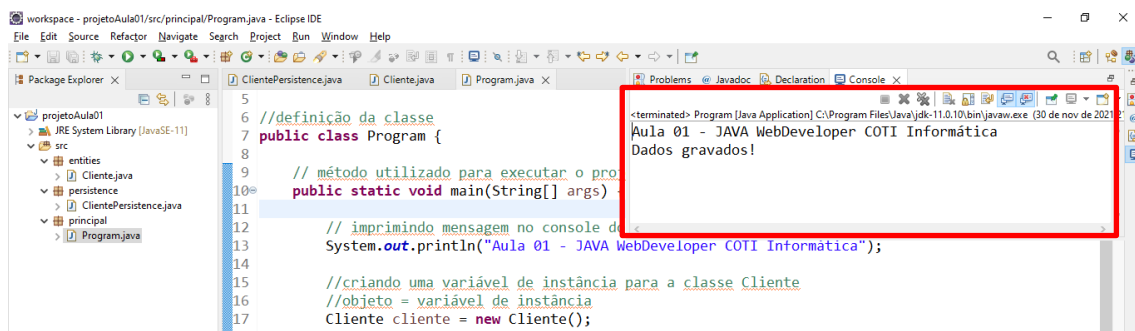
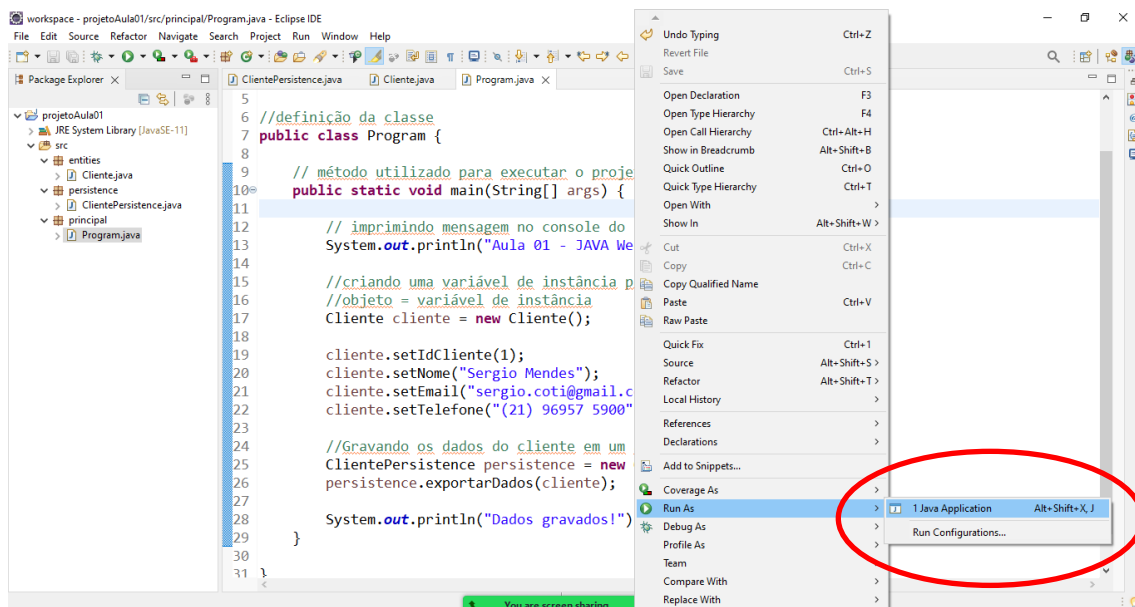
        //criando uma variável de instância para a classe Cliente
        //objeto = variável de instância
        Cliente cliente = new Cliente();

        cliente.setIdCliente(1);
        cliente.setNome("Sergio Mendes");
        cliente.setEmail("sergio.coti@gmail.com");
        cliente.setTelefone("(21) 96957 5900");

        //Gravando os dados do cliente em um arquivo
        ClientePersistence persistence = new ClientePersistence();
        persistence.exportarDados(cliente);

        System.out.println("Dados gravados!");
    }
}
```

Executando: CTRL + F11



Arquivo gerado:

