



C206/C06 – Programação Orientada a Objetos com Java



Pacotes

Prof. Christopher Lima
christopher@inatel.br



Objetivos

- ☕ Usar Pacotes para **organizar** as Classes Java
- ☕ Entender o conceito de **fully-qualified-name** da Classe Java
- ☕ Verificar o que são **imports**



Organização de Classes



- ☕ Imagine que **você tenha criado uma Classe “Matematica”**, com alguns métodos que realizam operações matemáticas.
- ☕ Agora imagine que **sua colega também criou uma classe “Matematica”, com algumas melhorias**, e deseja lhe passar essa classe.
- ☕ **Como o seu projeto pode lidar** com classes que tenham o mesmo nome?

Organização de Classes



```
//Sua classe
public class Matematica {

    int soma(int x, int y) {
        return x + y;
    }

    int subtrair(int x, int y) {
        return x - y;
    }

    double dividir(int x, int y) {
        return x/y;
    }
}
```

```
//Classe do seu amigo
public class Matematica {

    int soma(int x, int y) {
        return x + y;
    }
    int subtrair(int x, int y) {
        return x - y;
    }
    double dividir(double x, double y) {
        if(y == 0) {
            throw new RuntimeException();
            //Nao se preocupem com isso
        }
        return x/y;
    }
}
```

Organização de Classes



- ☕ Uma primeira ideia seria simplesmente “copiar” o código por cima. Mas essa solução **não é escalável**.
- ☕ Adicionalmente, a sua classe pode ter outros métodos úteis. Assim você deseja que as **duas classes possam coexistir** no mesmo projeto.
- ☕ Pensando em **sistema operacional**, é possível ter dois arquivos com o mesmo nome no **mesmo diretório**?
 - ☕ Não! E como organizamos nossos arquivos no computador?



Organização de Classes



☕ Usamos diretórios

☕ No Java, os **pacotes** se comportam exatamente como diretórios (pastas).

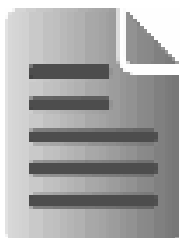


Organização de Classes



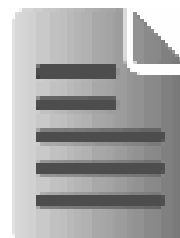
- ☕ Assim como os diretórios contém os nossos arquivos, os pacotes contém as classes.
- ☕ Na verdade, classes Java são arquivos com extensão “.java”. Então podemos ver pacotes como diretórios!

Matematica.java



Sua classe!

Matematica.java



Da sua amiga!



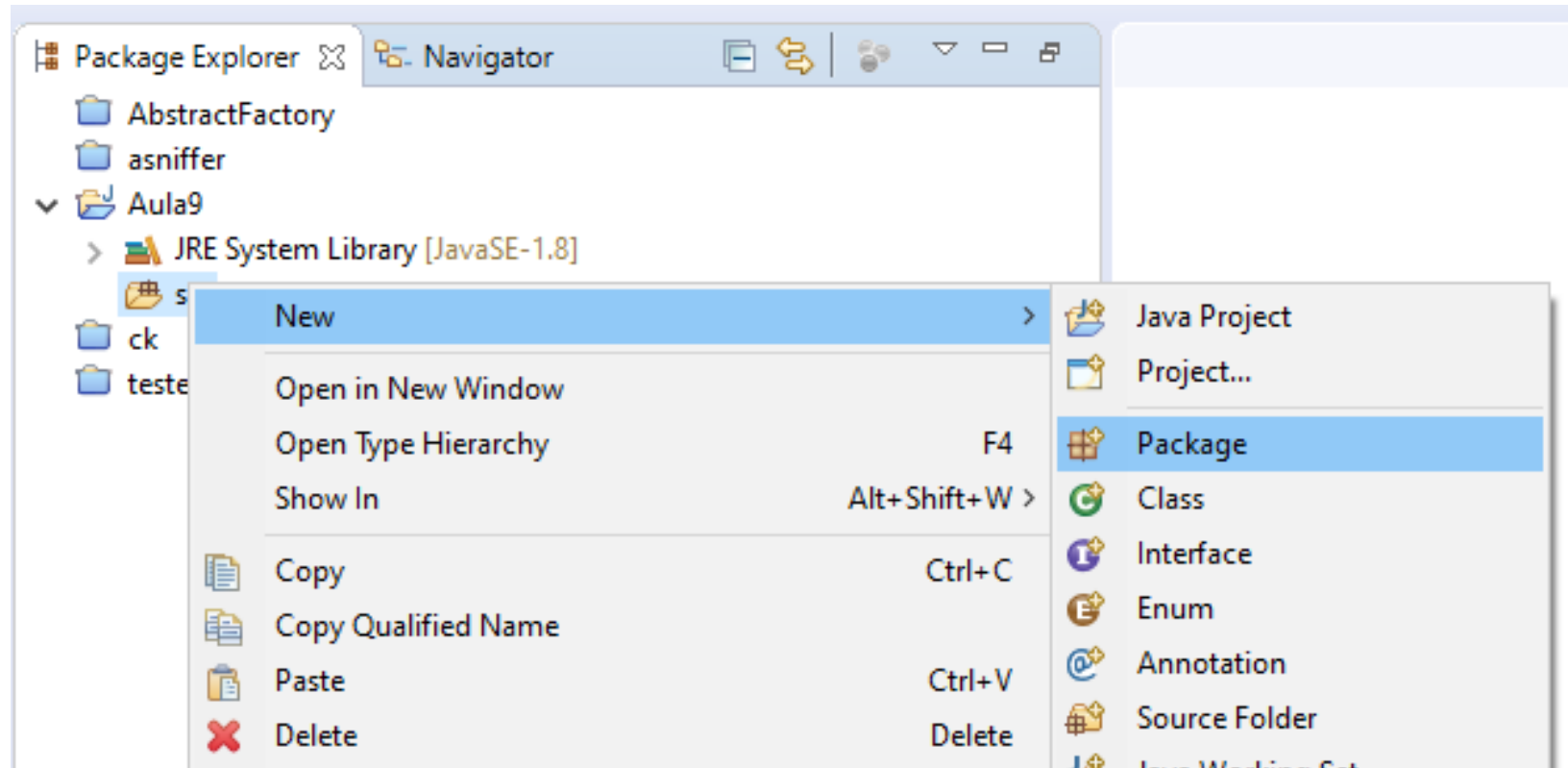
Criando pacotes



- ☕ Vamos criar nossos pacotes!
- ☕ O próximo exemplo considera que ainda não criamos a classe Matemática!
- ☕ Criaremos um pacote chamado “matemática” para colocar a nossa classe “Matematica”.



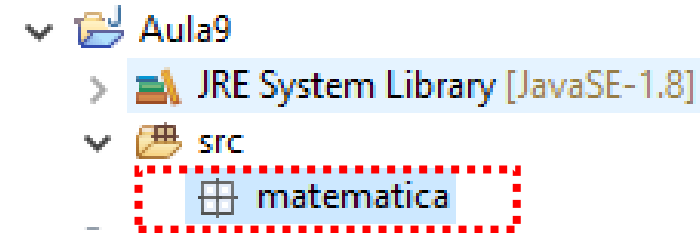
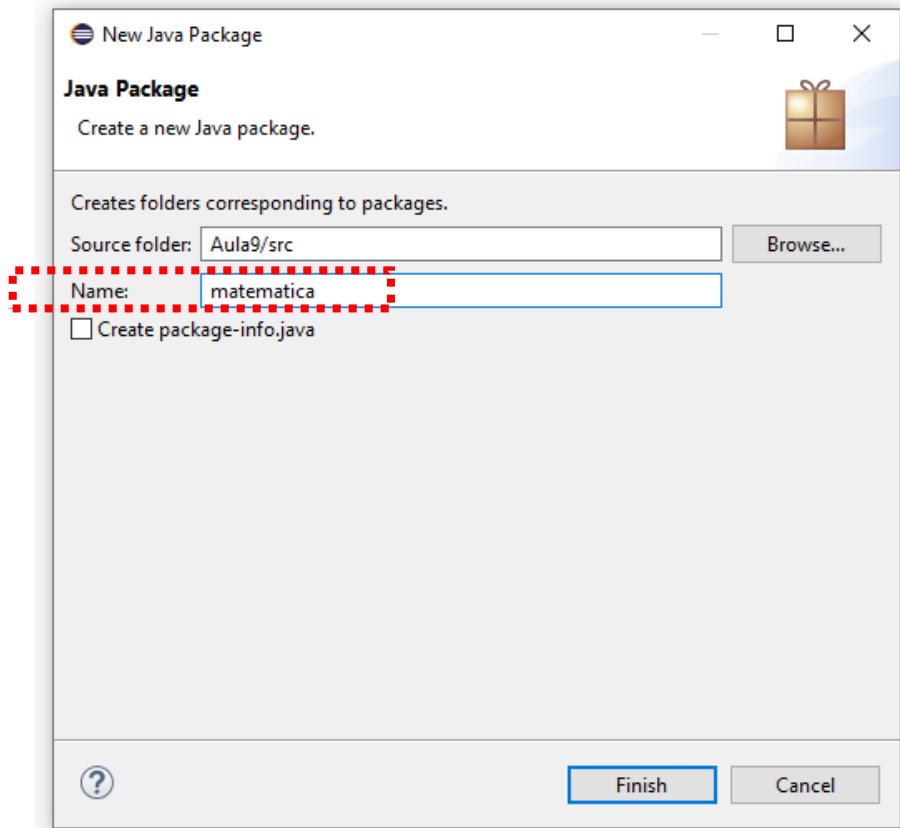
Criando pacotes



☕ Clique com o botão direito em “src”, depois vá em New -> Package



Criando pacotes




☕ Observe no “Package-Explorer” que agora temos dentro de “src” um novo diretório chamado “matemática”. Está em branco pois não tem nenhuma classe dentro

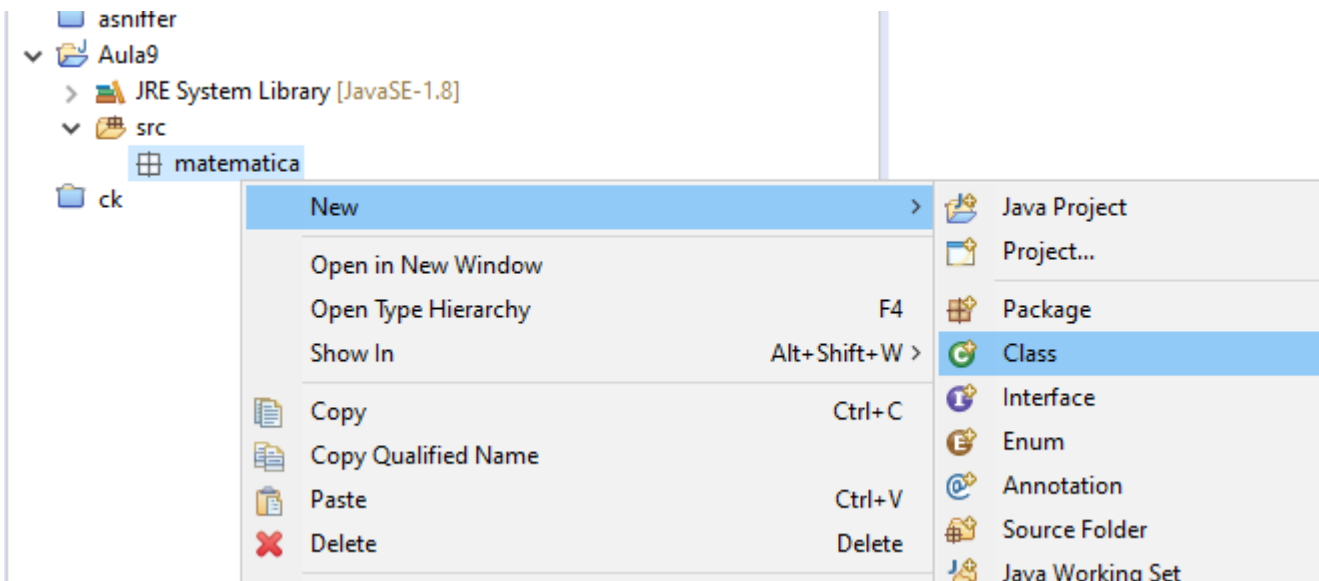
☕ Dê um nome para o pacote. No exemplo foi colocado “matemática”



Criando pacotes



 Vamos adicionar uma nova classe chamada “Matematica” nesse pacote!



☕ Observe que o campo “Package” foi preenchido automaticamente!

☕ O nome da classe precisamos preencher manualmente

New Java Class

Java Class

Create a new Java class.

Source folder: Aula9/src Browse...

Package: matematica Browse...

☐ Enclosing type: Browse...

Name: Matematic

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

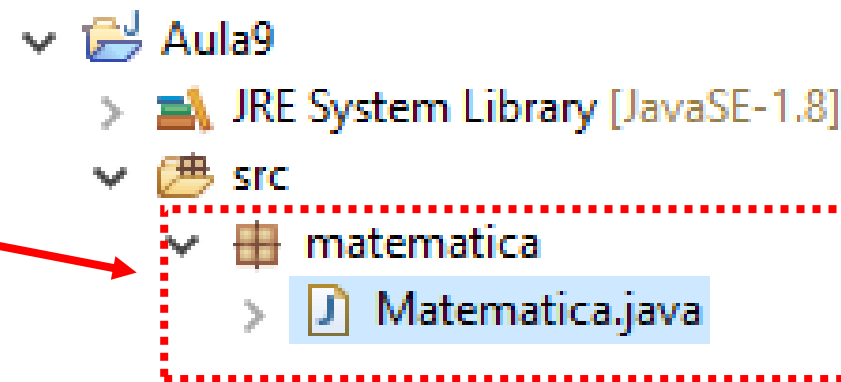
Finish Cancel





Criando pacotes

☕ Observe como ficou a estrutura do projeto após a criação da classe





Criando pacotes

```
package matematica;  
  
public class Matematica {  
  
}
```

- ☕ Observe a palavra-chave “package”. Ela indica em qual pacote se encontra a classe.
- ☕ Nesse exemplo a classe “Matematica” se encontra no pacote “matemática”



Criando pacotes – Convenção de Nome



☕ O padrão da Sun (empresa que criou o Java) para dar **nome aos pacotes** é **relativo ao nome da empresa que desenvolveu a classe**. Exemplos:

☕ `br.com.nomedaempresa.nomedoprojeto.subpacote`

☕ `br.com.nomedaempresa.nomedoprojeto.subpacote2`

☕ `br.com.nomedaempresa.nomedoprojeto.subpacote2.subpacote3`

☕ Cada “.” (ponto) indica um novo **subpacote**, isto é, pacote dentro de pacote. Pode ser visto como um **diretório dentro de outro**.

☕ Vamos alterar o pacote “matemática” para “**br.inatel.cdg.matematica**”. Veremos como o Eclipse pode nos auxiliar



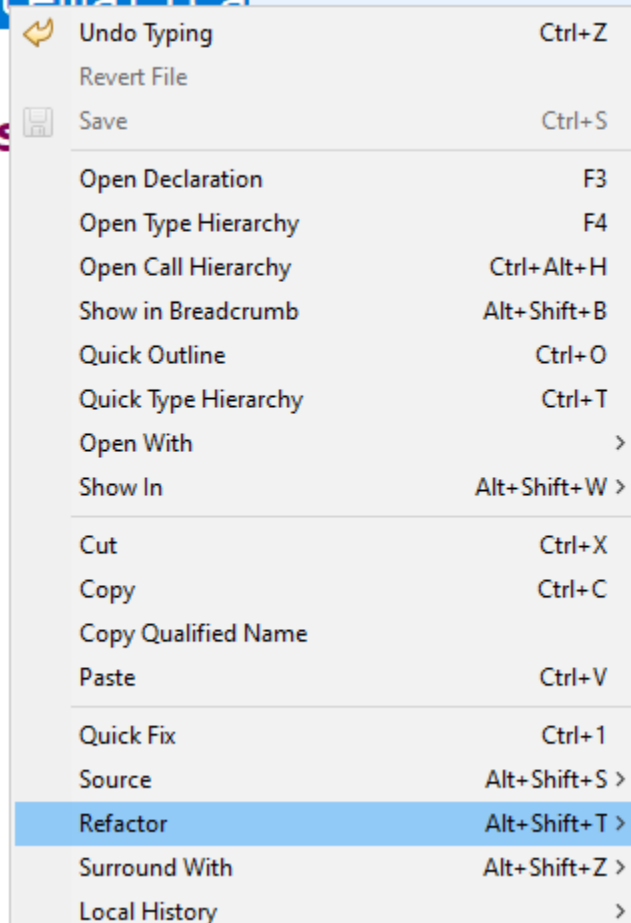
Criando pacotes – Convenção de Nome



```
package matematica;
```

```
public class
```

```
}
```

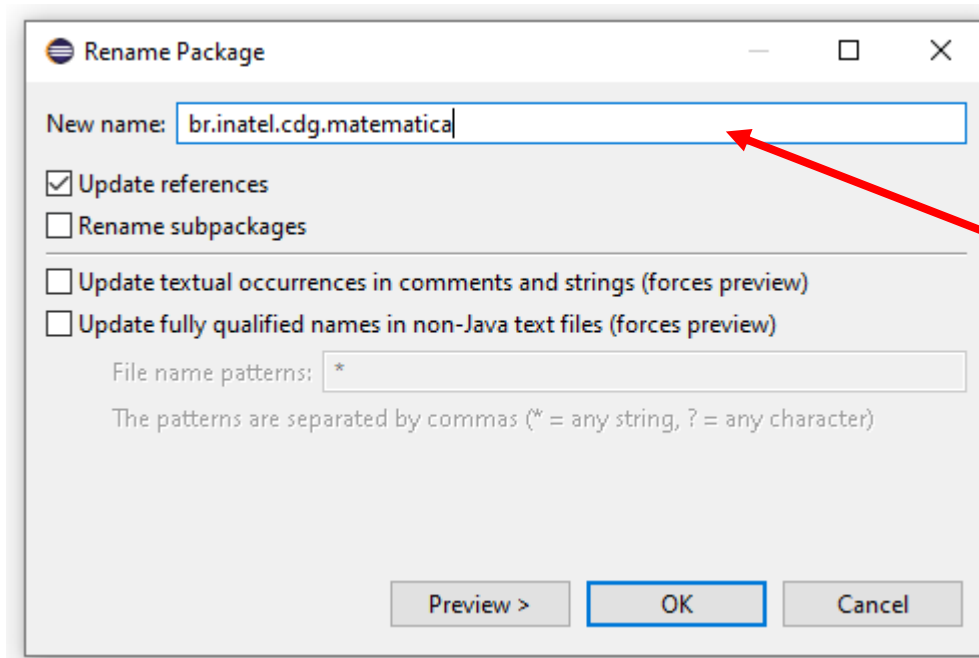


Clique com o Botão Direito em
"matematica"

Rename... Alt+Shift+R



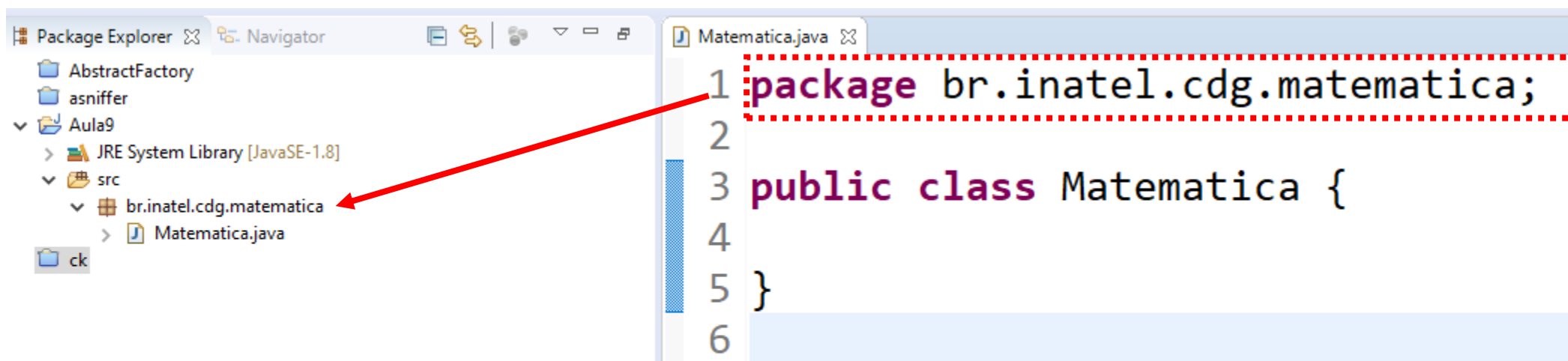
Criando pacotes – Convenção de Nome



Coloque o novo nome do
pacote



Criando pacotes – Convenção de Nome



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays a project structure with a package named `br.inatel.cdg.matematica` under the `src` folder. A red arrow points from this package in the Package Explorer to the corresponding package declaration in the code editor. The code editor shows the file `Matematica.java` with the following code:

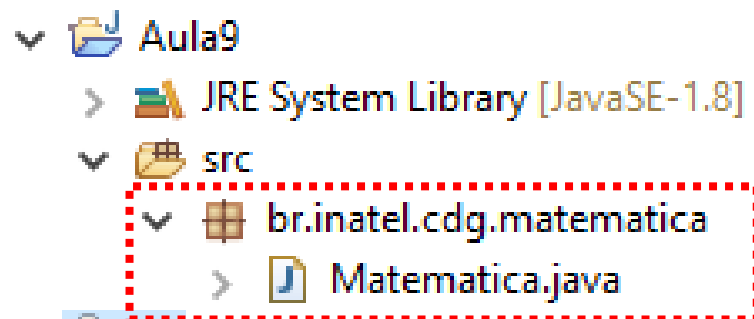
```
1 package br.inatel.cdg.matematica;  
2  
3 public class Matematica {  
4  
5 }  
6
```

The package declaration on line 1 is highlighted with a red dashed border.

☕ O próprio Eclipse já cuidou de renomear o pacote, além de ter criado a nova estrutura “`br.inatel.cdg.matematica`”.



Criando pacotes – Convenção de Nome



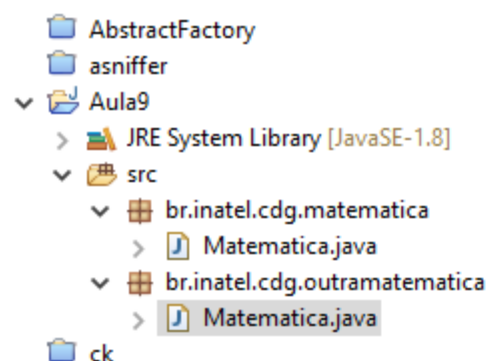
☕ Não confunda e pense que foi criado um único pacote chamado “br.inatel.cdg.matemática”. Na verdade temos 4 pacotes.

☕ br, inatel, cdg, matematica.



Criando Pacotes

☕ Vamos agora criar outro pacote para conter a classe “Matematica” criada pelo seu colega.



```
1 package br.inatel.cdg.outramatematica;  
2  
3 public class Matematica {  
4  
5 }  
6
```

☕ Agora podemos ter duas classes chamadas “Matematica”, sem problema! Ambas estão em pacotes diferentes



Fully-Qualified-Name (Nome Completo)



- ☕ Agora que temos as duas classe “Matematica”, como podemos utilizá-las?
- ☕ Precisamos informar o caminho completo da classe que desejamos.
- ☕ Vamos criar uma classe “Main” dentro do pacote br.inatel.cdg, para começarmos a testar



New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

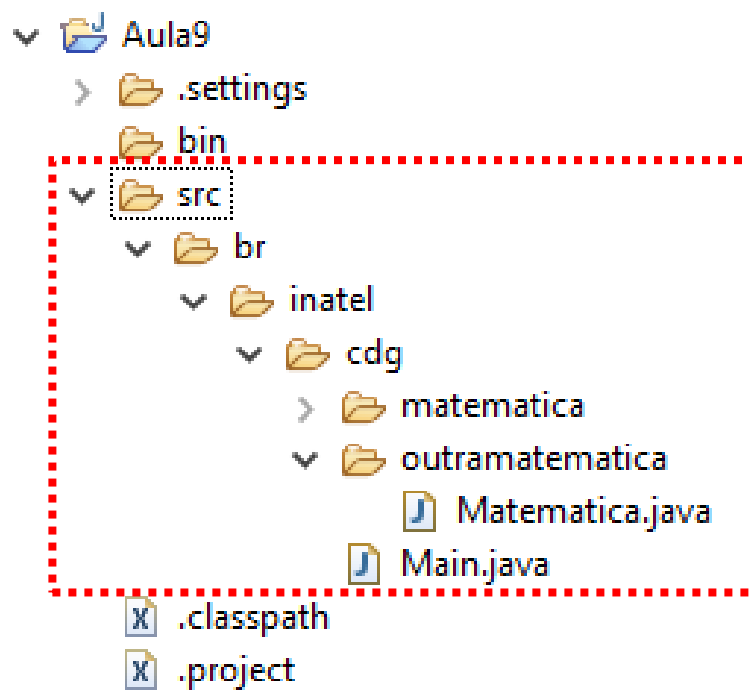
☐ Generate comments



Fully-Qualified-Name (Nome Completo)



☕ Para reforçar a ideia de diretório, observe como está o projeto na aba “Navigator”. Essa aba mostra a hierarquia de arquivos, exatamente como no Sistema Operacional (pelo Windows Explorer ou Finder).





Fully-Qualified-Name (Nome Completo)



☕ Dentro do método “main” vamos criar uma instância de Matematica, mas qual delas?

```
public static void main(String[] args) {  
    //O compilador não sabe qual Matematica queremos usar  
    Matematica m = new Matematica();  
}
```

☕ Uma opção é passar o nome completo, ou *Fully-Qualified-Name*.



Fully-Qualified-Name (Nome Completo)



☕ O Fully-Qualified-Name contém o nome de todos os pacotes onde a classe está presente. Assim, o compilador sabe exatamente qual classe queremos.

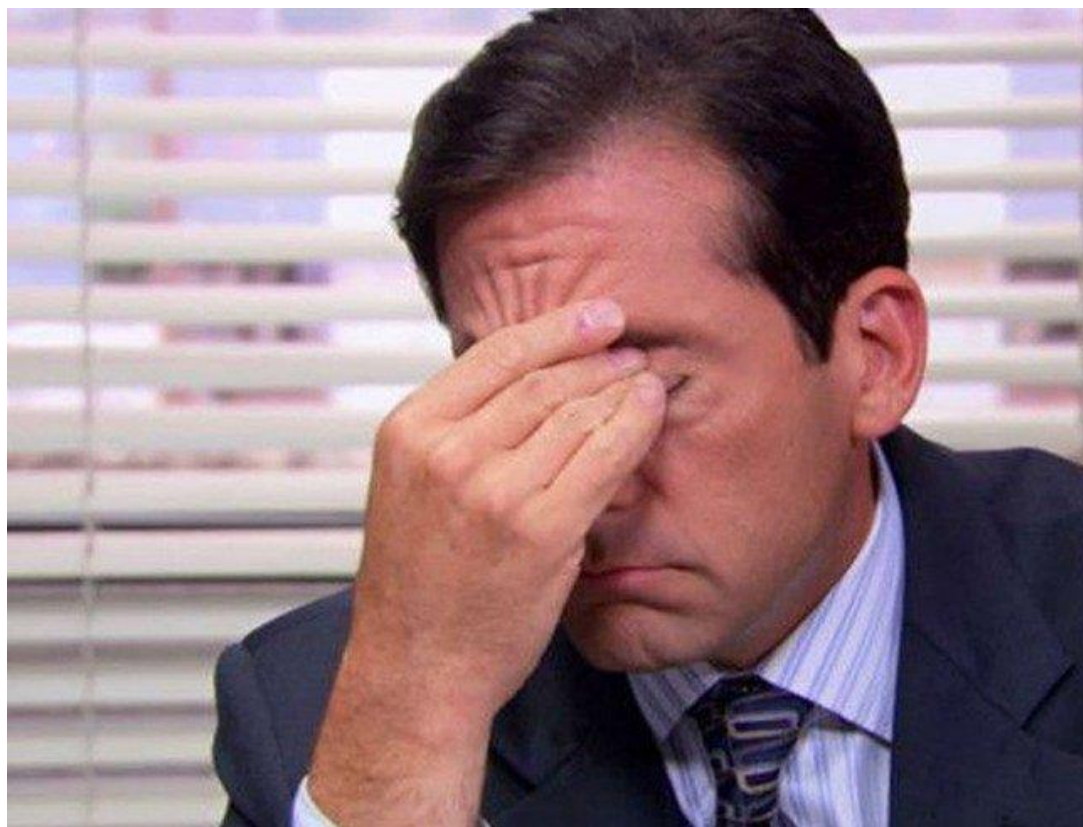
```
public static void main(String[] args) {  
    //Utilizando o fully-qualified-name  
    br.inatel.cdg.matematica.Matematica  
        m = new br.inatel.cdg.matematica.Matematica();  
}
```



Fully-Qualified-Name (Nome Completo)



☕ Parece divertido programar assim!





Utilizando Imports



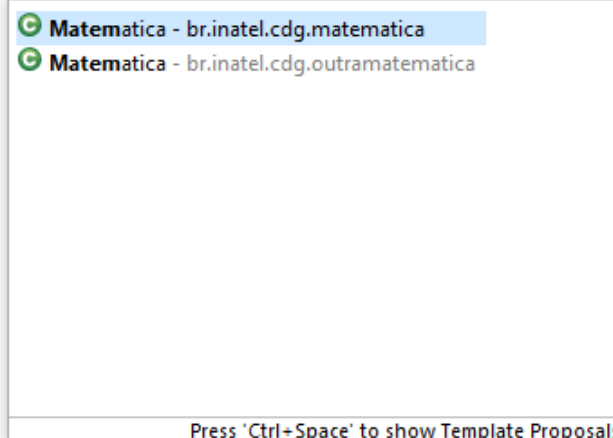
- ☕ Podemos **facilitar a nossa vida**, utilizando a palavra chave *import*
- ☕ Com ela podemos, no início da classe, **descrever quais outras classes de outros pacotes vamos utilizar**.
- ☕ O Eclipse/IntelliJ nos ajuda com isso!



Utilizando Imports

- ☕ Comece a escrever “Matematica” e pressione Ctrl-Espaço
- ☕ O Eclipse irá lhe mostrar algumas opções de potenciais classes.
- ☕ Vamos escolher a primeira opção e pressionar “Enter”

```
public static void main(String[] args) {  
    //Utilizando imports  
    Matem  
}
```





Utilizando Imports



```
package br.inatel.cdg;
```

```
import br.inatel.cdg.matematica.Matematica;
```

```
public class Main {
```

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

```
        //Utilizando imports
```

```
        Matematica m = new Matematica();
```

```
    }
```

```
}
```

☕ O “package” fica logo no início da classe. Ele informa a qual pacote a classe atual (“Main”) pertence.

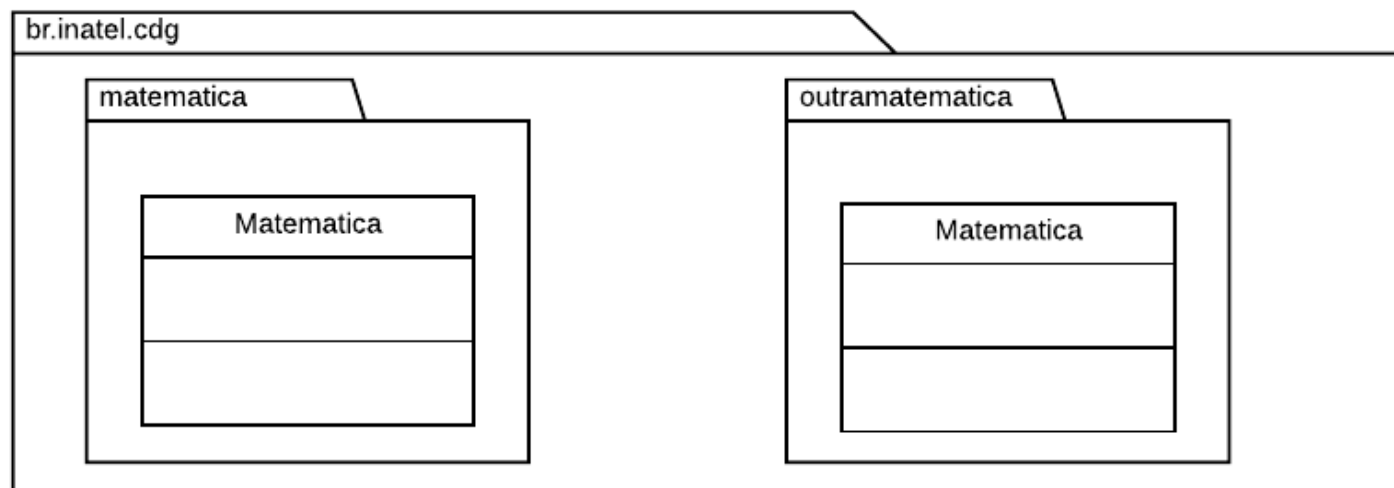
☕ Em seguida temos o “import”. Informando quais outras classes estamos usando. Repare que ele possui o *fully qualified name* da classe “Matematica”.

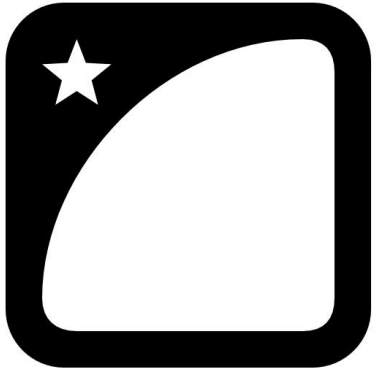
☕ Não há dúvidas de qual “Matematica” estamos usando!



Diagrama UML com Pacotes

- Observe que nesse caso não especificamos os membros e métodos das classes, pois estamos interessados na organização dos pacotes. Estamos uma camada acima.





Material Complementar



 Capítulo 7 da apostila FJ-11 (pg 102)

 Pacotes - Organizando suas Classes e Bibliotecas