

Java: Pacotes (Packages)

Pacotes (Pacakges)

- ❖ O conceito de pacote em Java existe para evitar conflitos de nomes
 - ❖ Imagine que o projeto é composto por diferentes equipes, onde inicialmente foi-se definido as Interfaces que cada grupo iria oferecer ao outro.
 - ❖ Os grupos isoladamente implementam a sua parte no projeto sem muito contato com as demais, de posse das interfaces dos outros grupos, eles simplesmente assumem que um dia elas estarão implementadas.
- ❖ Pense, qual a probabilidade de neste projeto existirem classes com o mesmo nome?

Pacotes (Pacakges)

- ❖ Acreditem, a chance é grande.
- ❖ Nomes como GerenciadorDe*, Calculadora, Relatorio, Resumo, Crendencial, todos são nomes comuns para classes em projetos pequenos.
 - ❖ Certamente que em projetos muito grandes ele pode aparecer duas vezes.
- ❖ Um outro fator que contribui para o choque de nomes é o uso de bibliotecas de terceiros.

Pacotes (Pacakges)

- ❖ Importamos uma classe da biblioteca padrão de Java algumas vezes durante o curso:

```
import java.util.Date;
```

- ❖ Indicamos aqui que a classe Date que queremos está no pacote **java.util**.

Pacotes (Pacakges)

- ❖ Este conceito está longe de ser uma exclusividade do Java.
- ❖ Por exemplo no C# tem se os **namespaces** (Espaço de Nomes).
- ❖ Essas técnicas existem para evitar choque de nomes.

Pacotes (Packages)

- ❖ Em um projeto java que utiliza pacotes (praticamente todo projeto sério o faz) as classes são organizadas em uma hierarquia de diretório.
- ❖ Essa hierarquia é baseada em um domínio, geralmente o domínio de uma empresa, seguida pelo nome do pacote. Por exemplo, um domínio como **meu-app.exemplo.org** se tornaria o pacote seguinte:

```
package org.exemplo.meu_app
```

- ❖ Para indicar que uma classe pertence ao pacote, a linha acima deverá ser a primeira linha d arquivo contendo sua classe.

Pacotes (Packages)

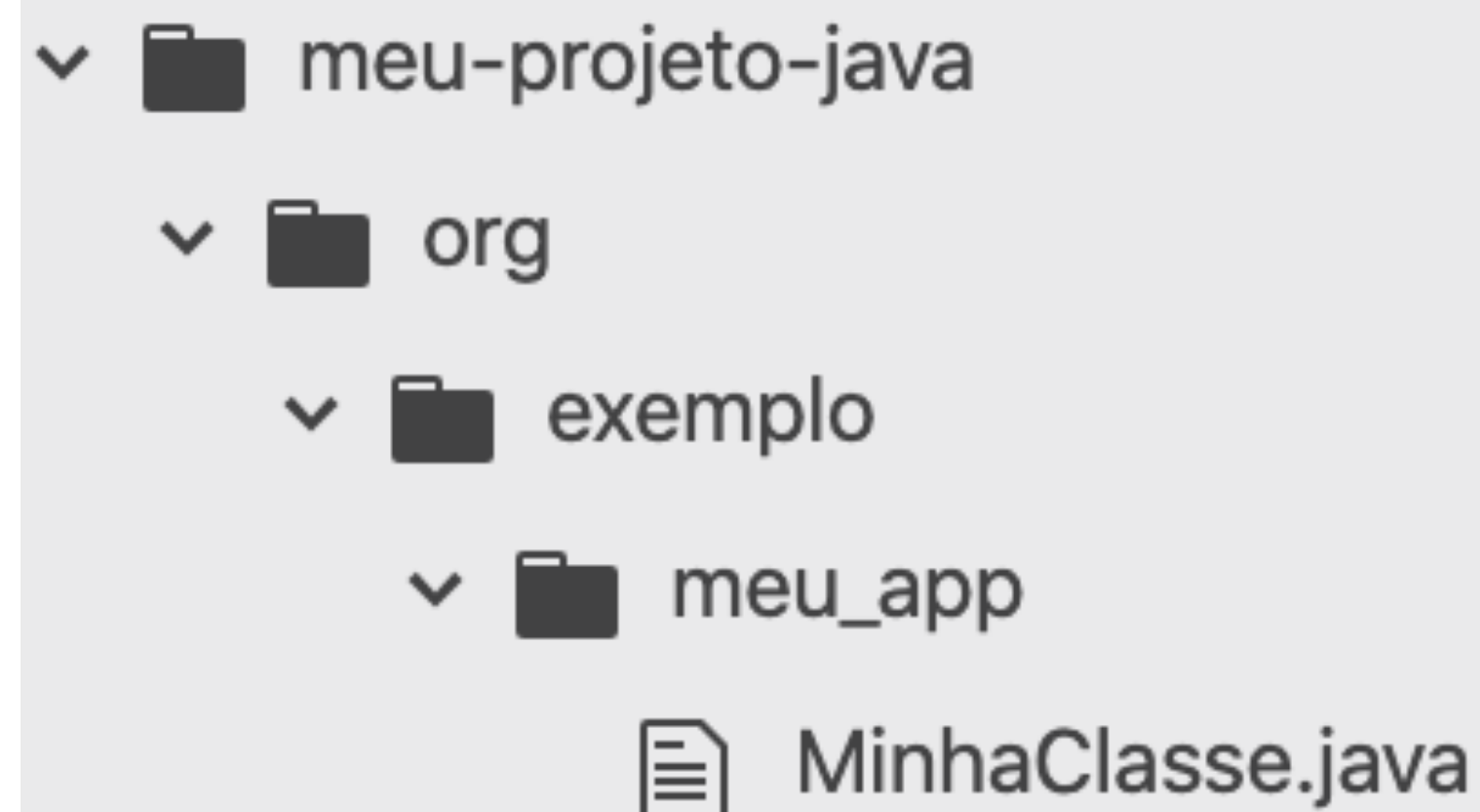
ATENÇÃO

O uso de um domínio não é obrigatório para nomear um pacote, mais altamente recomendado e o padrão da indústria. Um exemplo de não uso é o pacote **java** que contém as classes padrões da linguagem.

Pacotes (Packages)

`package org.exemplo.meu_app`

- ❖ Não só deverá ser a primeira linha do seu programa, como o arquivo fonte **.java** deve estar em uma hierarquia de diretório com o mesmo caminho do nome do pacote, neste caso:



```
graph TD; A[meu-projeto-java] --> B[org]; B --> C[exemplo]; C --> D[meu_app]; D --> E[MinhaClasse.java];
```

Diagrama de estrutura de diretórios:

- meu-projeto-java
 - org
 - exemplo
 - meu_app
 - MinhaClasse.java

Pacotes (Pacakges)

- ❖ Parece chato de gerenciar não?
- ❖ Mas as melhores IDEs irão cuidar de deixar esse trabalho adicional por conta deles.
- ❖ A lógica por trás do uso de domínio de uma empresa é que assim se evitará colisões.
 - ❖ E as empresas podem utilizar o espaço de nomes depois do domínio como bem entenderem.

Pacotes (Pacakges)

- ❖ Vejamos alguns exemplos:
 - ❖ `java.sql.Date`
`java.util.Date`
 - ❖ `com.mysql.jdbc.Driver`
`org.mariadb.jdbc.Driver`
- ❖ Acima, mesmos nomes para tipos diferentes, organizar o código em pacotes evita conflitos.

Pacotes (Pacakges)

- ❖ Quando importamos uma Classe, como por exemplo em:

```
import java.util.Date;
```

- ❖ Podemos utilizá-la referindo-se apenas ao seu nome **Date**.
- ❖ Mas se precisarmos utilizar as duas que possuem o mesmo nome?
 - ❖ Para isso temos duas soluções....

Pacotes (Pacakges)

- ❖ Quando precisamos utilizar duas classes com o mesmo nome, devemos:
 - ❖ Omitir a importação de uma delas (não fazer o import)
 - ❖ Utilizar sempre o caminho completo do tipo que você não importou, exemplo:

```
import java.util.Date;  
  
public MinhaClasse {  
    private Date data1;  
    private java.sql.Date data2;  
}
```


Pacotes (Pacakges)

- ❖ Vamos ver na prática como isto fica...
 - ❖ Código do exemplo disponível no repositório da disciplina no GitHub...

Java: Classe e Método Main

Classe e Método Main

- ❖ Todo programa possui um ponto de partida, aquele código que é invocado quando o sistema operacional passa o controle para a aplicação.
- ❖ Em C, estamos acostumados com a função **int main()**.
- ❖ Até agora utilizamos o BlueJ para criar objetos e executar métodos, mas na vida real existirá uma Classe que funcionará como a cola principal do nosso programa OO.

Classe e Método Main

- ❖ Em qualquer classe Java podemos criar um método main.
- ❖ Sendo assim, quando pedirmos para a JVM rodar aquela classe ele irá passar o controle para esse método.
- ❖ Algumas pessoas recomendam que o método main esteja em uma classe separada geralmente chamada de Main.
 - ❖ Isso pode se transformar em um flamewar...

Classe e Método Main

Bom, eu recomendo que se escreva o método main em uma classe separada.
Explico porque...

Classe e Método Main

- ❖ Ao iniciar um programa muitas vezes você irá precisar checar uma série de condições para que o seu programa possa executar corretamente:
 - ❖ O arquivo de configuração está presente e correto?
 - ❖ Configurar o sistema de Log.
 - ❖ Verificar conectividade com a internet.
- ❖ Porém nada impede de que o faça em uma outra classe do projeto. Separação de responsabilidades porém é chave em OO.

Classe e Método Main

- ✧ Vamos a um exemplo com uma classe Main:

```
public class Main {  
  
    public static void main(String[] args) {  
        GerenciadorDeFormas gerenciador = new GerenciadorDeFormas();  
        gerenciador.iniciar();  
    }  
  
}
```


Classe e Método Main

- ❖ O main em uma classe relevante...

```
import java.util.Date;

public class GerenciadorDeFormas {

    public static void main(String[] args) {
        GerenciadorDeFormas.iniciar();
    }

    public static void iniciar() {
        // código que inicia o gerenciador de formas....
    }
}
```


Classe e Método Main

- ❖ Para compilarmos o nosso programa, utilizamos o java compiler, que é distribuído no pacote Java SDK.
- ❖ O comando é:
 - ❖ **javac Main.java**
- ❖ O comando irá compilar a classe e gerar um arquivo **Main.class**, para executarmos ele, executamos:
 - ❖ **java Main**
- ❖ O java passará o comando para o método main da classe que especificamos no comando.

Classe e Método Main

- ❖ Se os seus arquivos Javas estão espalhados pela hierarquia do pacote, use os seguintes comandos na raiz do projeto:
 - ❖ **`javac org.meupacote/*.java`**
- ❖ Para executar rode:
 - ❖ **`java org.meupacote.Main`**

Classe e Método Main

- ✧ Vamos ver um exemplo....
 - ✧ Código disponível no repositório do GitHub