



# Desenvolvimento OO com Java

## Pacotes

João Paulo A. Almeida

Adaptado de

Vítor E. Silva Souza

([vitorsouza@inf.ufes.br](mailto:vitorsouza@inf.ufes.br))

<http://www.inf.ufes.br/~vitorsouza>

Departamento de Informática  
Centro Tecnológico

Universidade Federal do Espírito Santo



Este obra foi licenciada sob uma Licença [Creative Commons Atribuição 3.0 Não Adaptada.](https://creativecommons.org/licenses/by-nd/3.0/)

# Pacotes (*packages*)

- À medida que **aumenta** o número de classes, aumenta a chance de **coincidência** de nomes;
- Precisamos separar as classes em **espaços** de nomes;
- Java possui o conceito de **pacotes**:
  - Espaço de nome para **evitar** conflitos;
  - **Agrupamento** de classes semelhantes;
  - Maneira de construir **bibliotecas** de classes;
  - Estabelecimento de políticas de **acesso** às classes.

- As APIs Java (ex.: Java SE) são divididas em pacotes:
  - `java.lang`: classes do núcleo da plataforma;
  - `java.util`: classes utilitárias;
  - `java.io`: classes para I/O (entrada/saída);
  - Dentre muitos outros...
- Pacotes são organizados em níveis hierárquicos:
  - `java`
    - `lang`
    - `util`
    - . . .
  - `javax`
    - `swing`
    - `xml`
    - . . .

- Coleção de arquivos .class;
- Compilados de códigos-fonte .java;
  - Geralmente uma classe pública por arquivo fonte.
- Declaração do mesmo pacote:
  - Primeira linha não comentada da classe.

```
package meupacote;

public class MinhaClasse {

}
```

# Importação de pacotes

- Para usar classes de **outros** pacotes, é preciso **importá-las**;
- Uma **IDE** ajuda nesta tarefa.

```
package outropacote;

// Importa todas as classes do meupacote.
import meupacote.*;

public class OutraClasse {
    MinhaClasse mc;
}
```

# Outras opções de importação

```
package outropacote;  
  
// Importa uma classe específica.  
import meupacote.MinhaClasse;  
  
public class OutraClasse {  
    MinhaClasse mc;  
}
```

```
package outropacote;  
  
public class OutraClasse {  
    // Uso do nome completo da classe.  
    meupacote.MinhaClasse mc;  
}
```

- As **classes** do pacote `java.lang` são importadas automaticamente;
- Não é necessário:
  - `import java.lang.String;`
  - `import java.lang.Math;`
  - `import java.lang.*;`

# Importação estática

- A partir do Java 5 é possível importar os membros estáticos de uma classe:
- Antes:

```
/* ... */  
r = Math.exp(x) + Math.log(y) -  
    Math.sqrt(Math.pow(Math.PI, y));
```

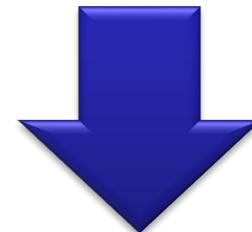
- Depois:

```
import static java.lang.Math.*;  
  
/* ... */  
r = exp(x) + log(y) - sqrt(pow(PI, y));
```

Também pode importar somente um específico.

- Para não haver **conflito** com ninguém, sugere-se usar seu **domínio** na Internet ao contrário:

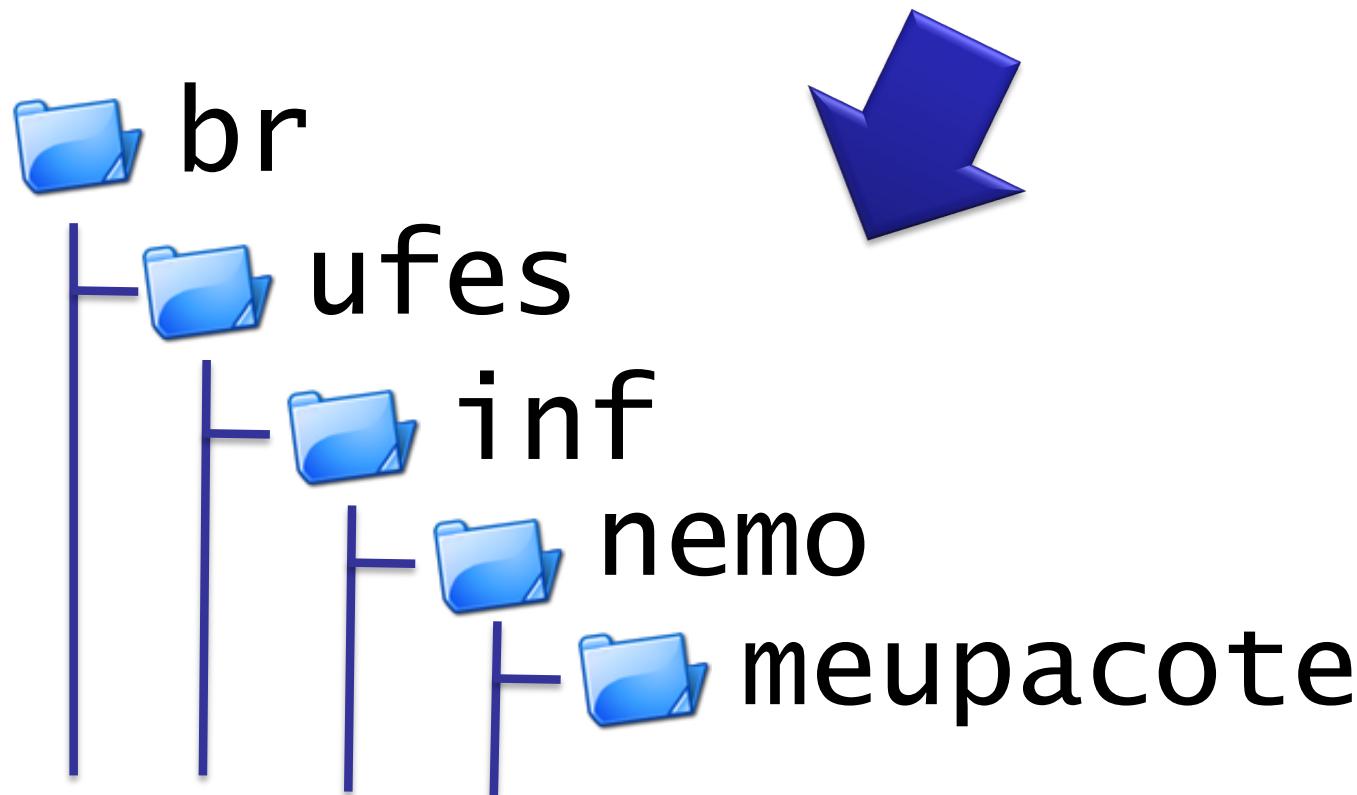
`http://nemo.inf.ufes.br`



`br.ufes.inf.nemo`

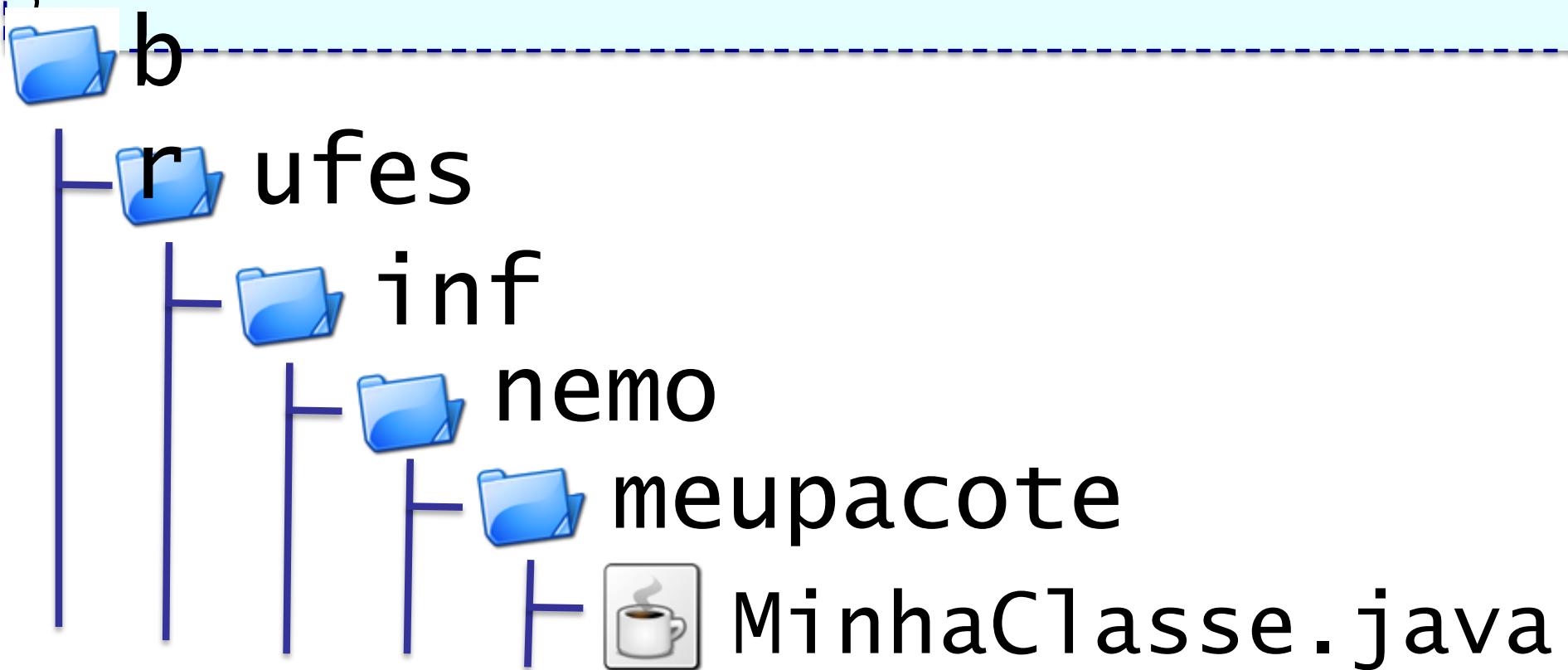
- Como dispor arquivos .class em pacotes?
- Maioria das JVMs utiliza pastas no sistema de arquivos do SO:

br.ufes.inf.nemo.meupacote



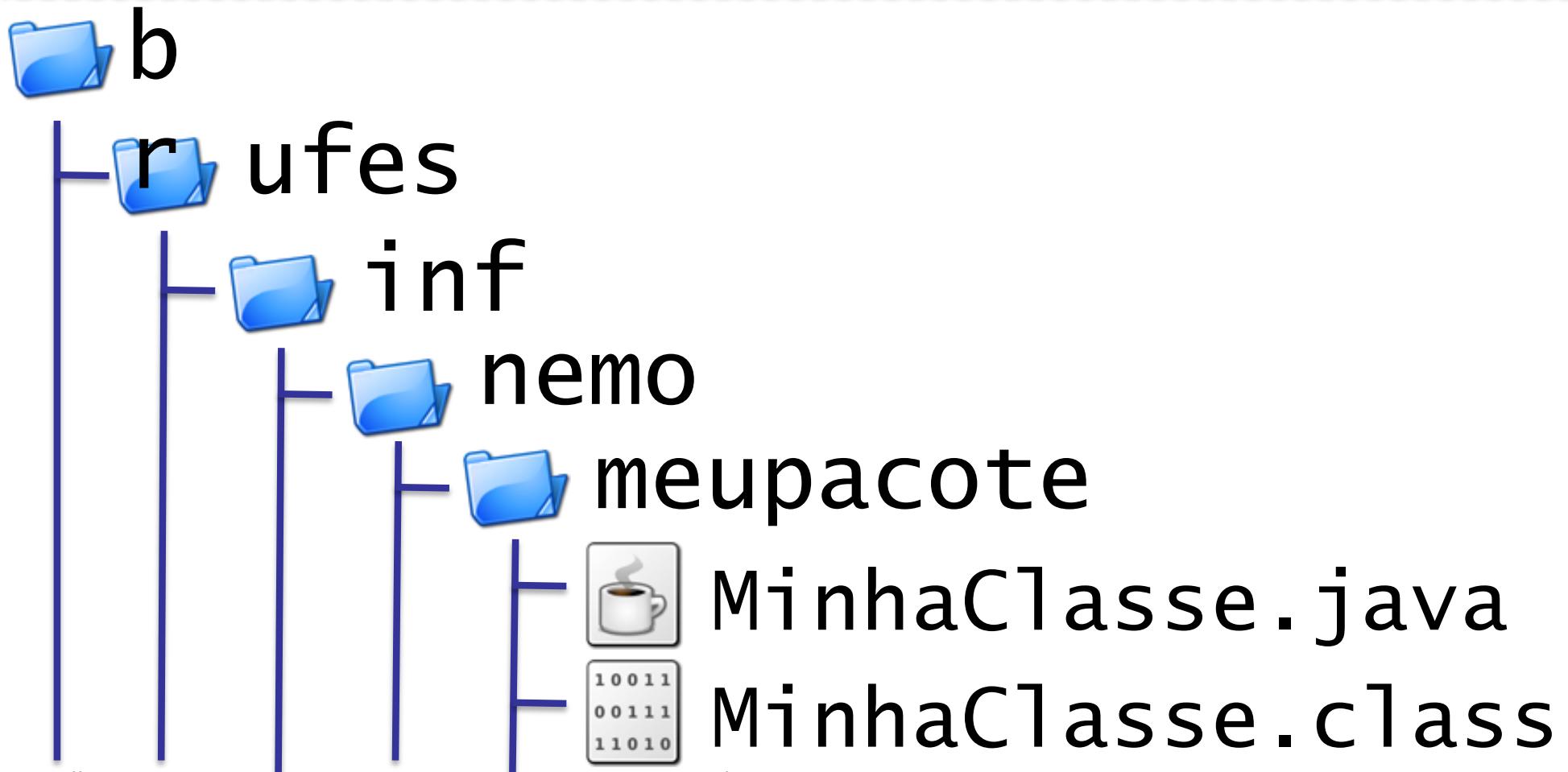
# Localização de pacotes

```
package br.ufes.inf.nemo.meupacote;
import java.util.Date;
public class MinhaClasse {
    public static void main(String[] args) {
        System.out.println(new Date());
    }
}
```



# Localização de pacotes

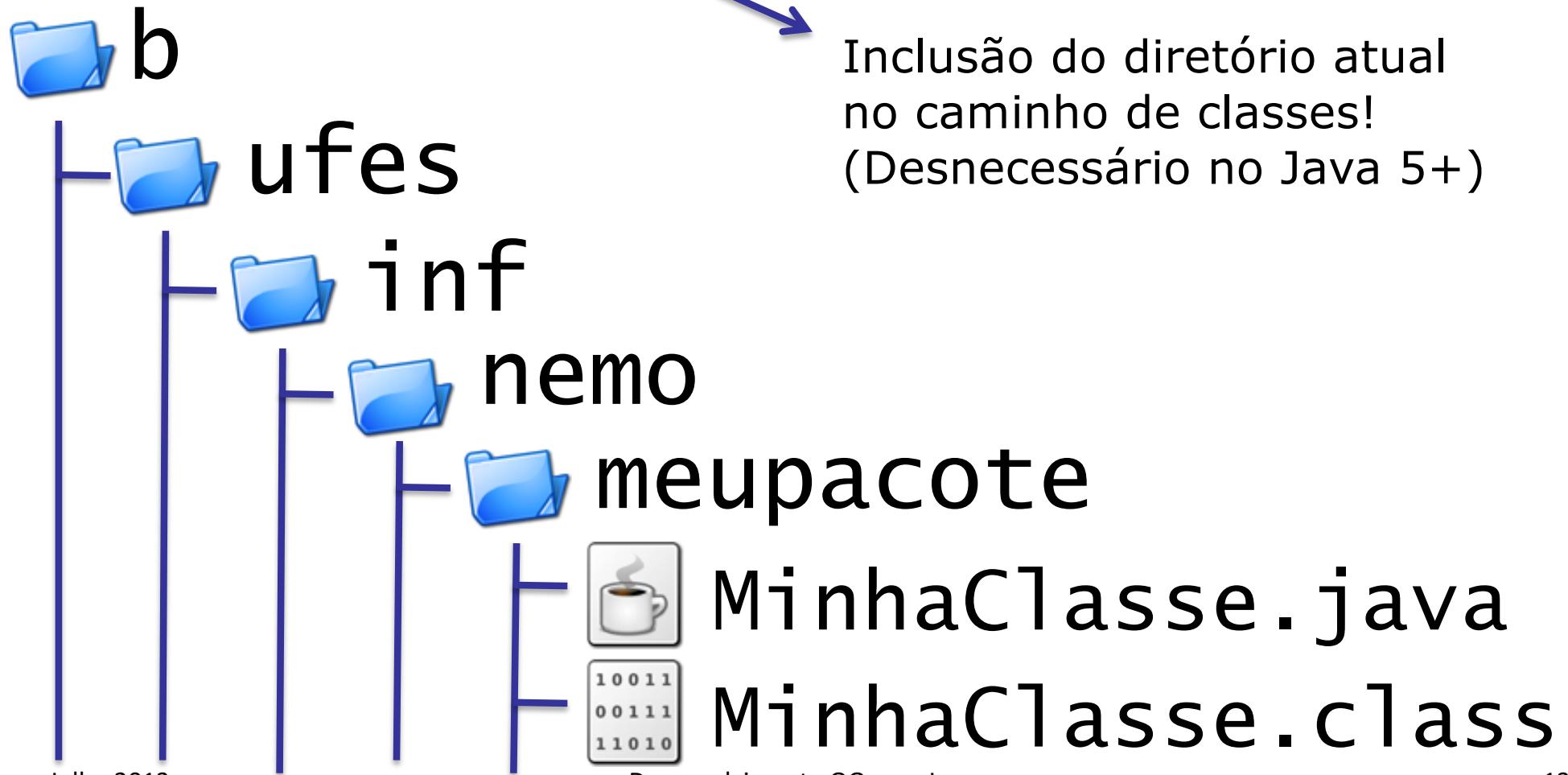
```
$ ls  
br  
  
$ javac br/ufes/inf/nemo/meupacote/MinhaClasse.java
```



# Localização de pacotes

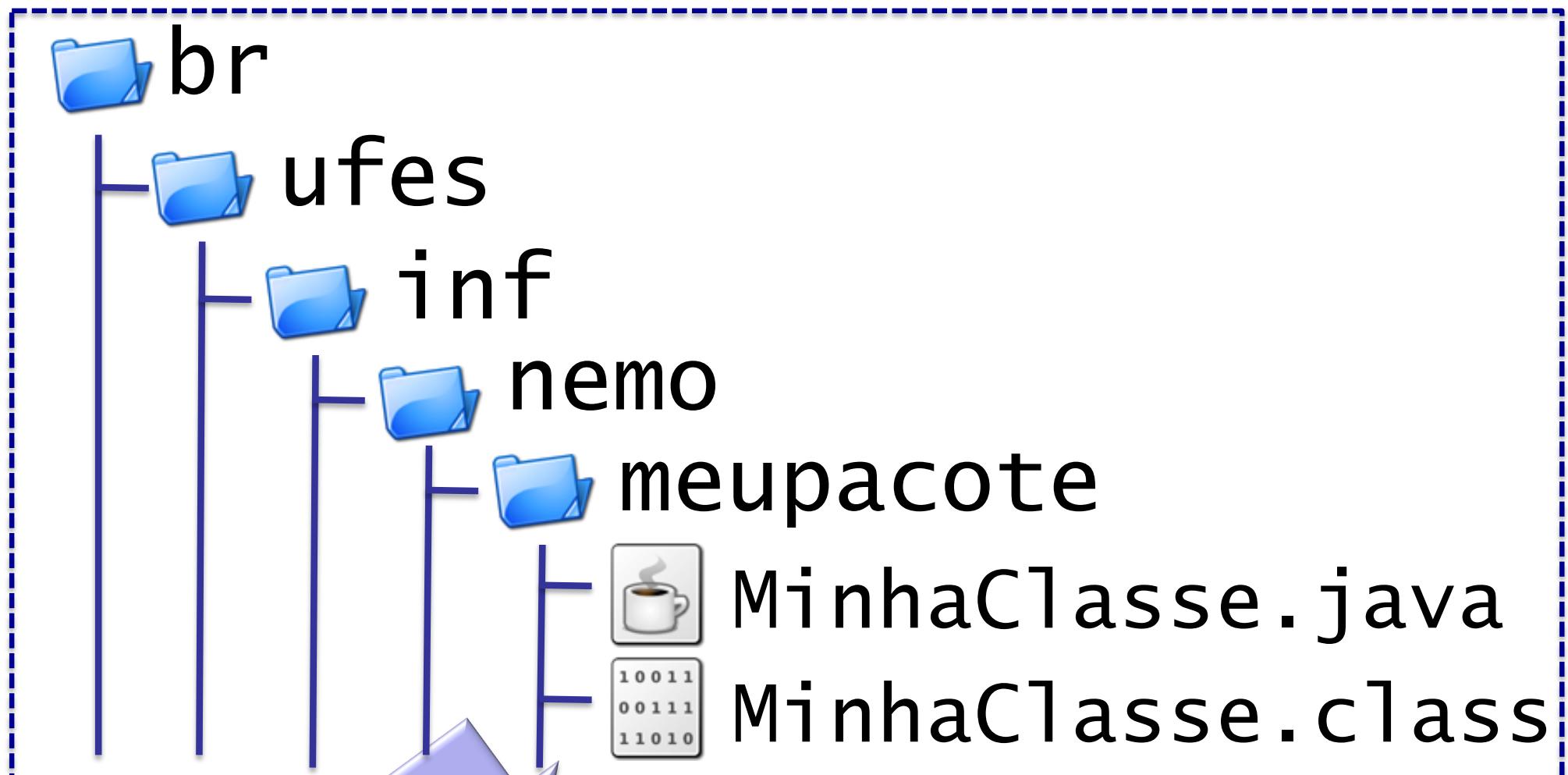
```
$ java -cp . br.ufes.inf.nemo.meupacote.MinhaClasse
```

Wed Jun 05 21:01:29 BRT 2013



- O “caminho de classes” ou “trilha de classes” é onde as ferramentas do JDK e a JVM **procuram** classes;
  - A partir dos **diretórios** do *classpath* procura-se as classes segundo seus **pacotes** (usa a 1<sup>a</sup> encontrada).
- Estão por **padrão** no *classpath*:
  - A biblioteca de classes da **API** Java SE;
  - O diretório **atual**.
- O *classpath* pode ser **alterado**:
  - Variável de **ambiente** (não recomendado);
  - Opção **-classpath** ou **-cp**.

- Ao **compilar** uma classe, se ela faz referência a outra que não foi compilada, esta última é **compilada** se o código está **disponível**;
- Se já foi **compilada**, mas o arquivo fonte está com data mais **recente**, ela é recompilada.
- Uso de **IDEs**:
  - Utilizar uma IDE **abstrai** todas estas preocupações;
  - A IDE cuida de todo o **processo** de compilação.



```
jar -c -f meujar.jar br/ufes/inf/nemo/meupacote/*.class
```



**meujar.jar**  
Desenvolvimento OO com Java

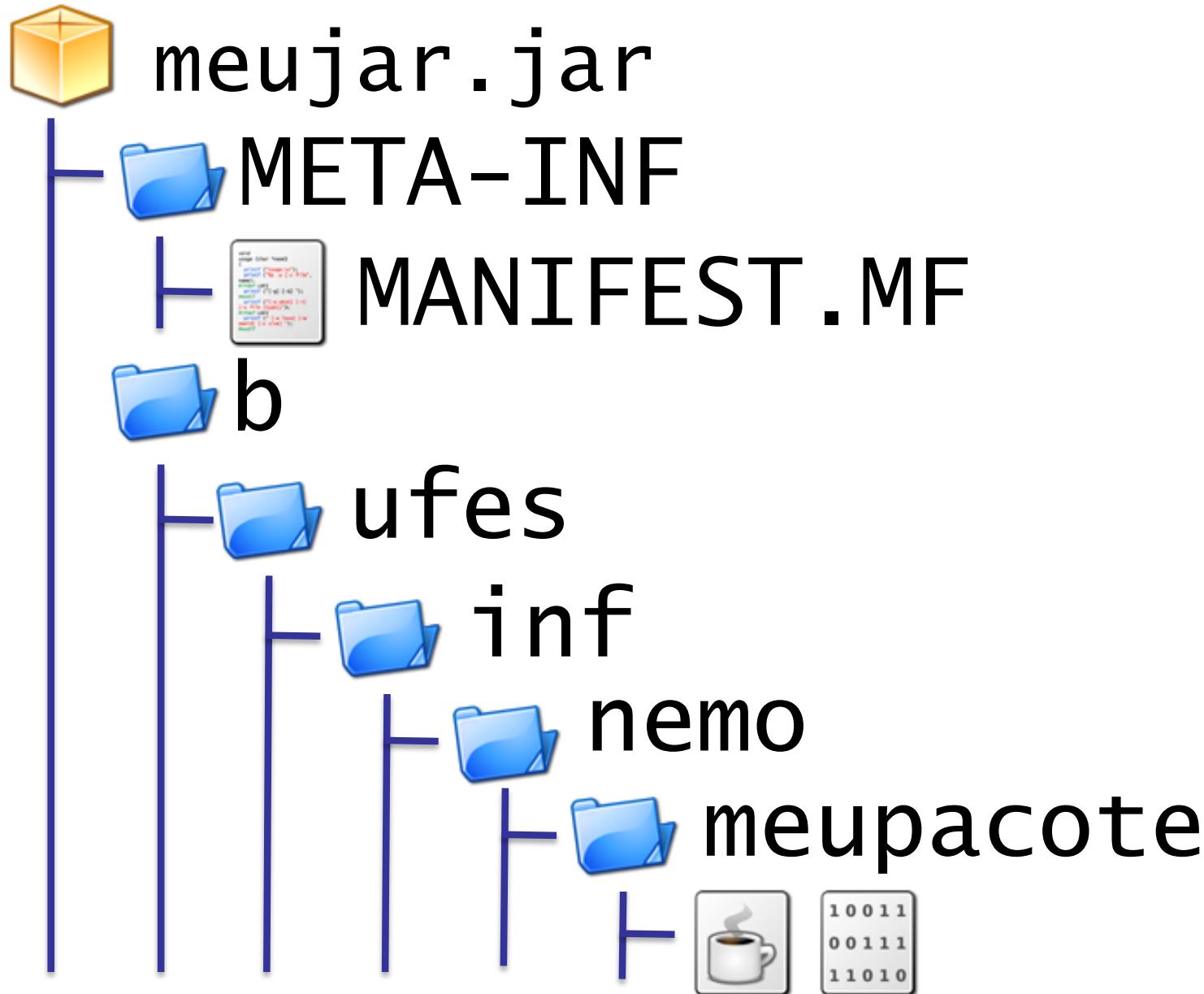
```
$ java -cp meujar.jar  
br.ufes.inf.nemo.meupacote.MinhaClasse
```

Wed Jun 05 21:15:06 BRT 2013



## meujar.jar

Arquivos JAR são compactados no formato ZIP e podem ser abertos por qualquer programa compatível.



# O arquivo MANIFEST

- Contém **meta-dados** sobre o pacote:
- Crie um **arquivo MANIFEST.MF**:

```
Main-Class: br.ufes.inf.nemo.meupacote.MinhaClasse
```

- Digite os seguintes **comandos**:

```
$ jar -c -f meujar.jar -m MANIFEST.MF  
br/ufes/inf/nemo/meupacote/*.class
```

```
$ java -jar meujar.jar  
Wed Jun 05 21:23:03 BRT 2013
```

- Toda classe que não especifica o pacote pertence ao pacote padrão;
- Seu .class deve estar numa pasta do *classpath*.

```
public class Bolo {  
    public static void main(String[] args) {  
        // Não há import, estão no mesmo pacote.  
        Torta t = new Torta();  
        t.f();  
    }  
}  
  
class Torta {  
    void f() { System.out.println("Torta.f()"); }  
}
```

# Especificadores de acesso

- Determinam a visibilidade de um determinado membro da classe com relação a outras classes;
- Há quatro níveis de acesso:
  - Público (`public`);
  - Protegido (`protected`);
  - Privado ao pacote (`package-private`). Default, ou seja, sem modificador
  - Privado (`private`);

Membro	Resultado
Classes	Classes públicas* podem ser importadas por qualquer classe.
Atributos	Atributos públicos podem ser lidos e alterados por qualquer classe.
Métodos	Métodos públicos podem ser chamados por qualquer classe.

\* Só pode haver uma classe pública por arquivo-fonte e os nomes (da classe e do arquivo) devem ser iguais.

# Membros públicos

```
public class A {  
    public int x = 10;  
    public void print() {  
        System.out.println(x);  
    }  
}
```

```
public class B {  
    public A a = new A();  
    public void f() {  
        a.x = 15;  
        a.print();  
    }  
}
```

 **letras**  
├  **A.java**  
└  **B.java**  
  
 **numeros**  
└  **Um.java**

```
import letras.B;  
public class Um {  
    B b = new B();  
    public void g() {  
        b.f();  
    }  
}
```

Membro	Resultado
Classes	Somente classes internas* podem ser declaradas privadas.
Atributos	Atributos privados só podem ser lidos e alterados pela própria classe.
Métodos	Métodos privados só podem ser chamados pela própria classe.

\* Tópico avançado, veremos posteriormente.

# Membros privados

```
public class A {
    private int x = 10;
    private void print() {
        System.out.println(x);
    }
    void incr() { x++; }
}
```

```
public class B {
    public A a = new A();
    public void f() {
        // Erro: a.x = 15;
        // Erro: a.print();
    }
}
```

 **letras**

-  **A.java**
-  **B.java**

 **numeros**

-  **Um.java**

```
import letras.B;
public class Um {
    B b = new B(); // OK
    public void g() {
        b.f(); // OK
    }
}
```

# Default (package-private)

Membro	Resultado
Classes	Classes package-private só podem ser utilizadas por classes do mesmo pacote.
Atributos	Atributos package-private só podem ser lidos e alterados por classes do mesmo pacote.
Métodos	Métodos package-private só podem ser chamados por classes do mesmo pacote.

# Membros package-private

```
class A {
    int x = 10;
    void print() {
        System.out.println(x);
    }
    void incr() { x++; }
}
```

```
public class B {
    A a = new A();
    public void f() {
        a.x = 15; // OK
        a.print(); // OK
    }
}
```

 **letras**  
 └──  **A.java**  
 └──  **B.java**  
 **numeros**  
 └──  **Um.java**

```
import letras.*;
public class Um {
    // Erro: A a;
    B b = new B();
    public void g() {
        // b.a.incr();
        b.f();
    }
}
```

Membro	Resultado
Classes	Somente classes internas* podem ser declaradas protegidas.
Atributos	Atributos protegidos só podem ser lidos e alterados por classes do mesmo pacote ou subclasses
Métodos	Métodos protegidos só podem ser chamados por classes do mesmo pacote ou subclasses

\* Tópico avançado, veremos posteriormente.

Acesso	Público	Protegido	Package -Private	Privado
Mesma classe	Sim	Sim	Sim	Sim
Classe no mesmo pacote	Sim	Sim	Sim	Não
Subclasse em pacote diferente	Sim	Sim	Não	Não
Não-subclasse em pacote diferente	Sim	Não	Não	Não