

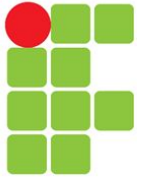


Componentes em Java com Pacotes e arquivos JAR

Desenvolvimento de Componentes (BRADECO)

Prof. Luiz Gustavo Diniz de Oliveira Vêras

E-mail: gustavo_veras@ifsp.edu.br



Conteúdo

- ✓ Ferramentas
- ✓ Pacotes
- ✓ Diagramas de Pacotes
- ✓ Compartilhando classes com Java ARchive (JAR)
- ✓ Adicionando um novo arquivo.jar ao Classpath
- ✓ Exercícios



Ferramentas

Se você não possui o VS Code:

- Visual Studio Code (Pacote com SDK embutido)-
<https://code.visualstudio.com/docs/languages/java>

Se você já possui o VS Code instalado:

- Instalar o pacotes de Extensões para Java no VSCode -
<https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack>
- Java JDK - Versão 21 - <https://bell-sw.com/pages/downloads/#jdk-21-lts>

Obs: Usualmente, a instalação da extensão já instala o JDK também.



Ferramentas

Vamos criar um projeto no Visual Studio Code.

Link: <https://code.visualstudio.com/docs/java/java-project>



Pacotes

São grupos de **tipos de dados (classes)** relacionados à um mesmo contexto que fornecem proteção de acesso e gerenciamento de **namespaces**.

Tipos podem ser **classes, interfaces, enums e annotations**.

Organizam o código de forma relacional.





Pacotes

Vantagens

- Com o uso de pacotes, fica fácil verificar que alguns tipos estão relacionados entre si.
- Os nomes dos tipos não irão conflitar com outros tipos de mesmo nome, devido ao *namespace* criado pelo pacote.
- É possível atribuir acesso restrito aos tipos inseridos em um pacote. Por exemplo, classes de outro pacote podem ter acesso bloqueado ao pacote restrito.





Pacotes

Criando pacotes:

Inserir o código abaixo no início de cada arquivo que faz parte do pacote.

```
package classes;
```

Classes do pacote classes

```
package classes;  
Class ClasseA{}
```

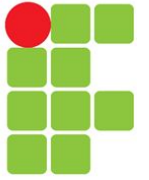
```
package classes;  
Class ClasseB{}
```

```
package classes;  
Class ClasseC{}
```

```
package classes;  
Class ClasseD{}
```

```
Class ClasseE{}
```

Classes que não estão inseridas em nenhum pacote ficam no pacote padrão (default package).



Pacotes

Usando pacotes

Inserir o pacote no início de cada arquivo, como no código abaixo, permite usar os tipos daquele pacote.

```
import classes.ClasseA;

class ClasseExterna{
    ClasseA a;
}
```

Outros modos de usar pacotes

Fazendo o **import** de várias classes do pacote.

```
import classes.A;
import classes.B;

class ClasseExterna{
    ClasseA a;
    ClasseB b;
}
```

O ***** importa todos os tipos do pacote.

```
import classes.*;

class ClasseExterna{
    ClasseA a;
    ClasseB b;
}
```

Podemos referenciar o nome da classe com o pacote dentro do código para diferenciarmos de outras classes com mesmo nome.

```
import classes.ClasseA;
class ClasseExterna{
    ClasseA a;
    outropacote.ClassA a2;
}
```




Pacotes

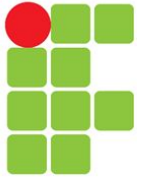
Nomeando pacotes

- Pacotes devem ter nomes em minúsculas.
- Usam o nome do domínio da organização invertido.

Exemplo:

Domínio: bra.ifsp.edu.br

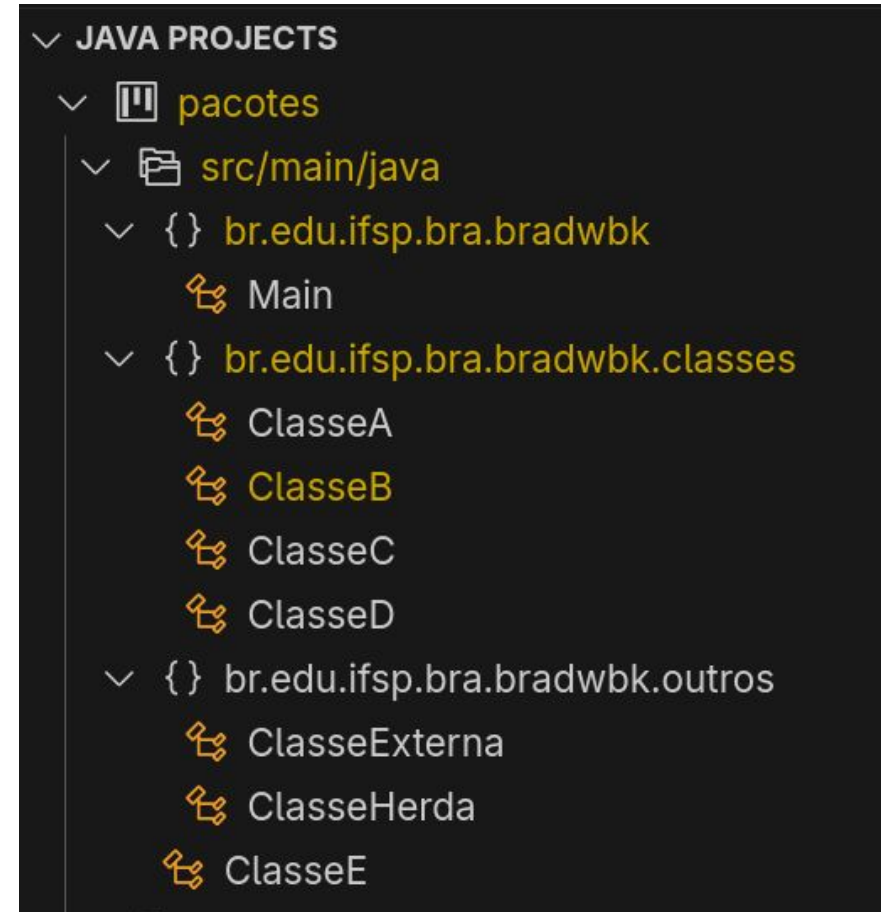
Pacote: br.edu.ifsp.bra

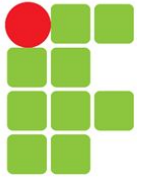


Pacotes

Estrutura no Visual Studio Code

Para o pacote raiz: bra.ifsp.edu.br.bradwbk





Pacotes

Pacotes e Modificadores de acesso

Os modificadores de acesso alteram o nível de permissão (segurança) dos membros de uma classe.

Podem ser aplicados a **Classe**, **Construtor**, **Método** e **Atributo**.

Public

Todos podem
acessar o
componente da
classe com este
modificador

Protected

Somente
subclasses, classes
no mesmo pacote e
a própria classe
podem acessar os
componentes com
este modificador

Default

Somente as classe
do seu pacote
podem visualizar
componentes com
este modificador

Private

Só a própria
classe pode
acessar os seus
componentes
com esse
modificador

Exemplo

`br.com.ifsp.classes`

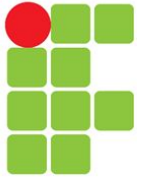
```
public class ClasseA{  
    public int valor1;  
}
```

```
public class ClasseB{  
    private int valor3;  
    protected int calc(){  
        ClassC c = new ClassC();  
        return c.valor2+1;  
    }  
}
```

```
class ClasseC{  
    int valor2;  
}
```

```
package br.edu.ifsp.bra.bradwbk.outros;  
  
import br.edu.ifsp.bra.bradwbk.classes.ClasseA;  
  
class ClasseExterna{  
    public int metodo(){  
        ClasseA a = new ClasseA();  
        return a.valor1;  
    }  
}
```

```
package br.edu.ifsp.bra.bradwbk.outros;  
  
import br.edu.ifsp.bra.bradwbk.classes.ClasseA;  
  
class ClasseHerda extends ClasseB{  
    protected int metodo(){  
        return calc();  
    }  
}
```



Pacotes

Pacotes e Modificadores de acesso

A Tabela abaixo apresenta para quem ficará acessível os elementos de classe de acordo com o seu nível de acesso

Modificador/ Acessível por	Classe	Pacote	Subclasse	Classes em outro Pacotes
public	Sim	Sim	Sim	Sim
protected	Sim	Sim	Sim	Não
default	Sim	Sim	Não	Não
private	Sim	Não	Não	Não



Pacotes

Pacotes e Modificadores de acesso

- Observe que uma classe não pode ser **private** (o que a tornaria inacessível para qualquer outra classe, exceto para a própria classe) ou **protected**.
- Então você tem apenas duas opções para acesso à classe: acesso **default** ou **public**.
- Se você não quer que mais ninguém tenha acesso a essa classe, você pode fazer todos os construtores serem **private**, impedindo assim que alguém, exceto a própria, dentro de um membro estático, crie um objeto dessa classe.

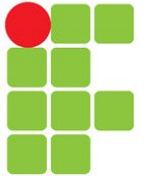
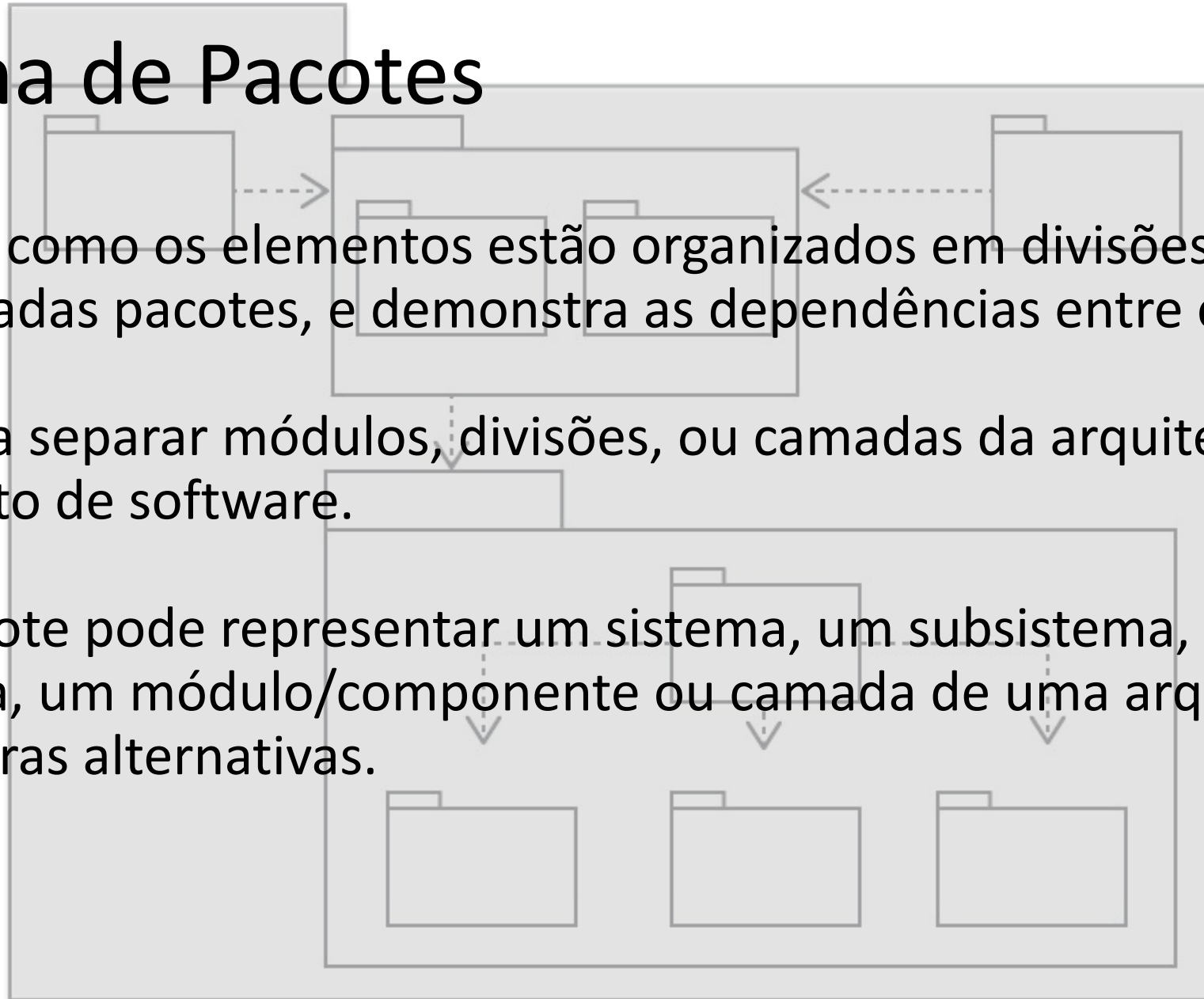


Diagrama de Pacotes

- Descreve como os elementos estão organizados em divisões lógicas, denominadas pacotes, e demonstra as dependências entre eles;
-
- É útil para separar módulos, divisões, ou camadas da arquitetura de um projeto de software.
-
- Cada pacote pode representar um sistema, um subsistema, uma biblioteca, um módulo/componente ou camada de uma arquitetura, entre outras alternativas.



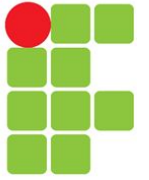


Diagrama de Pacotes

Vamos tomar a figura abaixo como exemplo:

A notação para um pacote é a de uma pasta com uma aba!



As ligações entre pacotes na Figura acima são relacionamentos de dependência entre os pacotes. Um pacote **Payment** depende de outro **Shopping cart** se algum elemento contido em **Payment** depende de algum elemento contido em **Shopping cart**.

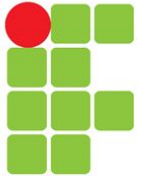


Diagrama de Pacotes

Exemplo

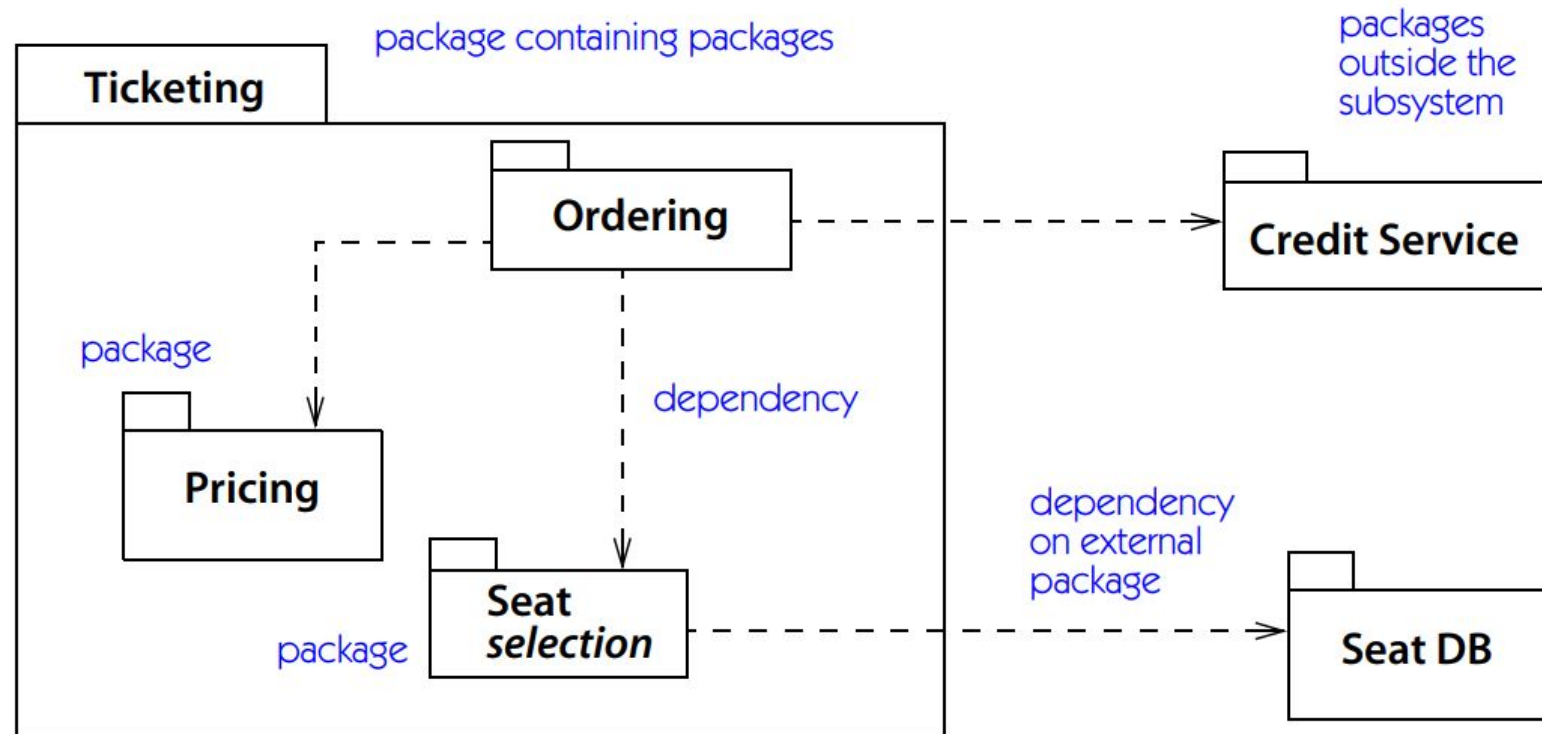
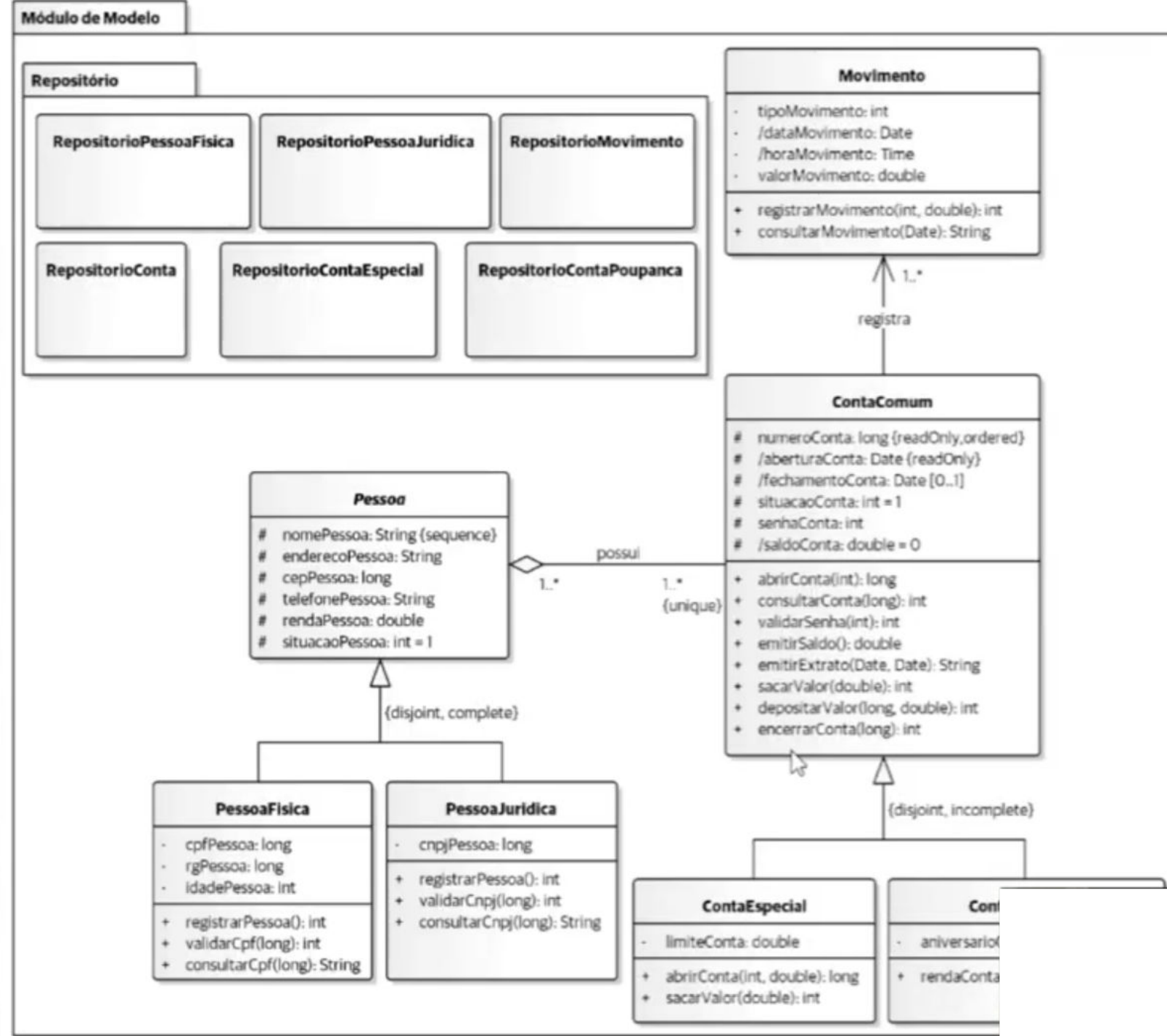


Figure 11-1. Packages and their relationships

Diagrama de Pacotes

Como especificar quais classes estão dentro de um pacote?



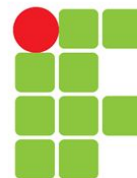


Gerando arquivos .jar

JAR significa Java ARchive.

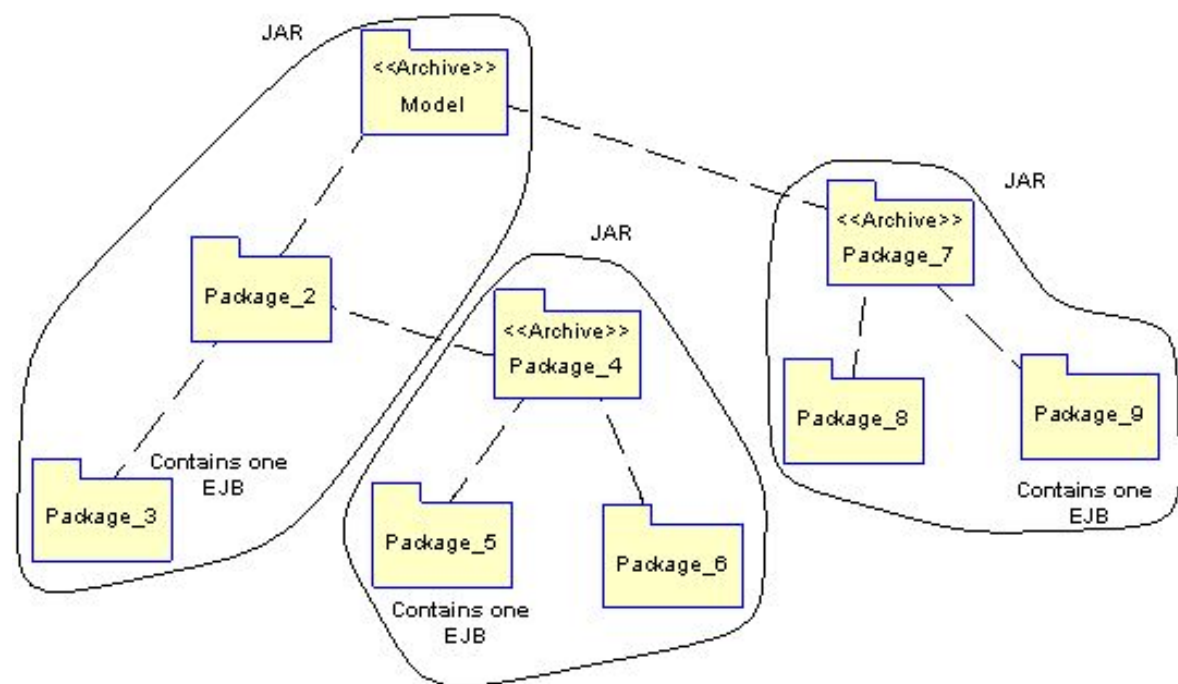
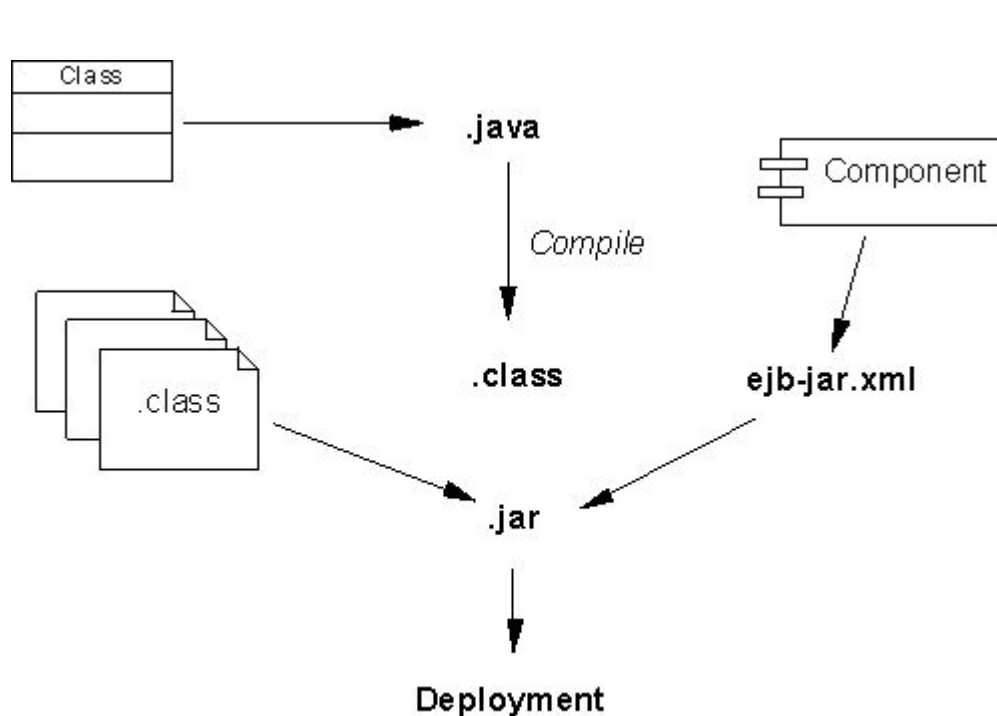
É um formato de arquivo baseado no popular formato ZIP e é usado para agregar vários arquivos em um.

Embora o JAR possa ser usado como uma ferramenta geral de arquivamento, a principal motivação para seu desenvolvimento foi permitir que applets Java e seus componentes necessários (arquivos .class, imagens e sons) pudessem ser baixados para um navegador em uma única transação HTTP, em vez de abrir uma nova conexão para cada item. Isso melhora significativamente a velocidade com que um applet pode ser carregado em uma página da web e começar a funcionar.



Gerando arquivos .jar

JAR significa Java ARchive.





Gerando arquivos .jar

Como criar arquivos .jar no Visual Studio Code?

Tutorial Link:

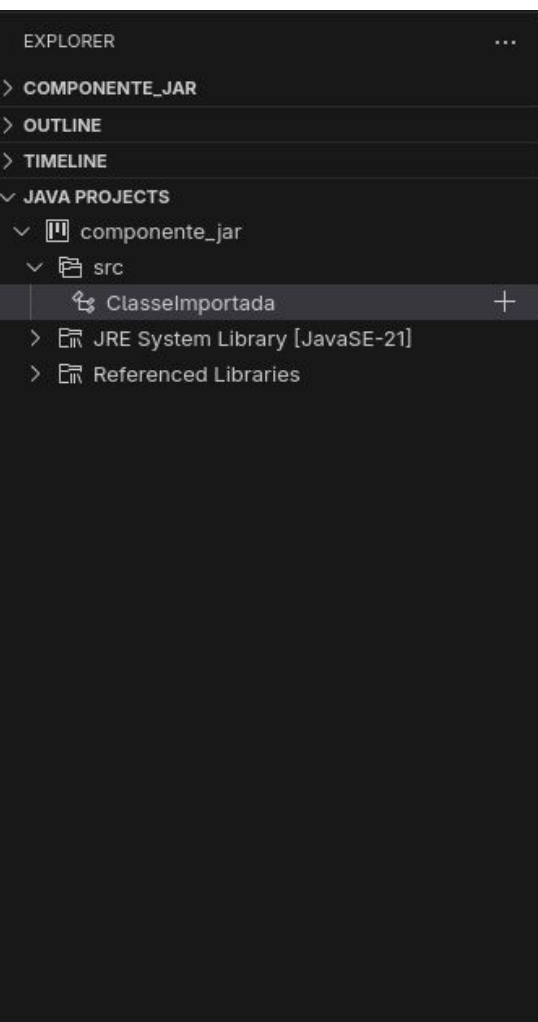
https://code.visualstudio.com/docs/java/java-project#_export-to-jar



Gerando arquivos .jar

Como criar arquivos .jar no Visual Studio Code?

1. Crie um projeto vazio
2. Implemente o código que será exportado
3. Gere o arquivo .jar, como indicado no tutorial.



```
1 public class ClasseImportada {
2     private String valor1;
3     private double valor2;
4
5     public ClasseImportada(){
6
7     }
8
9     public ClasseImportada(String valor1, double valor2) {
10         this.valor1 = valor1;
11         this.valor2 = valor2;
12     }
13
14     public String getValor1() {
15         return valor1;
16     }
17
18     public void setValor1(String valor1) {
19         this.valor1 = valor1;
20     }
21
22     public double getValor2() {
23         return valor2;
24     }
25
26     public void setValor2(double valor2) {
27         this.valor2 = valor2;
28     }
29 }
30
```



Após exportado, o seu projeto irá virá um componente .jar. Ele será salvo no diretório raiz do projeto.



/home/lgveras/dev/ifsp_aulas/bradwbk/componente_jar



bin



componente_jar.jar



lib



README.md



src



.vscode

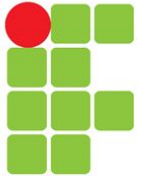


Importando Arquivos .jar

Como importar arquivos .jar no Visual Studio Code?

Tutorial Link:

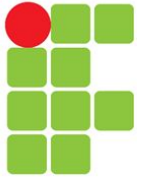
https://code.visualstudio.com/docs/java/java-project#_manage-dependencies-for-unmanaged-folder



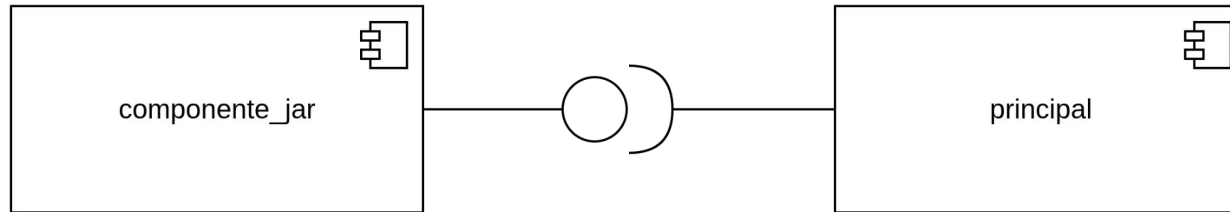
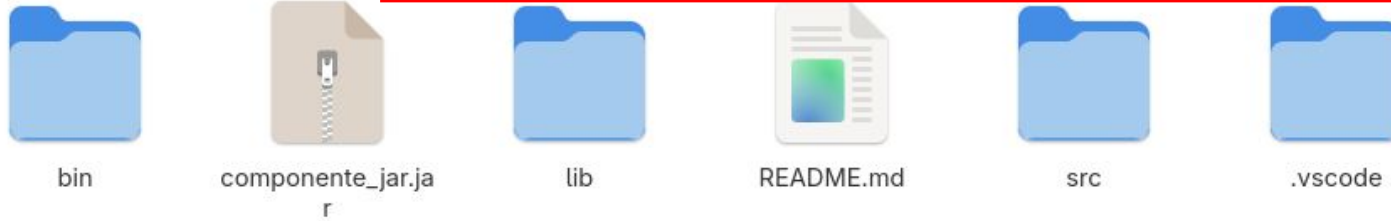
Importando Arquivos .jar

Como importar arquivos .jar no Visual Studio Code?

1. Abra um novo projeto;
2. Importe o arquivo .jar gerado anteriormente, como indicado no tutorial;
3. Utilize em seu código as classe exportada pelo .jar.



/home/lgveras/dev/ifsp_aulas/bradwbk/componente_jar

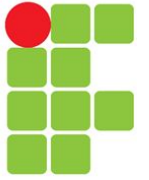


Após importado,
podemos utilizar as
classes disponíveis no
.jar.

```
> PACOTE_UNMANAGED
> OUTLINE
> TIMELINE
> JAVA PROJECTS
  > pacote_unmanaged
    > src
      > App
  > JRE System Library [JavaSE-21]
  > Referenced Libraries
    > componente_jar.jar /home/lgveras/dev/ifsp_a...

src > J App.java > ...
1  public class App {
2      Run | Debug
3      public static void main(String[] args) throws Exception {
4          ClasseImportada ci = new ClasseImportada("Valor adicionado", 500);
5          System.out.println(ci.getValor1() + "-" + ci.getValor2());
6      }
7  }
8
9
10
```

Exercícios



Exercício 1 - Crie os pacotes e classes conforme a tabela abaixo:

Nome do Pacote	Nome da Classe
br.ifsp.bra.servidor	Professor, Tecnico, Aluno
br.ifsp.bra.util	ImpressoraDadosServidor

Cada pacote deverá ser transformado em um arquivo .jar separado. Em seguida, importe esses .jar em um projeto principal, onde deve ser implementado um menu simples que permite:

- criar objetos de qualquer um dos tipos definidos nos pacotes;
- exibir suas informações utilizando a classe **ImpressoraDadosServidor**.

Demais detalhes:

- Todos os campos de classe devem ser privados;
- Cada classe deve possuir pelo menos dois construtores: um sem parâmetros (construtor padrão); outro com parâmetros (para inicializar os atributos);
- A classe **ImpressoraDadosServidor** deve implementar sobrecarga de métodos para imprimir informações de cada tipo de servidor (Professor, Tecnico, Aluno);
- O estudante deve definir as regras e lógicas de funcionamento das classes que não foram especificadas aqui no enunciado.



Exercícios

Exercício 2 - Crie os pacotes e classes conforme a tabela abaixo:

Nome do Pacote	Nome da Classe
br.ifsp.bra.servidor	Professor, Tecnico, Aluno
br.ifsp.bra.util	ImpressoraDadosServidor

Para a atividade do slide anterior, desenhe:

- Diagrama de pacotes, com as suas classes constituintes;
- Diagrama de componentes.

Exercícios



Exercício 3 - Selecione um dos exercícios da última aula para os quais criou diagramas de componentes e implemente-o em Java, criando arquivos .jar e utilizando-os em um projeto principal.

Crie as classes e regras de negócio que achar necessário, porém de forma simplificada.



Referências

- Diagrama de Pacotes - UML - Prof Gilleanes Guedes Engenharia de Software e UML. Disponível em <https://www.youtube.com/watch?v=2PhyRWgONoM>
- Rumbaugh, James et al. The unified modeling language reference Manual. Second Edition, Addison-Wesley, 2005.