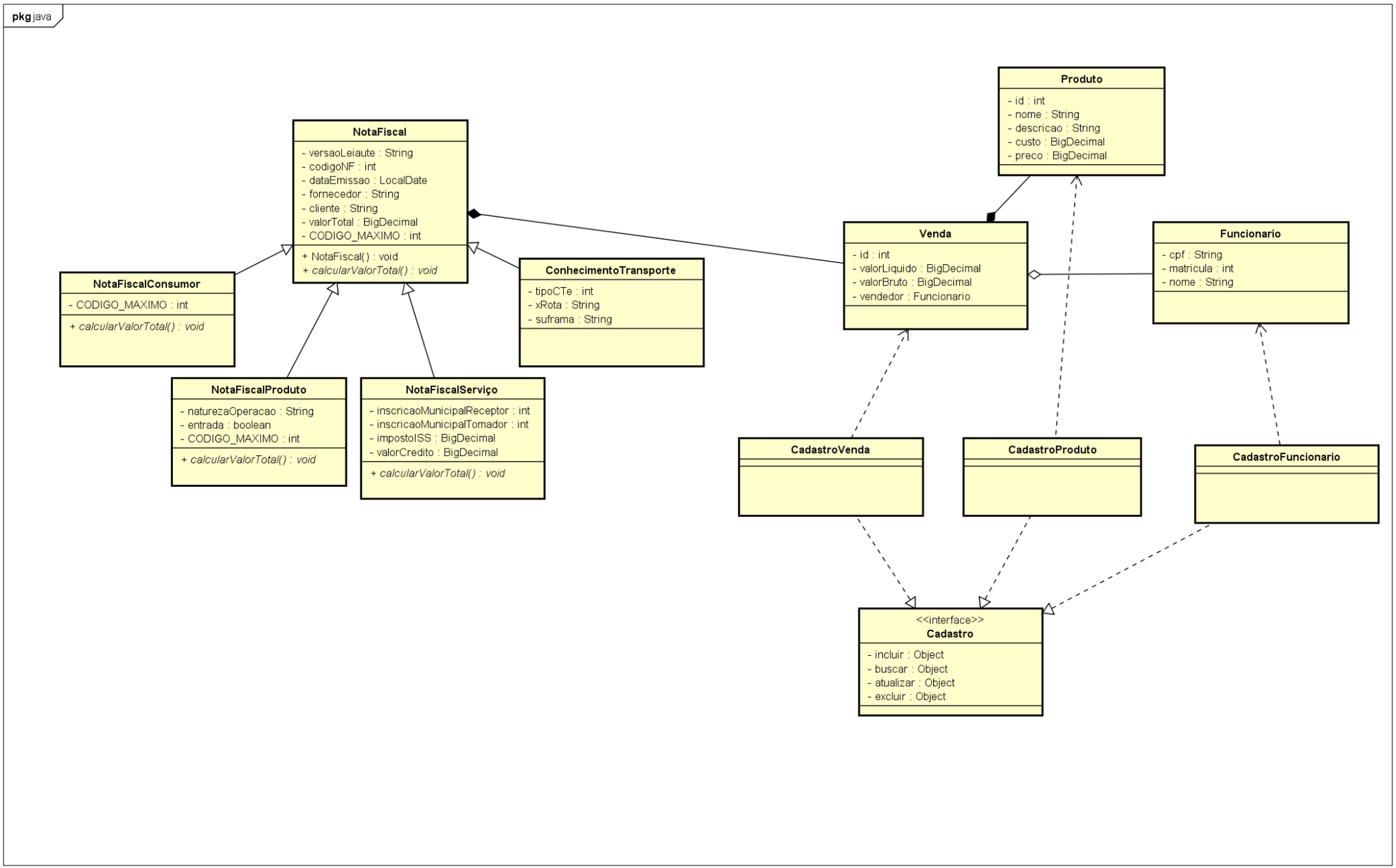
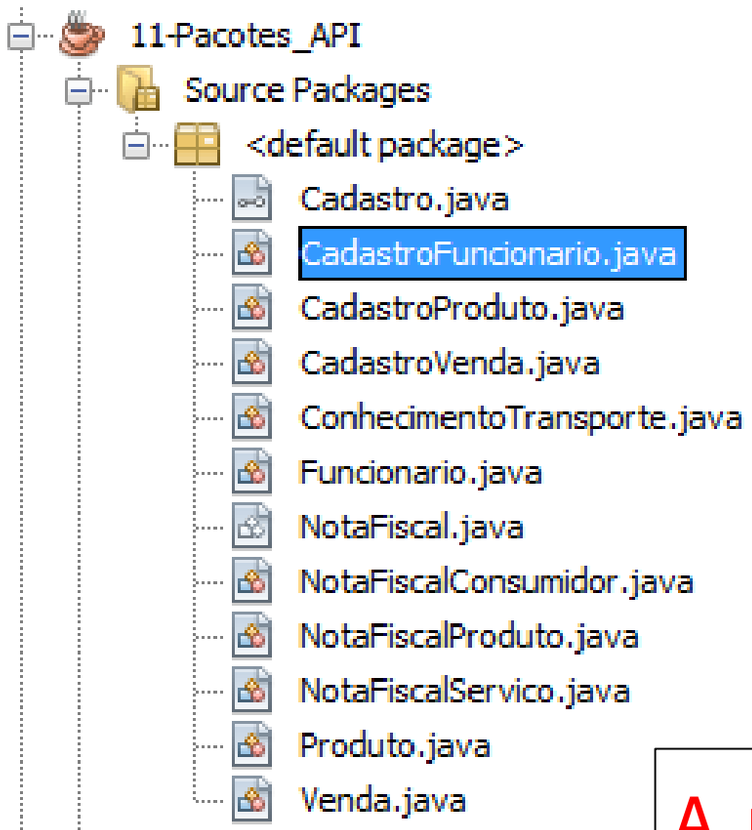
Pacotes e API do Java

Capítulo X

Um sistema de vendas hipotético

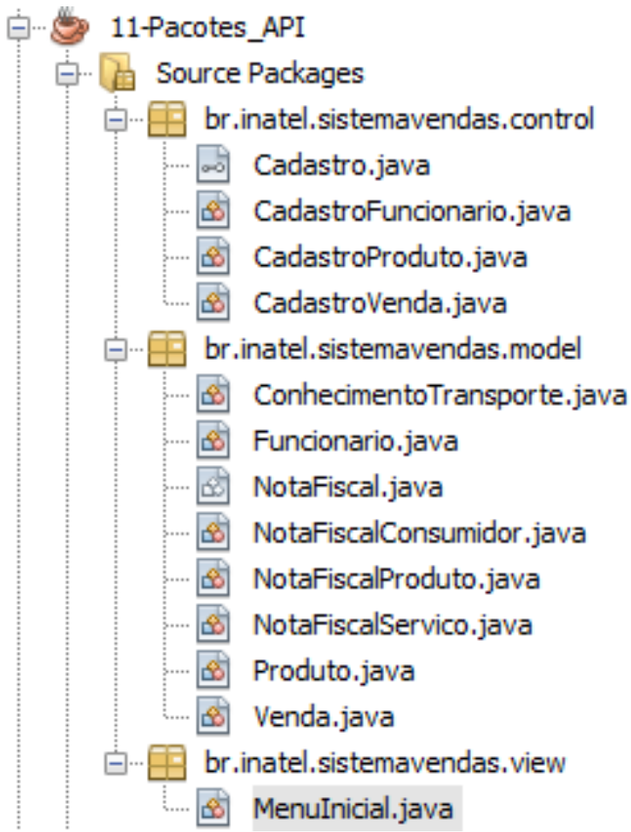


Um sistema de vendas hipotético



A medida que a quantidade de classes em nosso Sistema aumenta, as coisas não começam a ficar confusas?

Como organizar nossas classes

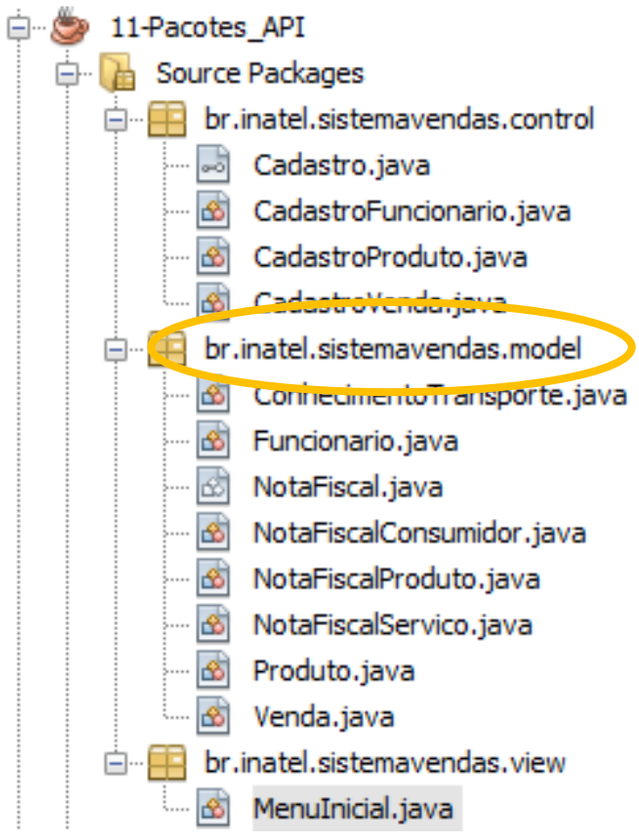


Para facilitar a localização de informações, evitar conflitos de nomes e controlar o acesso nós devemos utilizar os pacotes.

Existe uma forma “padronizada” de organizarmos nosso código. Conhecida como arquiteutra MVC.

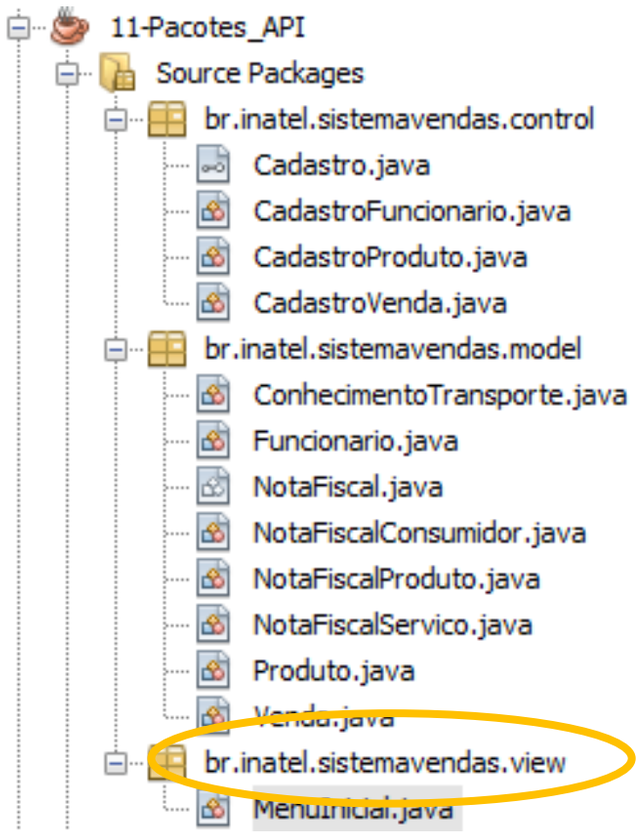
Informações: Classes, Interfaces, Enumerações e Anotações.

Como organizar nossas classes



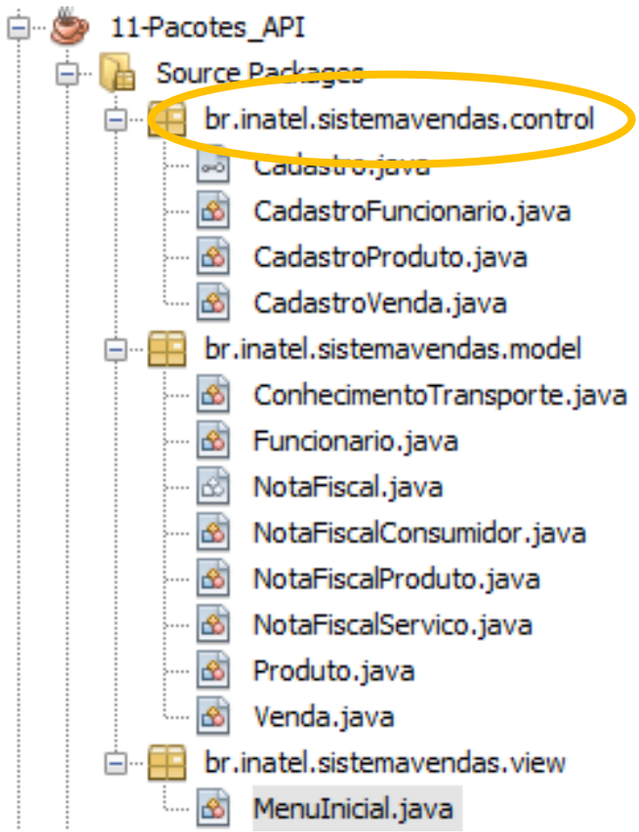
Este pacote representa os dados e as regras de negócio de nossa aplicação.
Geralmente, ele serve como uma aproximação do “mundo real”

Como organizar nossas classes



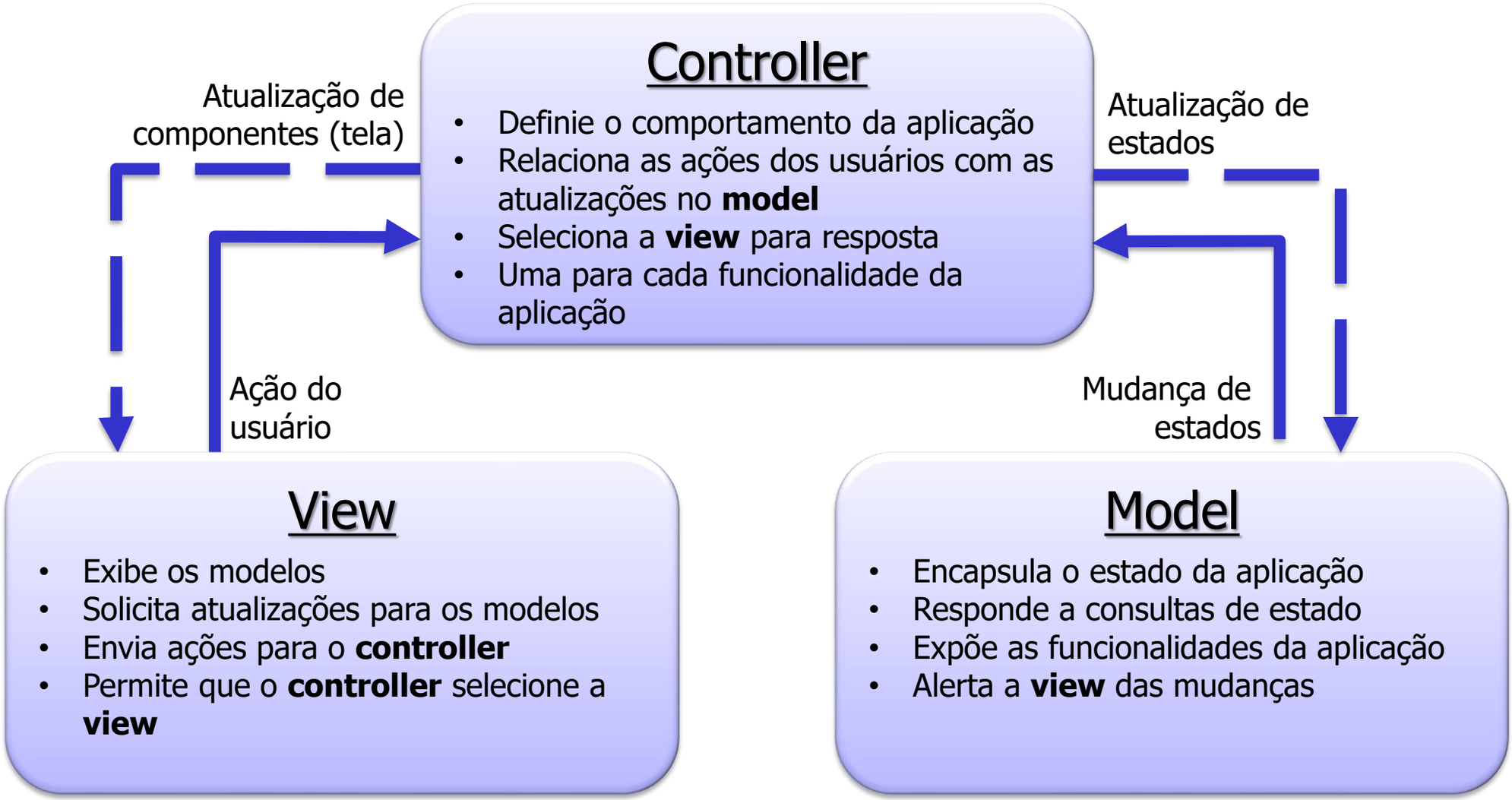
Este pacote é o responsável pela interação com o usuário, ou seja, onde ele pode visualizar e manipular os estados dos modelos através das lógicas de negócio.

Como organizar nossas classes



Controla o fluxo da aplicação, ou seja traduz as interações do usuário em ações que o modelo irá executar.
Em Java, é neste pacote que os eventos da GUI são tratados.

Como organizar nossas classes



Como organizar nossas classes

Através da palavra reservada package indicamos onde está a nossa classe/interface, esta deve ser a primeira linha da nossa classe!

```
package br.inatel.sistemavendas.view;

/**
 *
 * @author Simi
 */
public class MenuInicial {
```

Como organizar nossas classes

```java
package br.inatel.sistemavendas.control;

/**
 *
 * @author Simi
 */
public class Main {

    public static void main(String[] args) {
        br.inatel.sistemavendas.view.MenuInicial menu = new br.inatel.sistemavendas.view.MenuInicial();


    }


}
```

fully qualified name

Sempre que possível evite o acesso através do Fully Qualified Name, prefira importar as classes através do import que deve vir logo após a declaração do pacote.

```java
package br.inatel.sistemavendas.control;

import br.inatel.sistemavendas.view.MenuInicial;

/**
 *
 * @author Simi
 */
public class Main {

    public static void main(String[] args) {
        MenuInicial menu = new MenuInicial();


    }


}
```

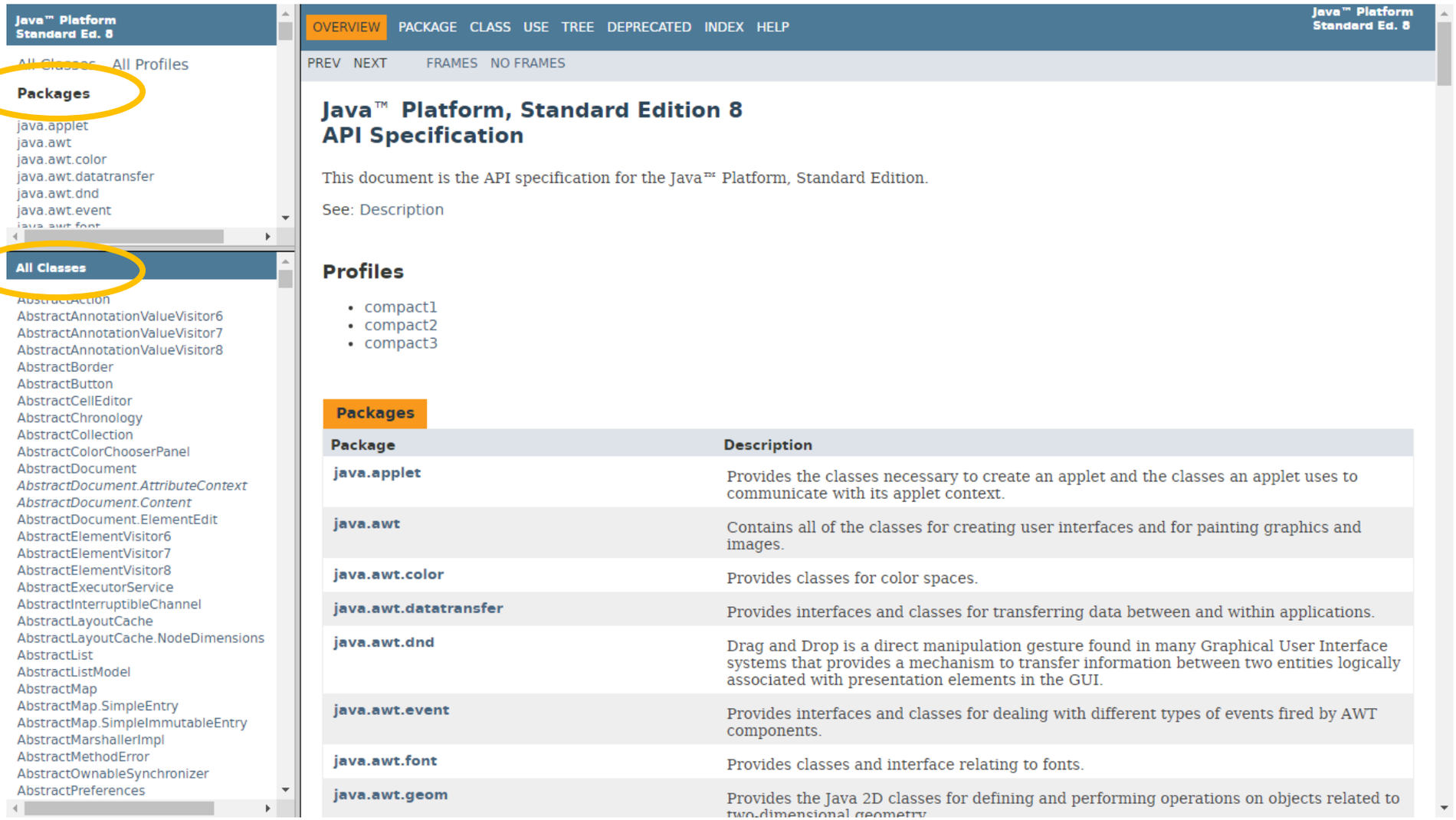
Conhecendo melhor o Java

- ✓ O que é uma API (**A**pplication **P**rogramming **I**nterface)?
  - É um conjunto de rotinas, classes, protocolos e ferramentas para a construção de *softwares*.
  - Assim como uma interface gráfica torna mais fácil para as pessoas usarem *softwares*, as APIs tornam mais fácil para os desenvolvedores a utilização de certas tecnologias na construção de aplicações. Pois, ela somente expõe os objetos ou ações abstraindo assim a forma como eles são implementados, facilitando muito o a criação de novas aplicações.

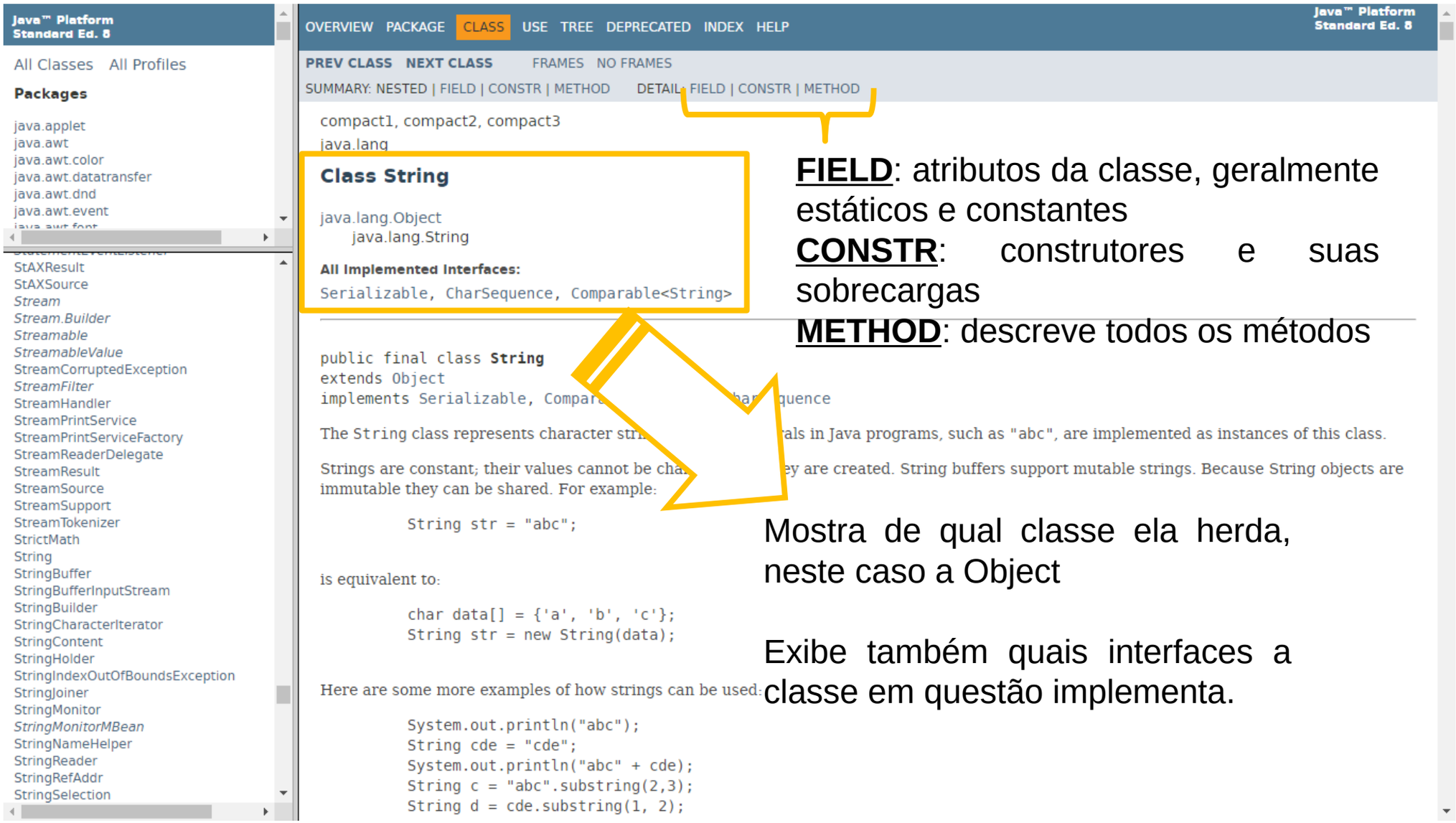
Fonte: wikipedia

Conhecendo melhor o Java

- ✓ No link: https://docs.oracle.com/javase/8/docs/api/ você encontra a API da linguagem Java



Conhecendo melhor o Java



FIELD: atributos da classe, geralmente estáticos e constantes

CONSTR: construtores e suas sobrecargas

METHOD: descreve todos os métodos

Mostra de qual classe ela herda, neste caso a Object

Exibe também quais interfaces a classe em questão implementa.

O pacote java.io

- ✓ Controla a entrada e saída de dados
  - FileInputStream → lê fluxo de bytes de um File
  - InputStreamReader → transforma os bytes em códigos Unicode
  - BufferedReader → Buffer da Entrada

O pacote java.io

```
public static void main(String[] args) {
    InputStream fluxoEntrada = null;
    InputStreamReader leitorFluxoEntrada = null;
    BufferedReader bufferEntrada = null;
    String linha = null;
    try {
        fluxoEntrada = new FileInputStream("d:\\Workspaces\\Netbeans\\log.txt");
        leitorFluxoEntrada = new InputStreamReader(fluxoEntrada);
        bufferEntrada = new BufferedReader(leitorFluxoEntrada);
        linha = bufferEntrada.readLine();
        while(linha != null){
            System.out.println(linha);
            linha = bufferEntrada.readLine();
        }
    } catch (FileNotFoundException ex) {
        System.err.printf("FileNotFoundException: %s.%n", ex);
    } catch (IOException ex) {
        System.err.printf("IOException: %s.%n", ex);
    } finally {
        try {
            bufferEntrada.close();
        } catch (IOException ex) {
            System.err.printf("IOException: %s.%n", ex);
        }
    }

}
```

O pacote java.io

- ✓ Controla a entrada e saída de dados
  - FileOutputStream → escreve fluxo de bytes de um File
  - OutputStreamReader → transforma códigos unicode em bytes
  - BufferedWriter → Buffer de Saida

O pacote java.io

```
public static void main(String[] args) {
    OutputStream fluxoSaida = null;
    OutputStreamWriter geradorFluxoSaida = null;
    BufferedWriter bufferSaida = null;
    String linha = null;
    try {
        fluxoSaida = new FileOutputStream("d:\\Workspaces\\Netbeans\\log.txt", true);
        geradorFluxoSaida = new OutputStreamWriter(fluxoSaida);
        bufferSaida = new BufferedWriter(geradorFluxoSaida);
        linha = "Vamos gerar uma nova informação e salvar no arquivo...";
        bufferSaida.write(linha);
        bufferSaida.newLine();
    } catch (FileNotFoundException ex) {
        System.err.printf("FileNotFoundException: %s.%n", ex);
    } catch (IOException ex) {
        System.err.printf("IOException: %s.%n", ex);
    } finally {
        try {
            bufferSaida.close();
        } catch (IOException ex) {
            System.err.printf("IOException: %s.%n", ex);
        }
    }
```

Classes para manipular arquivos

- Object
  - File
  - InputStream
    - FileInputStream
    - FilterInputStream
      - DataInputStream
  - OutputStream
    - FileOutputStream
    - FilterOutputStream
      - DataOutputStream
  - RandomAccessFile

Classes para entrada ou saída baseada em bytes

  - Reader
    - BufferedReader
    - InputStreamReader
      - FileReader
  - Writer
    - OutputStreamWriter
      - FileWriter
    - PrintWriter

Classes para entrada ou saída baseada em caracteres

Classes para manipular arquivos

```
=======================
FileOutputStream          (Input)  bytes
OutputStreamWriter        (Reader) caracteres
BufferedWriter            (Reader) string
=======================
FileWriter                (Input) caracteres
BufferedWriter            (Reader) string
=======================
RandomAccessFile          Ler e escreve em arquivos
=======================
FileOutputStream          (Input) bytes
ObjectOutputStream        (Input) objetos
```

Exercícios

Faça crítica das entradas de dados, armazene as informações e liste o conteúdo em arquivo

1. Implemente uma calculadora que realize as seguintes operações entre 2 números: soma, subtrair, dividir e multiplicar. (as operações realizadas deverão ser armazenadas em arquivo, ex.: “2 + 3 = 5”)

2. Implemente um programa para o controle de inventário de equipamentos da sua empresa. Neste primeiro momento serão levantados notebooks e smartphones:
   i. Notebooks: Marca, modelo, matrícula do responsável e número de série do aparelho
   ii. Smartphone: Marca, modelo, IMEI, Centro de Custo e matrícula do responsável

3. Você trabalha em um grande varejista, e foi designado para criar um sistema de cadastro das filiais da empresa. Devido ao tamanho de algumas filiais elas podem implementar uma função de Centro de Distribuição, ou seja, elas recebem as mercadorias dos fornecedores e as transferem para unidades menores.
   i. Pessoa: matrícula, nome, telefone e e-mail
   ii. Filial: código, endereço, pessoa responsável
   iii. Mercadoria: código, descrição e valor
   iv. Centro de Distribuição: transfere várias mercadorias de uma filial para outra

Exercícios

Crie um sistema de cadastro de pessoas utilizando de JOptionsPane, armazene os dados em um array e ao final salve os dados em um arquivo .txt. Quando o programa for aberto novamente deverá resgatar todos os dados já cadastrados.

Pessoa: Nome, Endereço.
Pessoa Física: CPF, Idade, Sexo.
Pessoa Jurídica: CNPJ, Ramo.

Obs: Não se esqueça da critica de dados, tratamento de exceções e encapsulamento de dados