

Fundamentos de Java

Organização do Código Java



Tópicos Abordados

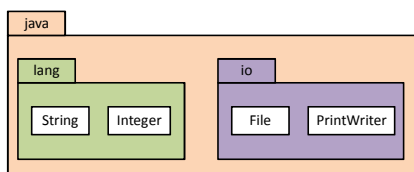


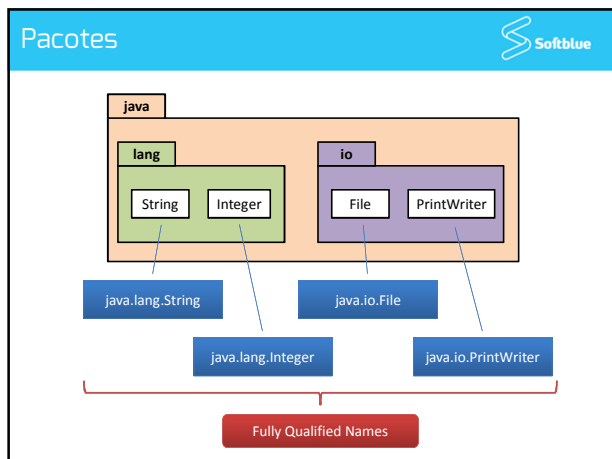
- Pacotes
 - Mapeamento de um pacote
 - Convenção no nome dos pacotes
- O uso do *import*
- Visibilidades *package* e *protected*
- Javadoc
- Criação de arquivos JAR
- Convenções do código Java

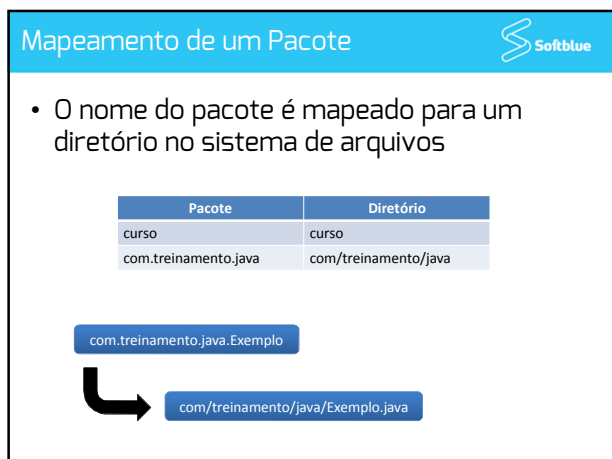
Pacotes

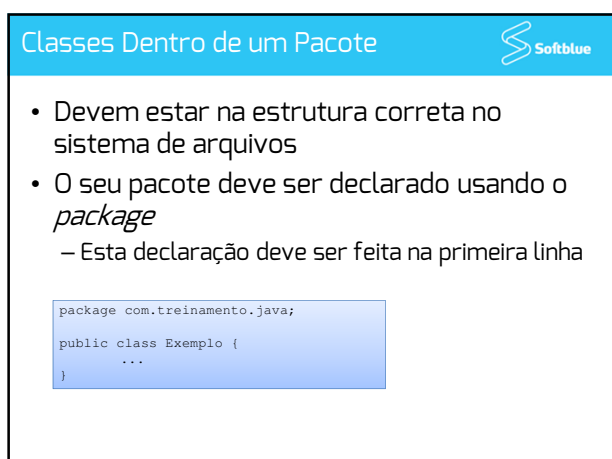


- As classes podem ser organizadas em pacotes
- Objetivos
 - Organização
 - Possibilitar que classes possam ter o mesmo nome









Convenção de Nome dos Pacotes



- Contém apenas letras minúsculas
- Normalmente é definido um nome que não terá conflito com pacotes criados por terceiros

Como Encontrar as Classes



- Usar o *fully qualified name*

```
com.java.Exemplo e = new com.java.Exemplo();
```

- Usar o import

```
import com.java.Exemplo;  
...  
Exemplo e = new Exemplo();
```

Uso do *import*



- Os imports devem ser usados logo após o uso do *package* (caso exista)
- É possível importar todas as classes de um pacote

```
import com.java.*;
```

- Importar classe por classe é preferível por questões de organização de código

A Visibilidade *package*



- Quando não definimos modificadores para classes, atributos, métodos ou construtores, eles assumem a visibilidade *package* por padrão
- Package significa que todas as classes do mesmo pacote possuem o acesso

A Visibilidade *protected*



- Quando um método, atributo ou construtor possui o modificador *protected*
 - As subclasses têm acesso ao atributo
 - Outras classes do mesmo pacote também têm acesso

Exportando o Javadoc



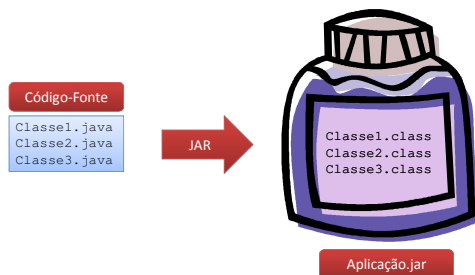
- O Javadoc é a documentação do seu projeto
 - Classes, construtores, métodos, pacotes, etc.
- Gerado a partir de comentários no código
- O Java possui uma ferramenta para exportar o Javadoc
- A própria API do Java é gerada a partir da ferramenta Javadoc

Criando JARs



- **J**ava **AR**chive
- Conjunto de classes compactadas no padrão ZIP, mas com extensão JAR
- O JAR é um componente de software
 - Agrupa código comum
 - Possui relativa independência
- O JDK possui um utilitário para gerar arquivos JAR

Criando JARs




Convenções do Código Java



- Códigos escritos em Java devem seguir algumas convenções
 - Esta padronização ajuda na manutenção do código
 - Facilita a leitura do código por outros desenvolvedores

Convenções de Nomes



• Classe e interface

– Deve ser um substantivo

– Começa com letra maiúscula e segue a notação *camel case*

class Estado

interface DVDPlayer


class CasaDeMadeira

class estado

class comprar

class casa de madeira

Convenções de Nomes



• Método

– Deve ser um verbo

– Começa com letra minúscula e segue a notação *camel case*

void comer()

int getIdade()


void processarPagamento()

void comer ()

void idade ()

int processarPagamento()

Convenções de Nomes



• Variável

– Deve ter um nome que descreva seu propósito de forma clara

– Começa com letra minúscula e segue a notação *camel case*

double nota

int qtdeItens


Casa casaDaPraia

double nota

int qtdeItens


Casa casaDaPraia

Convenções de Nomes




- Constante
 - Todas as letras são maiúsculas e o caractere "_" é usado para separar as palavras
 - A regra se aplica a elementos de enums e atributos com os modificadores *static final*

```
int VALOR
String ARQUIVO_CONFIG
enum Prioridade {
    ALTA,
    MEDIA,
    BAIXA
}
```



```
int valor
String Arquivo_Config
int PRIORIDADE_ALTA
```

Convenções para Blocos de Código

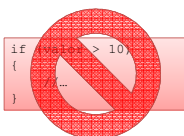


- Convenção para estruturas que usam "{" e "}" para delimitar um bloco de código

```
if (valor > 10) {
    //...
}

for (int i = 0; i < 10; i++) {
    //...
}


public class Caneta {
    //...
}
```



```
if (valor > 10)
{
    //...
}

public class Caneta {
    //...
}
```

Use a indentação adequada



7