## **Pacotes em Java**

O desenvolvimento de programas para o mundo real inclui algumas situações com as quais ainda não precisamos nos preocupar até aqui. Quando o projeto se torna maior, mais complexo, é normal que tenhamos de organizar nossas classes de forma mais ordenada em vez de deixarmos todas juntas. Para isso, utilizamos o conceito de pacote. Os pacotes nos fornecem mais que organização. Através deles, podemos definir, em um mesmo projeto, classes de mesmo nome.

## **O que são pacotes? Por que devo utilizá-los?**

Pacotes são recipientes para as classes (e para outros elementos) do nosso programa, ou seja, funcionam como pastas para os nossos arquivos do computador. Adiante, você perceberá que essa semelhança entre pacotes e pastas não é uma mera comparação. Vamos utilizar este cenário para entender o porquê da necessidade de usarmos pacotes.

Imagine que você fez um pequeno resumo após estudar a primeira aula. Depois, você criou uma pasta chamada “Resumos de Aula”. Dentro dessa pasta, colocou um arquivo chamado “Resumo”, contendo suas observações sobre a primeira aula que você assistiu. Entretanto, depois de assistir à segunda aula, você criou outro arquivo, também chamado “Resumo”, e tentou colocar na mesma pasta. O que aconteceu? Você não pode ter dois arquivos com o mesmo nome dentro da mesma pasta, por isso, a solução seria dar um novo nome a um dos arquivos. Isso se repetirá toda vez que você tentar adicionar um novo "Resumo" na mesma pasta. Para resolver, você pode criar pastas para cada aula, chamadas “Aula 1”, “Aula 2” e assim por diante, para organizar os arquivos relacionados a cada aula estudada.

De modo parecido com a situação anterior, enquanto desenvolvemos programas em Java, é comum ocorrerem situações em que o nome da classe que tentamos criar já existe. Mesmo que não tenham sido criadas por nós, quando utilizamos classes desenvolvidas por outros programadores, é possível que já existam algumas com os nomes que gostaríamos de usar.  
   
Se todas as nossas classes estiverem juntas, a única solução será renomeá-las para evitar conflitos de nomes. Os nomes dados às nossas classes são importantes, pois é o que nos falam sobre o que elas fazem ou representam, por isso, não devem ser modificados arbitrariamente. Daí, surge a necessidade de separar nossas classes em pacotes.  
   
Outro ponto importante é a organização. Quando trabalhamos com projetos de grande e de médio portes, a quantidade de classes tende a crescer bastante e, mesmo que não haja duplicidade de nomes, torna-se impraticável mantê-las juntas. Imagine uma oficina mecânica onde todas as ferramentas e todas as peças são guardadas em um mesmo lugar: chaves, porcas, discos de freio, lanternas, espelhos etc. Seria muito mais complicado e demorado encontrar algo de que se precisa. Uma boa tática seria ter um lugar para cada tipo de material. Da mesma forma que acontece com essa organização, os pacotes nos ajudam a organizar as classes pelas suas características ou pela função que desempenham no projeto, tornando o acesso a elas mais fácil e produtivo.

## **Criação de pacotes em Java**

Criar um pacote em Java é simples. Se você utiliza uma IDE como [Eclipse](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=Eclipse), NetBeans, [BlueJ](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=BlueJ) entre outras, ela, provavelmente, fornecerá uma maneira automatizada de criar um pacote. Atente para os passos a seguir, para criar um pacote em Java na oficina.

* Selecione o botão Criar Projeto;
* Crie um pacote (com o botão Criar Pacote) e o nomeie com o nome que o pacote deverá ter. Aqui, valem as mesmas regras para a definição de nome de classes: o nome não pode começar com números, não pode conter espaços ou caracteres especiais etc. O ideal é que o nome do pacote seja todo em letras minúsculas;
* Mova suas classes para dentro do pacote criado (se você já possuir alguma). Para mover, basta levar a classe para dentro do pacote criado;
* Adicione ao código-fonte de cada classe a indicação do pacote ao qual ela pertence (tanto as novas quanto as que já existirem).

Pronto! Agora, a classe está empacotada corretamente. É claro que você também pode criar seus pacotes antes de criar as classes que ficarão dentro dele. Além disso, pode mover classes entre pacotes, tendo apenas o cuidado de atualizar a definição sobre a qual pacote cada classe pertence, depois de movê-las.  
   
Vamos conferir o aprendizado?!  
   
Você está construindo um jogo em Java. Cada uma das três fases do jogo tem um ‘chefe’ com características completamente diferentes, de modo que você precisa criar uma classe para representar cada um deles. Crie a estrutura de pacotes (pastas) para as fases do jogo com a classe Chefe de cada uma dessas fases.

Solução

Você tem a liberdade de escolher o nome do pacote, baseando-se, por exemplo, no melhor motivo que levaria o pacote a agregar certas classes dentro dele. Na figura 2, iremos conferir uma das respostas possíveis para a construção da estrutura de pacotes do jogo em Java.

A figura 1 representa os passos para a criação de um pacote para uma classe já existente. O primeiro passo é entrar na pasta (ou diretório) do projeto, nesse caso. Em seguida, criar a pasta com o nome “meupacote”, O próximo passo é mover o arquivo “MinhaClasse.java” da raiz do projeto para a pasta meupacote.


## **Hierarquia de Pacotes**

Os pacotes em Java devem seguir uma hierarquia, ou seja, estar subordinados uns aos outros. Assim, conseguimos criar uma estrutura de pacotes que indicam uma especialização das classes em que eles englobam, baseados em seus nomes. Por exemplo, na figura a seguir, a classe Carro.java, contida no pacote heroi, diz respeito ao carro do herói, enquanto a outra classe Carro diz respeito ao carro dos inimigos, pois está no pacote chamado inimigos.

De fato, uma classe contida em um pacote passa a ter seu nome completo baseado no nome do pacote que a contém. Esse fato também é conhecido como namespace. Atente-se para o exemplo da figura a seguir:

Figura 3: Duas classes de mesmo nome dentro de uma hierarquia de pacotes. Imagem representa a estrutura de pastas (diretórios) de um projeto, que possui os seguintes pacotes: jogo, fase01, heroi (que possui o arquivo Carro.java) e inimigos (que também possui um arquivo Carro.java).


Perceba que, na figura 3, temos duas classes denominadas Carro, porém, por pertencerem a pacotes diferentes, elas, na verdade, são jogo.fase01.heroi.Carro e jogo.fase01.inimigos.Carro, ou seja, elas estão em namespaces diferentes. Dizemos que o pacote heroi está dentro do pacote fase01, que, por sua vez, está dentro do pacote jogocode>.

Perceba, ainda, que criamos uma sequência de nomes para os pacotes. Essa sequência de nomes nos dá uma noção de camadas, ao mesmo tempo em que cada pacote nos dá uma noção de profundidade. No nível mais alto, está o pacote jogo. Dentro dele, está o pacote fase01, que também possui classes relacionadas ao jogo, mais especificamente para a fase 1. Por fim, em um nível mais profundo, estão os pacotes heroi e inimigos, nos quais estão as classes relacionadas ao heroi e aos inimigos, respectivamente, mas apenas da fase 1 deste jogo. É dessa forma que devemos tentar nomear nossos pacotes.

## **Praticando**

Sabendo o que são pacotes e para o que eles servem, vamos agora praticar um pouco criando um projeto com alguns pacotes, para modularizar nosso projeto.

Crie um projeto com um nome à sua escolha. Crie o pacote “modelos” e o pacote “principal”. Dentro do pacote “modelos”, crie uma classe chamada “Aluno”. Nesta classe, crie os atributos nome (do tipo String), nota1 (do tipo double) e nota2 (do tipo double). Crie um construtor que receba todos os parâmetros, que atribua os valores às variáveis (por exemplo, “public Aluno (String nome, double nota1, double nota2)”). No pacote “principal”, crie uma classe chamada Principal que possua o método main() e crie também o método calcularMedia() que receba dois valores do tipo double e retorne a média aritmética deles. No método main(), crie um objeto chamado “aluno” do tipo “Aluno” e atribua os valores através do construtor (por exemplo, “Aluno aluno = new Aluno(“Pedro”, 8.75, 6.25);”). Observe que você precisará importar a classe Aluno do pacote modelos (por exemplo, “import modelos.Aluno;”). Após isso, Imprima os atributos do objeto aluno através dos métodos get e imprima a média, calculada pelo método calcularMedia().

I made the exercise on the eclipse at packages(modelos e principal).

Para conferir se você fez corretamente, os códigos das classes estão na página seguinte.

## **Visibilidade e acesso às classes empacotadas**

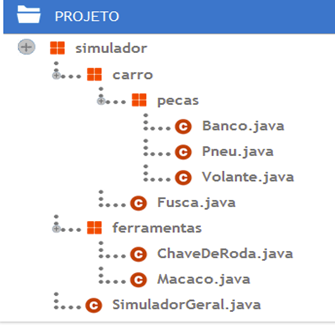
Uma vez que sua classe foi empacotada, ela passa a sofrer restrições de visibilidade. Isso quer dizer que, diferentemente de quando mantínhamos todas as classes juntas, para certos casos, precisaremos dizer ao compilador Java onde encontrar uma classe.

Vamos supor que temos uma estrutura de pacotes de um projeto para construir um programa simulador de carros. Pensando na questão da organização, criaremos um pacote para as classes que representam peças desse carro. Dentro do pacote simulador, temos a classe SimuladorGeral e os subpacotes carro e ferramentas. Dentro do pacote carro, temos o subpacote pecas (representando peças) e as classes Banco, Pneu e Volante. Dentro do pacote ferramentas, temos as classes ChaveDeRoda e Macaco.

Perceba que o pacote pecas está subordinado ao pacote carros. Nesse caso, toda classe, no pacote carros, pode “enxergar”, ou seja, acessar as classes do pacote pecas. Já a classe Fusca, só poderá “enxergar” as classes dentro do pacote ferramentas que forem públicas, pois os pacotes carros e ferramentas estão no mesmo nível. Vamos formalizar as regras?  
  
 1. Classes em um pacote podem acessar qualquer classe em seus subpacotes, ou seja, em níveis inferiores a ela.  
  
 2. Classes dentro de pacotes de mesmo nível são visíveis entre si apenas se a classe buscada for pública.

Vamos conferir o aprendizado!  
  
Utilizando como referência a figura 4, verifique se:

* 1. A classe simulador.SimuladorGeral pode “enxergar”, ou seja, acessar a classe simulador.ferramentas.Macaco.
  2. A classe simulador.ferramentas.ChaveDeRoda pode acessar a classe simulador.carro.pecas.Pneu, considerando-se que ela (Pneu) é pública.



Resposta  
  
 Os itens 1 e 2 são verdadeiros. No primeiro, a classe SimuladorGeral enxerga qualquer classe do projeto, pois ela está no nível mais alto da hierarquia de pacotes. No segundo, a classe ChaveDeRoda pode acessar Pneu, contanto que essa classe seja pública.

## **Acessando classes empacotadas**

Você se lembra de quando tínhamos todas as classes juntas no diretório raiz do projeto? Na verdade, o Java considerou que elas estavam dentro de um pacote especial, implícito, ou seja, que não precisava ser declarado, o pacote default (padrão). Quando colocamos mais de uma classe em um mesmo pacote, elas se “enxergam” normalmente, pois estão em um mesmo nível. Para prover o acesso às classes fora do seu pacote, temos duas opções:

Especificar o nome completo da classe (pacote + nome da classe) a cada vez que for necessário referenciá-la;

Utilizar a diretiva [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) para tornar a classe (ou um pacote inteiro) acessível apenas pelo seu nome.

Vamos continuar utilizando o exemplo da figura 4. Vamos escrever o método principal da classe SimuladorGeral e tentar instanciar nele um objeto do tipo Fusca chamado carroSimulado. Se escrevermos a menção à classe Fusca apenas pelo seu nome, sem indicar o pacote ao qual ela pertence, seremos advertidos de que a classe não foi encontrada, conforme mostra o exemplo a seguir.

1 package simulador;

2 public class SimuladorGeral{

3 public static void main(String args[]){

4 Fusca carroSimulado = new Fusca();

5 }

6 }

Console

Erro linha: 4 coluna: 7

Mensagem 1: \simulador\SimuladorGeral.java:4: cannot

find sybol symbol: class Fusca

location: class simulador.SimuladorGeral

Embora a classe SimuladorGeral possa “enxergar” a classe Fusca por estar em um nível superior, é preciso, como foi dito antes, fornecer o nome completo da classe (pacote + nome) para acessá-la em SimuladorGeral. Assim, corrigiremos o erro adicionando o namespace do pacote simulador.carro. Note:

package simulador;

public class SimuladorGeral{

public static void main(String [] args){

simulador.carro.Fusca carroSimulado =

new simulador.carro.Fusca();

}

}

Porém, convenhamos que mencionar o nome completo da classe é um tanto inconveniente. Em vez disso, o Java nos dá a possibilidade de "importar” uma ou mais classes para o nosso código-fonte. Na verdade, o que ocorre é que, quando utilizamos a diretiva [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) seguida do nome da classe ou do pacote alvo, estamos instruindo o compilador para onde ele deve procurar por classes que mencionamos no código-fonte. Agora, incluiremos o comando [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) simulador.carro.Fusca; ao início do nosso código, logo após a diretiva package. Pronto, a partir daí, poderemos nos referenciar à classe Fusca apenas pelo seu nome, sem precisar mencionar o seu pacote. O resultado final será assim:

package simulador;

[import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) simulador.carro.Fusca;

public class SimuladorGeral{

public static void main(String[] args){

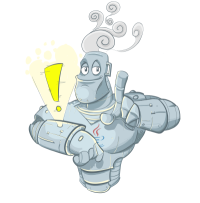
Fusca carroSimulado = new Fusca();

}

}

Além de permitir que façamos referência a uma única classe, a diretiva [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) também serve para incluir um pacote inteiro. Por exemplo, se, na classe SimuladorGeral, adicionarmos [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) simulador.carro.peças.\*; , teremos acesso a todas as classes dentro do subpacote pecas, ou seja, poderemos utilizar as classes Banco, Pneu e Volante. Este é um recurso que evita que tenhamos que “[import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import)ar” uma classe por vez.

Por exemplo, se adicionássemos [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) simulador.carro.pecas.Fusca; em vez do que foi proposto anteriormente, teríamos acesso apenas à classe Fusca; o subpacote pecas e todas as classes que ele comporta não seriam inclusos. Esse uso do [import](https://leadfortaleza.com.br/ead2pcd/app/glossario?termo=import) não implica em lentidão ou aumento do tamanho do programa.



O uso do asterisco (\*) serve para tornar visíveis apenas as classes do pacote mencionado antes dele, ou seja, não inclui classe dentro de subpacotes.