01

Enxugando seu código

Transcrição

[00:00] Voltando para o nosso código, reparem que estamos fazendo tudo em um arquivo só, que é um pouco feio de trabalhar.

[00:12] Vamos separar os nossos arquivos de acordo com as nossas classes. Vamos manter no arquivo livro.rb, a classe livro, e vamos criar um arquivo chamado sistema.rb, que vamos colar todo o resto.

[00:21] Eu seleciono todo o resto e jogo no arquivo “sistema.rb”, eu preciso adicionar no meu arquivo “sistema.rb”, o arquivo “livro.rb”. Então require, “livro.rb”, não precisa do “rb”, ee só coloco require “livro”. Eu tento rodar a minha aplicação e deu erro, ele não encontra o arquivo livro. Isso porque eu preciso falar exatamente onde está o arquivo “livro”, ou setar um path de procura.

[00:35] Vamos usar o “require relative”, pra dizer que é relativo ao meu arquivo. Então, “require_relative livro”.

[00:56] Executo de novo o meu programa, agora está tudo certo. Repara que ainda temos outros arquivos pra extrair, por exemplo, o contador. Então eu crio um arquivo chamado “Contador.rb”, pego o código do nosso contador e jogo lá.

[01:10] Salvo, volto no sistema, e dou de novo: “require_relative “contador””. Rodo a nossa aplicação, tudo certo. Vamos extrair mais um, estoque. Seleciono todo o estoque: “require_relative “estoque””, crio o arquivo “estoque.rb”, colo tudo lá dentro, e perfeito. Mas repara que dentro do arquivo estoque, ele precisa de uma referência para o contador.

[01:43] Seria arriscado alguém adicionar o estoque sem adicionar o contador. Por isso dentro do arquivo “estoque”, eu vou colocar um “require_relative” do contador.

[01:51] Voltando para o nosso sistema, onde damos o require do livro, contador e estoque, eu não preciso mais dar require em contador. Por quê? Porque o nosso estoque vai automaticamente requerer tudo que ele precisa. Que no nosso caso, é o contador.

[02:03] Rodo o meu programa novamente e está tudo certo. Nosso sistema tem que suportar também quem é a editora de um livro.

[02:15] Então lembra dos outros capítulos? Adicionamos um reader, se for necessário, para a editora, e adiciona também, um consultor a editora, atribuindo valor para a variável membro, ao atributo.

[02:24] Agora a minha empresa quer construir um ranking, de qual é o título de livro mais vendido. Qual é o título de revista mais vendida? Qual é o ano de lançamento de livros mais vendidos? Qual é a editora que possui mais revistas vendidas, por exemplo.

[02:38] Então para isso, o nosso método “remove” do estoque, vira um método de venda, afinal eu estou vendendo do estoque, não removendo. E eu vou atribuir isso em uma variável de vendas, uma array de vendas. Lembra do nosso initialize? Vamos criar lá a nossa array de vendas também.

[02:56] E vamos criar algum método que calcula a quantidade de vendas que foram feitas de um título específico.

[03:07] Por exemplo, eu passo um título de um produto, de um livro, uma revista ou qualquer coisa, e vou na minha array de vendas e conto quantas vendas tiveram o título do meu produto. Isto é, quantas vezes esse produto foi vendido.

[03:22] E agora eu quero saber qual foi o livro que mais vendeu, por título. Isto é, qual é o título mais vendido. Então eu vou ordenar as minhas vendas, comparando a capacidade de venda de cada um dos livros. Eu verifico se o primeiro livro foi vendido mais ou menos do que o outro.

[03:43] Então eu comparo a quantidade de vendas do título 1, com a quantidade de vendas do título 2, e eu devolvo o último deles. O que teve maior número de vendas.

[03:52] Venho adicionar as editoras, de cada um desses livros, do livro de algoritmo, de arquitetura, de programmer e de ruby, adiciono as editoras que eu bem entender, vou tirar todas essas impressões, não estou interessado nelas.

[04:08] Vou adicionar o livro de algoritmos duas vezes, o livro de ruby uma vez, o livro de programmer, de arquitetura, e mais duas vezes o livro de ruby.

[04:18] Eu vendo três vezes o livro de ruby, e duas vezes o livro de algoritmos. Eu quero imprimir o livro que mais vendeu, por título. Vou imprimir o título dele.

[04:30] Rodamos o programa e ele imprime o livro de ruby. E se eu vender só um livro de ruby? Eu rodo de novo e o mais vendido é o livro de algoritmos.

[04:42] Agora eu quero saber o livro que mais vendeu por ano de lançamento. Eu vou ordenar de novo e agora ao invés da quantidade de vendas e título, seria pelo ano. Pelo ano de lançamento.

[04:55] Então poderia já generalizar essa chamada desse método, pra passar qual é o método que eu gostaria de chamar. Eu gostaria de chamar o método ano de lançamento, então eu faço um lambda com o método ano de lançamento.

[05:10] E no caso de cima, o lambda do método “título”, alteramos o método pra receber o nosso lambda, que é o campo que vamos chamar, chamamos esse campo no nosso produto, e chama esse campo na nossa venda.

[05:26] No caso de ano de lançamento, ele vai se chamar “venda.ano_lancamento”, no caso de título ele vai se chamar “venda.titulo”. Tudo isso através desse parâmetro, que é o nosso lambda, que é o bloco que está sendo passado.

[05:42] Lembrando desses dois pontos que eu esqueci aqui, que é o símbolo que é passado para o lambda. Rodamos o programa de novo e tudo funciona.

[05:51] E agora, queremos o livro ordenado pela editora, mais vendido pela editora: “por_editora”. Fica bem mais fácil, teríamos que executar aquele método “quantidade_de_vendas_por”, basta passar o lambda e fazer a invocação de editora.

[06:04] Só que repara que esse método é sempre igual: “vendas.sort”, “vendas.sort”. A única coisa que muda é: editora, ano de lançamento e título.

[06:16] Vamos refatorar, vamos extrair um método, criar um método que chama: “livro_que_mais_vendeu_por(&:editora)”, e aí eu falo qual é o lambda que eu quero passar.

[06:27] Vamos definir esse método: “def livro_que_mais_vendeu_por (campo)”, recebe o campo, colo aquele método das vendas e passo o campo pra frente. Repara que quando eu recebo o campo, eu tenho que falar que ele é um bloco, pode ser um lambda, etc. Então, & comercial nele.

[06:42] Todos os outros lugares, é só chamar agora o método “livro_que_mais_vendeu_por”, isto é, extrair um método dentro do nosso objeto ajuda muito a manter o código de uma maneira mais simples.

[06:52] Quando eu quero passar um bloco pra frente, eu preciso passar com o & comercial, de novo. Então "(& campo)", rodamos o programa de novo, e tudo funciona. O que eu quero fazer agora, é: além de livros, estamos vendendo também, ebooks e revistas.

[07:14] Então eu vou criar um campo, dentro da classe livros, pra representar que tipo de objeto é esse que estamos vendendo: é um livro? É uma revista? É um ebook?

[07:23] E da mesma maneira que temos um ranking de livros, eu vou querer agora um ranking de ebook, e um ranking só de revistas. Então eu poderia copiar todos esses métodos, mudar para revista.

[07:38] Nos métodos que eles chamam, também vão se chamar revista, e duplicamos esse método para "revista". No caso do livro, estamos interessados em filtrar, selecionar somente as vendas, cujo o tipo é um livro, e fazer o ranking.

[08:01] No caso da revista, queremos filtrar, queremos selecionar somente aqueles cujo o tipo é revista, e fazer o ranking. Vamos marcar os nossos quatro livros como livros, e vamos criar uma revista, a Revistona: "revistona = Livro.new ("Revista de Ruby", 10, 2012, true, "Revistas", "revista").

[08:34] Adiciono a revistona duas vezes ao meu estoque, efetuo uma venda dela, e mando imprimir a revista que mais vendeu por título. Rodo o programa, e ele imprime a Revista de Ruby.

[08:52] Mas repara que o método, "livro que mais vendeu por" e revista que mais vendeu por" são quase iguais, só mudam uma string. O que fizemos nesses casos? Extrai um método, e recebe essa string como parâmetro. É bem comum fazer isso.

[09:05] Então extraímos o método "quem_mais_vendeu_por", recebe o título que estamos interessado como parâmetro, e faz a verificação. Agora, "revista_que_mais_vendeu_por", vai passar revista. Então passamos a chamar método "que_mais_vendeu_por", nos três casos que temos revista.

[09:25] E nos casos que chamamos para o livro, chamamos o método de "que_mais_vendeu_por", passando livro como parâmetro. De novo, nos três métodos.

[09:41] Rodamos o programa e está tudo certo. Repara que simplificamos de novo: extrair um método da nossa classe ajuda a manter o nosso código.