10

Validação

Transcrição

[00:00] Novamente nossa função joga ficou bem complicada, porque eu saí escrevendo onde eu podia. Primeiro vou extrair todos os ifs para uma função que me diz se a nova posição é válida ou não. Uma função que pergunta e recebe um true ou false recebe ponto de interrogação. Quero saber se nesse mapa a nova posição é válida ou não. Como na última linha ele retorna, colocamos um true.

[01:23] Se minha linha for menor que zero ou maior igual ao mapa.size, a posição não é válida. Então coloco || para representar o ou. Costuma ser no teclado Shift e barra invertida. Se todos os casos forem bonitinhos, aí ele retorna verdadeiro, porque a posição é válida.

[02:25] Podemos tentar simplificar mais o código, chamando o primeiro cara de linhas e o segundo de colunas. Posso também querer saber se você estourou as linhas e se estourou colunas no outro caso. Também podemos simplificar o código mais ainda. Se você estourou linhas ou colunas, retorno falso. Se a posição no mapa é um x, também retorno falso. E por fim retorno verdadeiro se a posição é válida de verdade.

[03:12] Agora que tenho a posição válida mais bonitinha, posso também renomear a variável nova posição. Nova posição só fazia sentido no método da função joga. Aqui vou chamar direto de posição, porque estou validando uma posição.

[03:32] Dá para refatorar bastante esse código para ficar ainda mais bonito. Fica como exercício fazer várias refatorações nele.

[03:41] Na nossa função do joga, também queremos saber se a posição é válida. If posição válida, vou querer andar. Senão, não ando. Se a posição não é válida, passo o mapa e a posição. Lembra que o Ruby tem aquela sintaxe que para muitos é idiomática, que é o next if numa linha só. Eu acredito que aqui fica um pouco mais obscuro o que está acontecendo, por isso preferi usar não posição válida, então ele continua. Senão ele escreve troca posição do herói.

[04:30] Cuidado com erros de digitação. Passamos a invocar a função posição válida, implementamos, e depois que ela foi extraída Refatoramos o código lá dentro. Lembrando de só refatorar depois que o código está ok, porque aí ele tem que continuar funcionando. Se refatorarmos sem saber se ele funciona, corremos o risco de errar a implementação.