05

Comportamentos

Transcrição

[00:00] Como vimos, depois de passar a utilizar a classe herói, quando refatoramos o retorno da nossa função, quebramos o código de utilizar o retorno da função. No momento não, por exemplo, quando perguntávamos se o jogador perdeu. Mas em outro momento sim, quando tentamos usar para recalcular a posição. Depois que usamos a posição para mapear, para limpar o lugar do mapa. Em vários lugares quebramos quando refatoramos nosso código. É muito importante sempre analisar, entender uma mudança que vamos efetuar no retorno de uma função antes de fazer ela, porque podemos quebrar diversas partes do nosso código. Uma linguagem de programação que tem fase de validação, algo do gênero, tenta passar garantias durante a refatoração. Posso usar uma ferramenta um pouco mais inteligente do que o editor de texto, que possa fazer essa refatoração para mim.

[01:10] Em linguagens compiladas isso é mais fácil. Linguagens dinâmicas, não compiladas, fica mais difícil de detectar. No nosso caso, é importantíssimo que verifiquemos todo mundo que está usando esse herói, senão nosso código não vai funcionar, e para isso precisamos de garantias. Pode ser testes, ferramentas extras, seja o que for. No nosso caso, vamos ter que passar todo mundo que usa o herói.

[01:40] A nova posição é usada tanto na hora de validar quanto na hora de usar para limpar o mapa e colocar a posição do herói de novo. Vamos ter que alterar essas duas situações.

[01:52] A primeira situação é o posição válida. Ele verifica se a posição é válida ou não. Podemos mudar tudo isso para passar a usar o herói. Mas antes de refatorar a função, vamos ver quem chama ela. Ela é chamada tanto para o herói, quanto para um fantasma. Não quero alterar ela para que ela só funcione com o herói. Teria que alterar de uma maneira que funcionasse com os dois.

[02:25] O nosso foco vai ser resolver esse problema com uma abordagem de linguagem de programação. Outras técnicas podemos ver quando falarmos de orientação do objeto. Nesse nosso caso, queremos continuar suportando com a posição válida o array de duas posições. Quando chamo essa função, quero transformar a nova posição em um array de duas posições.

[03:01] Nós já transformamos inteiro em string, string em inteiro. Sempre tem um padrão quando vou transformar de um formato de objeto em memória para outro através de um to ou alguma coisa.

[03:14] Vamos falar que nossa nova posição tem um to_array. Não vamos abreviar o array com uma letra tosca. Se estou chamando um método da posição do objeto, preciso definir esse método no nosso herói. Vou definir o método to_array, que vai devolver um array com duas posições. Ele vai retornar um array com linha e coluna. Lembra que o retorno é opcional? É só devolver o array com linha e coluna.

[04:02] Agora nosso herói tem dois comportamentos. Um para calcular uma posição nova e devolver o herói em outra posição, e outro que é o to_array, que transforma ele em um array para você poder usar numa função que recebe um array.

[04:12] Voltando para o foge foge, estávamos usando o nova posição com zero e um. Vamos trocar para linha e coluna.