04

Funcionalidades

Transcrição

[00:00] Refatoramos quem usava uma estrutura que era um array de duas posições para passar a usar nossa estrutura, que é um objeto agora. Podíamos ter vários objetos do tipo herói. O calcula nova posição recebe um herói e a direção. Ele usa a direção só uma vez e o herói quatro vezes. Das seis instruções que existem no código, quatro delas estão ligadas ao herói. Esse código pertence ao herói. Se tudo que ele faz é um monte de coisa com o herói, ele pertence a um herói. Chamamos ele de invejoso. Ele é invejoso das funcionalidades do herói. É a feature envy. Inveja de uma funcionalidade, que é um code smell. Nós temos que evitar isso. Nós temos que jogar esse cara onde ele pertence. Essa nossa função pertence ao herói.

[02:11] Para eu poder chamar a função método dentro de um objeto, preciso invocar, preciso de um herói. É dentro dele que vou fazer essas coisas. Sendo assim, esse código, essa função já está sendo executada dentro de um herói. Nesse caso, chamamos esse cara de método, porque ele só pode ser executado dentro do meu herói.

[02:38] Se ele só pode ser executado dentro do herói, não preciso receber o herói como argumento. Só o herói é capaz de calcular sua nova posição. Mas se a variável herói não existe mais, quem é esse cara que quero duplicar? Sou eu mesmo. Se eu estou invocando esse método em um objeto, em um herói determinado, quem é o herói que quero duplicar? Eu mesmo. Eu mesmo é self. Self.dup. Duplico eu mesmo, e agora tenho um novo herói. Troco a linha, a coluna, e devolvo esse novo herói.

[03:35] Eu movi minha função para dentro da classe herói, tive um método com isso, não preciso mais receber o herói, porque o herói é o próprio cara. Se ele é o próprio cara, ao invés de receber como argumento, eu falo self, eu mesmo, self.dup. Dupliquei o cara para a variável herói, que é nova. Continuo com as palavras herói, mas está dentro da classe herói. Um herói conhece como ele funciona. Não o cara lá de fora que não tem nada a ver com ele.

[04:27] Movemos o calcula nova posição para cá. Agora como faço para invocar essa função? Dizemos para o herói calcular a nova posição. Chama a função, o método, calcula nova posição dentro do herói específico. Chamo a função ali dentro e passo a direção como argumento.

[05:07] Nosso objeto herói não só tem mais atributos de linha e coluna, como tem comportamento, ele se comporta como herói, ele se move como herói. Ele tem tanto atributos quanto comportamentos de herói.

[05:27] Falta voltar ao herói e falar que se eu invocar o método em mim mesmo, não preciso falar o self. O self é opcional. Posso falar duplica eu mesmo. Chamo o método dup no próprio herói e duplicar o herói. Agora referenciamos esse herói com uma variável nova. Tenho um novo herói, cuja linha e coluna são diferentes do herói antigo, dupliquei, troquei a linha e coluna e devolvi esse novo herói. E aí continuo com meu trabalho.

[06:17] Nosso próximo passo é usar a nova posição como herói, e não mais como array simples.