Separação de interesses

A seguir, listo a separação de interesses classe por classe. Por fim, informo que ainda estão sujeitas a modificação.

1. **Screen**

   Esta classe é uma boundary, ou seja, ficará responsável por eventos relacionados a impressão de dados e mensagens. Ainda não foi decidido se essa classe ficará responsável por todas as mensagens, tais como mensagens de erros por exemplo. Inicialmente, ela tem 3 classes, são elas:
   a. intro: responsável pelas mensagens do inicio, tal como “Bem vindo!” e etc;
   b. qtsPlayers: responsável pela pergunta de quantos jogadores existem.
   c. nearestSL: responsável pela mensagem que diz qual a snake e ladder mais pròxima do jogador.

2. **Control**

   Esta classe é um control, ou seja, ficará responsável pelo gerenciamento. Nessa implementação, não haverá sequência de jogos, ou seja, só há um jogo por execucção. Inicialmente, ela tem 4 classes, são elas:
   a. intro: responsável por chamar mensagens da screen entre outro coisas.
   b. nameAllotment: nomear os players
   c. initGame: inicar o jogo de fato.

3. **Game**

   Esta classe é uma entity, nela toda a lógica do jogo ocorrerá. Inicialmente, ela tem 2 classes, são elas:
   a. getWon: responsável por dizer quem foi o ganhador
   b. game: responsável por toda a lógica do jogo

4. **Board**

   Esta classe é uma entity, nela todas as questões do board do jogo ocorrerão. Inicialmente, ela tem 2 classes, são elas:
   a. Board: construtor que criará todos os tiles do board e guardará na mesma.
   b. check: responsável por instanciar as snakes e ladders do board.
5. **Player**

   Esta classe é uma entity, nela todas as questões do player ocorrerão. Inicialmente, ela tem 2 classes, são elas:
   a. imput: responsável por pegar a entrada do jogador e rodar o dado que será passado por parâmetro.
6. **Dice**

   Esta classe é uma entity, nela todas as questões do dice ocorrerão. Inicialmente, ela tem 1 classes, são elas:
   a. roll: responsável por devolver um lado do dado (inteiro).
7. **Obstacle**

Esta classe é uma entity, nela todas as questões do obstacle ocorrerão. Ela é uma classe abstrata que será pai de Snake e Ladders. Isso foi feito para diminuir linhas de código, visto que Snake e Ladders são parecidas. Havia dúvidas em como fazer esse pull, se deveria estar no player, ou seja ele fica responsável por dizer se está em uma snake ou em uma ladder. Porém, a classe obstacle ficaria sem métodos e, além disso, a classe player teria mais um método para lidar. Como é um evento referente às duas classes e por toda a explicação dada, decidi colocar esse método em obstacle. Inicialmente, ela tem 1 classes, são elas:

a. pull: responsável por eventos de subir ladder e descer snake

8. Snake

Esta classe é uma entity, nela todas as questões da snake ocorrerão. Inicialmente, ela tem 1 classes, são elas:

a. pull: responsável pelo evento descer snake.

9. Ladder

Esta classe é uma entity, nela todas as questões da ladder ocorrerão. Inicialmente, ela tem 1 classes, são elas:

a. pull: responsável pelo evento subir ladder.

10. Tile

Esta classe é uma entity, serve apenas para guardar informações de coordenadas, pois um board tem várias delas.

11. wildcard

Esta classe é uma entity, nela todas as questões da wildcard ocorrerão. A wildcard é uma carta especial que contem como atributo “effect” que possui um valor inteiro que pode ser negativo ou positivo. Se o player cair no mesmo tile que há uma wildcard ele andará de acordo com atributo.